北京邮电大学计算机学院

《Python 程序设计》大作业报告

学　　号：2020211435

姓　　名：何家豪

班　　级：2020211308

授课教师：王晶

完成日期：2022 年 12 月 30 日

目录

第一部分　实验目的

- 编写爬虫程序获取链家官网上 5 个城市的相关租房数据
- 对获取到的数据进行数据分析，包括：
  - 比较 5 个城市的总体房租情况，包含租金的均价、最高价、最低价、中位数等信息，单位面积租金（元/平米）的均价、最高价、最低价、中位数等信息并采用合适的图或表形式进行展示。
  - 比较 5 个城市一居、二居、三居的情况，包含均价、最高价、最低价、中位数等信息。
  - 计算和分析每个城市不同板块的均价情况，并采用合适的图或表形式进行展示。
  - 比较各个城市不同朝向的单位面积租金分布情况，采用合适的图或表形式进行展示。对于不同城市之间房屋朝向以及价格之间的关系进行分析。
- 结合其他相关数据进行分析，包括：
  - 查询各个城市的人均 GDP，分析并展示其和单位面积租金分布的关系
  - 查询各个城市的平均工资，分析并展示其和单位面积租金分布的关系。
- 结合上述数据和分析，自主设计数据分析主题，补充查找或爬取相关数据，完成题目设计、数据获取、数据分析及数据展示过程。

第二部分　实验环境

- 操作系统：Windows 11
- Python 解释器：Python 3.10.3
- 使用的相关第三方库的版本：
  - Scrapy 2.7.1
  - pandas 1.5.2
  - matplotlib 3.6.2
  - pyecharts 1.9.1

第三部分　实验内容

3.1　数据获取

3.1.1　类设计

RentItem

用于保存爬取到的每一条数据信息：

```
class RentItem(scrapy.Item):
    ID = scrapy.Field()
    租金 = scrapy.Field()
    面积 = scrapy.Field()
    房型 = scrapy.Field()
    板块 = scrapy.Field()
    朝向 = scrapy.Field()
```

MySpider

爬虫类，用于执行主要的爬取操作，在其内部定义了爬虫名字，允许访问的域名和起始 url。其中的成员函数用来爬取不同页面上的信息：

```
class MySpider(scrapy.spiders.Spider):
    name = 'rent'
    allowed_domains = ['bj.lianjia.com']
    start_urls =  ["https://bj.lianjia.com/zufang/"]
    def parse(self, response):
        pass
    def parse_block(self, response):
        pass
    def parse_page(self, response):
        pass
    def parse_info(self, response):
        pass
```

RentPipeline

处理爬取到的数据，主要负责文件写入：

```
class RentPipeline:
    def open_spider(self, spider):
        pass
    def process_item(self, item, spider):
        pass
    def close_spider(self, spider):
        pass
```

3.1.2 爬取数据

在本部分，均以北京为例。

设置筛选条件

经过观察发现，链家官网的租房数据最多只显示 100 页，即 3000 条。而实际上每个城市都有至少 30000 条数据，所以为了获取到每个城市的全部数据需要设定合理的筛选条件进行分批爬取。在设定筛选条件的时候主要考虑两点：

1. 筛选条件应当互斥，以保证相同的条目不会被爬取两遍；
2. 筛选条件应当在保证每次筛选结果不大于 3000 条的基础上尽可能地宽松，以保证每批爬取尽可能多的数据，从而尽量少分批。

出于以上考虑，决定采用不同板块进行筛选。

访问市级信息获取区级 url

先访问到城市的主页面，从主页面中找到这个城市中所有的区，然后再根据每个区的 url 访问到每个区的主页面：

```
class MySpider(scrapy.spiders.Spider):
    ...
    def parse(self, response):
        base_url = 'https://bj.lianjia.com'
        # 找到所有的区
        districts = response.xpath('//li[@data-type="
      district"]/a/@href').extract()
        for d in districts:  # 将区的href拼接到base_url后面形
      成新的url
            d_url = base_url + d
```

```
            print(d_url)
            yield scrapy.Request(d_url, self.parse_block)
        # 回调函数parse_block处理区级的信息
```

访问区级信息获取板块级 url

然后在每个区内再获取到该区所有的板块，再根据每个板块的 url 访问到每个板块的主页面：

```
class MySpider(scrapy.spiders.Spider):
    ...
    def parse_block(self, response):
        base_url = 'https://bj.lianjia.com'
        # 找到所有的板块
        blocks = response.xpath('//li[@data-type="bizcircle
    "]/a/@href').extract()
        for b in blocks:
            b_url = base_url + b  # 将板块的href拼接到
    base_url后面形成新的url
            yield scrapy.Request(b_url, self.parse_page)  #
     回调函数parse_block处理板块级的信息
```

访问板块级信息获取板块内房屋出租数量

然后在板块的主页面上获取到该板块共有多少套房屋出租，并根据出租房屋数量计算出总共需要爬取的页数，将页码拼接到 url 后面依次爬取每一页的信息：

```
class MySpider(scrapy.spiders.Spider):
    ...
    def parse_page(self, response):
        # 共有house_num套房屋出租
        house_num = eval(response.xpath('//span[@class="
    content__title--hl"]/text()').extract()[0])
        k = 1
        while house_num > 0:
            p_url = response.request.url + 'pg' + str(k)
            yield scrapy.Request(p_url, self.parse_info)  #
     回调函数parse_info处理每一页上的信息
            # 每爬取一页剩余房屋数量-30，页数+1
            house_num -= 30
            k += 1
```

爬取每一页上的有效信息

为了后续的数据分析与展示，我们需要爬取每一页上所有有效房屋的 data-house_code（相当于房屋 ID，用于后续去重）、租金、面积、房型、板块信息。首先我们观察每一页含有有效信息标签内的结构，大致如下所示（已略去无用信息）：

```
<div class="content__list">
    <div class="content__list--item" data-c_type="1"
        data-house_code="BJXXXX">
        <div class="content__list--item--main">
            <p>...</p>
            <p class="content__list--item--des">
                <a>房山</a>-<a>长阳</a>-<a>首开熙悦山澜庭</a>
                <i>/</i>
                98.00平米
                <i>/</i>东南 北
                <i>/</i>
                3室1厅2卫
                ...
            </p>
            <p>...</p>
            <p>...</p>
            <span class="content__list--item-price"><em>
        5500</em> 元/月</span>
        </div>
    </div>
</div>
```

在爬取对应信息时，基本上可以参照以上的结构来设计 xpath 获取到正确的数据，但是在实际操作过程中遇到了些许特例需要特殊处理，在下面会一一介绍。

首先提取出所有房屋信息的上层标签，组织为item_list：

```
def parse_info(self, response):
    item_list = response.xpath('//div[@class="content__list
        --item" and @data-c_type="1"]') # 设置data-c_type为了
        避免获取到“猜您喜欢”部分的内容
```

因为是按照不同板块进行爬取的，所以板块信息就是板块部分高亮显示的文字内容：

```
block = response.xpath('//li[@data-type="bizcircle" and
      @class="filter__item--level3  strong"]/a/text()')
      [0].extract()
```

然后对于item_list中的每个item进行具体的处理。

获取 ID：

```
for item in item_list:
        info = RentItem()
        info['ID'] = item.xpath('@data-house_code')[0].
      extract()
        ...
```

获取租金：

```
for item in item_list:
    ...
    info['租金'] = eval(item.xpath('./div/span[@class="
       content__list--item-price"]/em/text()')[0].extract()
       .split('-')[0])
    ...
```

获取到价格标签内的价格后，要按照'-' 号进行分割，因为有些房屋的价格是一个区间。遇到这种情况时，价格以区间下限为准。

获取 des 标签：

```
for item in item_list:
    ...
    des = item.xpath('./div[@class="content__list--item--
       main"]/p[@class="content__list--item--des"]')
    des_text = des.xpath('text()').extract()
    ...
```

之后的其余四条信息需要在 des 标签下进一步获取。

获取面积、朝向、户型信息：

```
...
try:
    info['面积'] = eval(des_text[-3].split('平米')[0].split(
       '-')[0].strip())
    info['朝向'] = des_text[-2].strip()
    if info['朝向'][0].isdigit():
        info['朝向'] = ''
    info['房型'] = eval(des_text[-1].strip()[0])
except:
```

```
    info['面积'] = eval(des_text[-4].split('平米')[0].split(
        '-')[0].strip())
    info['朝向'] = des_text[-3].strip()
    if info['朝向'][0].isdigit():
        info['朝向'] = ''
    info['房型'] = eval(des_text[-2].strip()[0])
...
```

对于一般情况，try 模块中的内容就可以获取到正确的信息，但是需要注意的是，有些时候对应条目不会展示房屋的朝向信息，如下图所示：

此时就需要设定房屋朝向为空。除此之外，有时房屋信息中会包含有“精品”字样，此时后续的所有信息都会错位，需要在 except 模块中调整索引，如下图所示：

将 info 数据对象 yield 给 pipeline 写入文件：

```
yield info
```

3.1.3 文件写入

在RentPipeline中将数据一条一条写入 csv 文件中：

```
class RentPipeline:
    def open_spider(self, spider):
        try:
            self.file = open('bj.csv', 'w', encoding='utf-8
        ', newline='')
        except Exception as err:
            print(err)
        self.writer = csv.writer(self.file)
        self.writer.writerow(['ID', '租金', '面积', '房型', '
        板块', '朝向'])
    def process_item(self, item, spider):
        dict_item = dict(item)
```

```
11          self.writer.writerow([dict_item['ID'], dict_item['
            租金'], dict_item['面积'],
12                dict_item['房型'], dict_item['板块'], dict_item[
            '朝向']])
13          return item
14      def close_spider(self, spider):
15          self.file.close()
```

3.1.4 数据概览

bj.csv

共 31546 条数据（节选 10 条）：

```
1  ID,租金,面积,房型,板块,朝向
2  ...
3  BJ1708799054017724416,4800,82.0,3,滨河西区,南
4  BJ1713475362458959872,2990,45.32,1,滨河西区,南
5  BJ1710661033976135680,3200,59.43,1,滨河西区,南
6  BJ1709468751704883200,3400,80.0,2,滨河西区,南 北
7  BJ1717395262508367872,2500,65.0,2,滨河西区,南 北
8  BJ1709096614435487744,3700,94.0,2,滨河西区,南
9  BJ1705452389688934400,5000,95.0,3,滨河西区,南 北
10 BJ1710505706802642944,3400,61.0,2,滨河西区,南
11 BJ1718132299540725760,3800,65.0,2,滨河西区,西
12 BJ1718627944668069888,2500,49.0,1,滨河西区,南 北
13 ...
```

sh.csv

共 29846 条数据（节选 10 条）：

```
1  ID,租金,面积,房型,板块,朝向
2  ...
3  SH1715360165034721280,1500,65.7,2,金山,南
4  SH1717092511052201984,3400,86.13,2,金山,南
5  SH1706953498060390400,2600,85.67,3,金山,南
6  SH1708397729513734144,3000,99.33,2,金山,南 北
7  SH1697474447167979520,4200,93.09,3,金山,南
8  SH1708075326422122496,3000,87.8,2,金山,南 北
9  SH1708681220205838336,3000,87.91,2,金山,南 北
```

```
SH1704048260273930240,4000,139.75,3,金山,南
SH1710493888872972288,3000,93.66,3,金山,南
SH1690207683895885824,2600,88.4,2,金山,南
...
```

gz.csv

共 87402 条数据（节选 10 条）：

```
ID,租金,面积,房型,板块,朝向
...
GZ1623557686433939456,2400,120.0,3,旧区,南
GZ1624384702188093440,1380,97.6,3,旧区,东
GZ1655949806419312640,2500,105.56,3,旧区,南
GZ1656604341634596864,2000,110.0,3,旧区,东南 南
GZ1718174783369117696,2000,59.0,1,旧区,东
242447,580,45.0,1,旧区,
GZ1707354409647734784,1500,53.49,1,旧区,西
201507,499,28.0,1,旧区,
GZ1701233236442611712,2000,84.0,2,旧区,北
417193,450,20.0,1,旧区,
...
```

sz.csv

共 46193 条数据（节选 20 条）：

```
ID,租金,面积,房型,板块,朝向
...
SZ1692522919604781056,4000,77.0,3,布吉关,南
SZ1684489713219534848,8500,107.4,3,布吉关,东南 南
SZ1663839653767151616,5700,95.0,3,布吉关,东南
380381,2250,30.0,1,布吉关,
SZ1664191275701633024,6000,75.0,3,布吉关,南
380495,1950,30.0,1,布吉关,
SZ1643560736930136064,3800,54.71,2,布吉关,南
380380,2350,30.0,1,布吉关,
SZ1668578151690665984,6700,68.0,3,布吉关,南
SZ1684841996230852608,1800,10.0,1,布吉关,南
...
```

lf.csv

共 36731 条数据（节选 10 条）：

```
ID,租金,面积,房型,板块,朝向
...
LF1702291944681504768,1800,114.0,3,汇福悦榕湾商圈,南 西南 北
LF1714508085600452608,1200,36.0,1,汇福悦榕湾商圈,北
LF1697861013996568576,1100,81.22,2,汇福悦榕湾商圈,南 北
LF1611301493091074048,2300,95.0,2,汇福悦榕湾商圈,东南
LF1703317927358365696,900,90.17,2,汇福悦榕湾商圈,南 北
LF1675833891199385600,1400,56.46,1,汇福悦榕湾商圈,北
LF1679090947259891712,2000,129.0,3,汇福悦榕湾商圈,南 北
LF1628660693072871424,2000,52.57,1,汇福悦榕湾商圈,北
LF1704038086695976960,2000,130.0,3,汇福悦榕湾商圈,南 北
LF1667852945078616064,1500,118.0,3,汇福悦榕湾商圈,南 西北
...
```

3.2 数据预处理

3.2.1 缺省值检测与处理

使用 pandas 库依次读取 csv 文件并使用`isna()`方法检测空值

```
bj_data = pd.read_csv('bj.csv')
...
print(bj_data.isna().sum())
...
```

观察输出结果，发现只有广州与深圳的朝向信息有缺省，如下所示

```
...
[广州数据缺省情况：]
dtype: int64
ID        0
租金        0
面积        0
房型        0
板块        0
朝向     3305
[深圳数据缺省情况：]
dtype: int64
```

```
ID          0
租金          0
面积          0
房型          0
板块          0
朝向      11213
```

朝向属于重要性低，缺失率低的属性，在此不做处理。

3.2.2 重复值检测与处理

使用duplicated()方法检测重复值，因为属性值中有 ID 字段，所以如果数据有重复代表着一定是同一房屋，而不可能是信息完全相同的两套不同房屋。

```
print('北京重复值：' + str(bj_data.duplicated().sum()))
...
```

得到的结果如下：

```
北京重复值：5386
上海重复值：6112
广州重复值：23635
深圳重复值：13113
廊坊重复值：14672
```

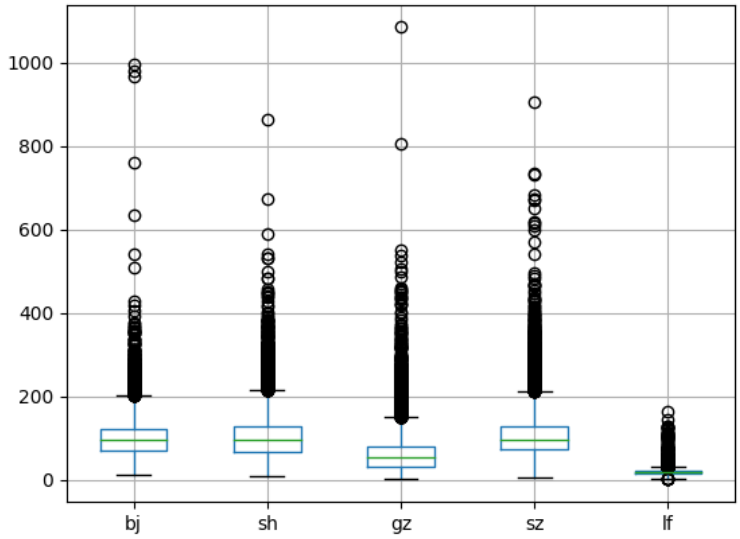
删除重复值：

```
bj_data.drop_duplicates(inplace=True)
sh_data.drop_duplicates(inplace=True)
gz_data.drop_duplicates(inplace=True)
sz_data.drop_duplicates(inplace=True)
lf_data.drop_duplicates(inplace=True)
```

3.2.3 异常值检测与处理

以每个城市的单位面积租金为指标，用箱型图法进行异常值处理：

```
rent = pd.DataFrame()
rent['bj'] = bj_data['租金'] / bj_data['面积']
...
rent.boxplot()
plt.show()
```



得到的结果如下图所示：

先在rent中找出异常条目，再按照rent中异常条目的索引从原DataFrame中删除异常值：

```
def drop_abnormal(data, name):
    q1 = rent[name].quantile(q=0.25)
    q3 = rent[name].quantile(q=0.75)
    low_limit = q1 - 1.5 * (q3 - q1)   # 计算下界
    high_limit = q3 + 1.5 * (q3 - q1)   # 计算上界
    # 从原DataFrame中删除异常条目
    data.drop(rent[(rent[name] > high_limit)|(rent[name] <
       low_limit)].index, inplace=True)

drop_abnormal(bj_data, 'bj')
...
```

最后将预处理完成的数据保存到新的 csv 文件中：

```
bj_data.to_csv('data/bj.csv', index=False)
...
```

3.3 数据展示与分析

3.3.1 总体租金情况

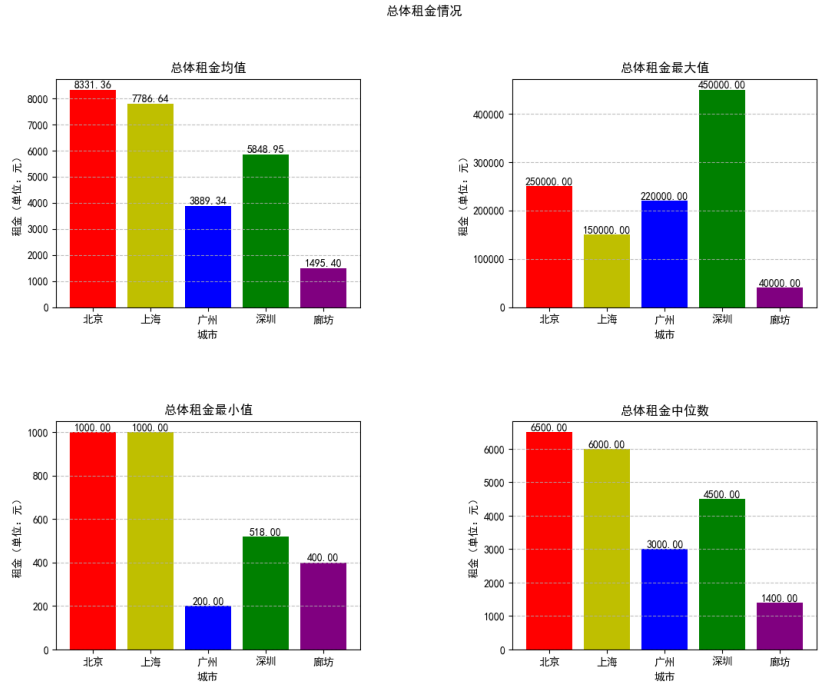
先使用mean()等函数计算出每个城市租金的均值、最大值、最小值和中位数：

```
bj_mean = bj_data.loc[:, '租金'].mean()
bj_max = bj_data.loc[:, '租金'].max()
bj_min = bj_data.loc[:, '租金'].min()
bj_mid = bj_data.loc[:, '租金'].median()
...
```

之后画一幅带有 2*2 共 4 幅柱状图的图来展示这些数据的情况：

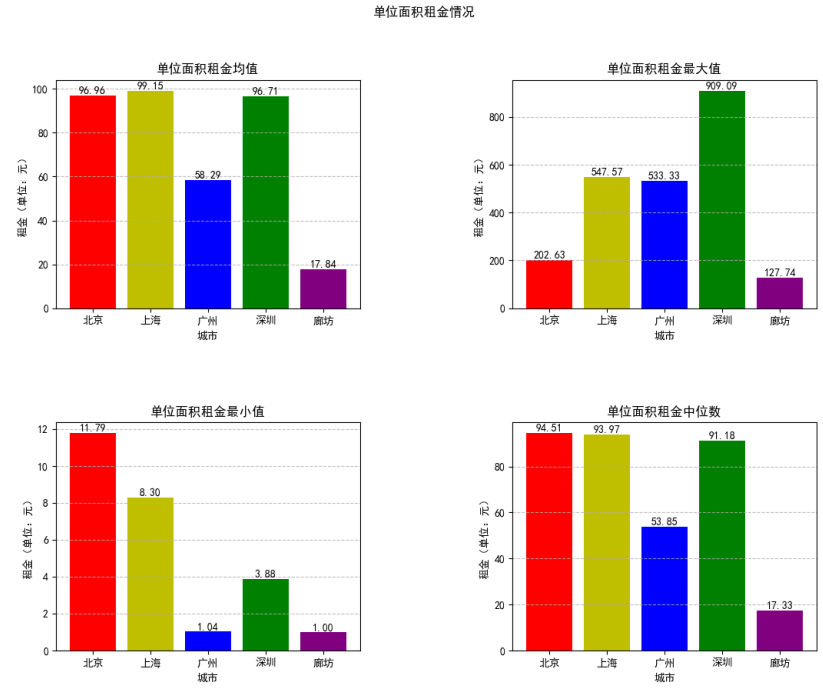
```
y1 = [bj_mean, sh_mean, gz_mean, sz_mean, lf_mean]
...

fig, axs = plt.subplots(2, 2)
axs[0, 0].bar(name, y1, color=colors)
axs[0, 0].set_title('总体租金均值')
for x, y in zip(name,y1):
    axs[0, 0].text(x, y, '%.2f' % y, ha='center', va='
       bottom')
...

for i in range(2):
    for j in range(2):
        axs[i, j].grid(axis='y', linestyle='--', alpha=0.8)
        axs[i, j].set_xlabel('城市', fontsize=10)
        axs[i, j].set_ylabel('租金（单位：元）', fontsize=10)

plt.subplots_adjust(hspace=0.5, wspace=0.5)
plt.suptitle('总体租金情况')
plt.show()
```

最后展示的图像如下所示：



3.3.2 单位面积租金情况

绘制方法与总体租金情况类似，在此不在赘述，最后展示的图像如下所示：

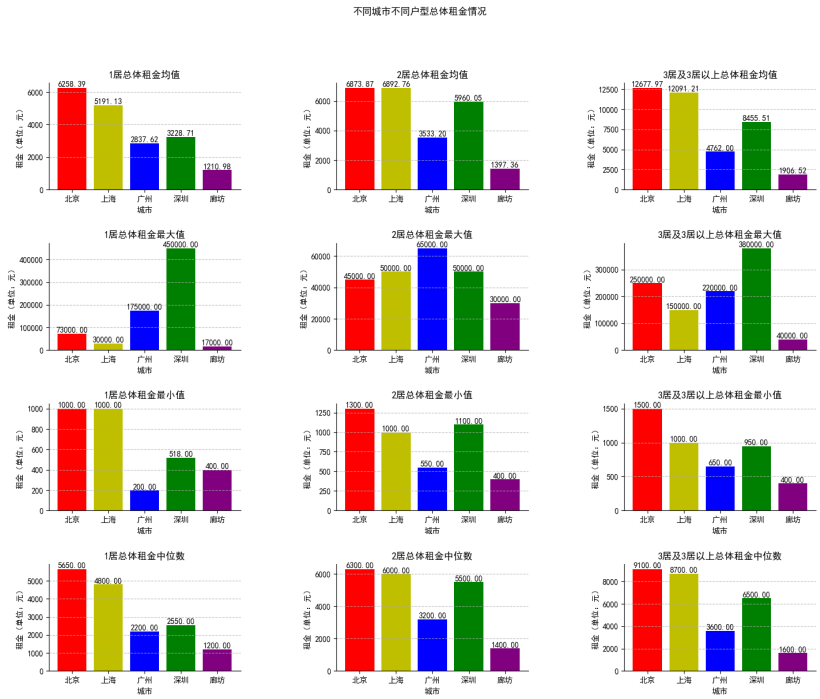


3.3.3 不同户型租金情况

通过复杂索引计算出不同城市，不同房型总体租金的均值、最大值、最小值和中位数，然后使用类似的方法进行图片绘制：

```
# 以均值为例
fig, axs = plt.subplots(4, 3)
for j in range(0, 4):
    for i in range(1, 4):
        if j == 0:
            y = [bj_data.loc[bj_data.房型 == i, '租金'].mean(),sh_data.loc[sh_data.房型 == i, '租金'].mean(),
            gz_data.loc[gz_data.房型 == i, '租金'].mean(),sz_data.loc[sz_data.房型 == i, '租金'].mean(),
            lf_data.loc[lf_data.房型 == i, '租金'].mean()]
            绘制图像
            ...
```

最后展示的图像如下所示：



3.3.4 不同板块租金情况

先使用百度地图 api 获取到每个城市不同板块的坐标，保存到 csv 文件中（同时顺便计算各板块均单位面积租金）：

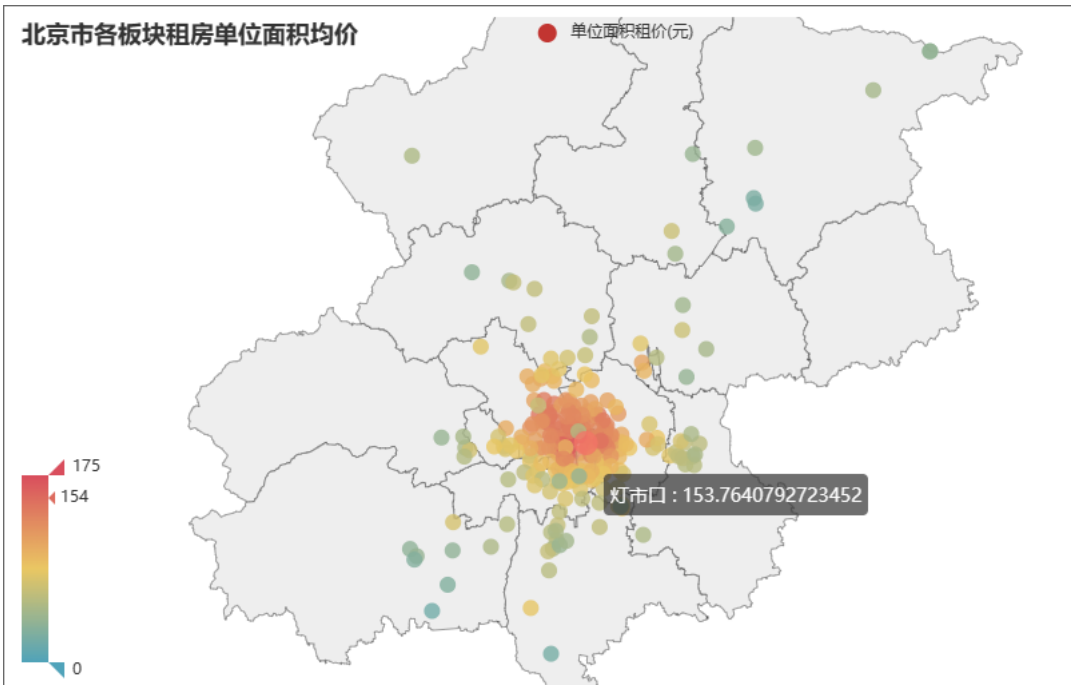
```
def geocoding(city, address):
    url = 'http://api.map.baidu.com/place/v2/search?'
    params = {
        "query": address,
        "region": city,
        "ak": 密钥,
        ...
    }
    response = requests.get(url, params=params)
    answer = response.json()
    places_ll = []
    if answer['status'] == 0:
        tmpList = answer['results'][0]
        coordString = tmpList['location']
        coordList = [coordString['lng'], coordString['lat']]
        places_ll.append([address, float(coordList[0]), float(coordList[1])])
        return float(coordList[0]), float(coordList[1])

bj_loc = pd.DataFrame(columns=['name', 'avg', 'x', 'y'])
bj_area = bj_data.板块.unique()
for i in range(len(bj_area)):
    x, y = geocoding('北京', bj_area[i])
    # 各板块计算平均单位面积租金
    bj_loc.loc[i] = [bj_area[i], bj_data.loc[bj_data.板块 == bj_area[i], 'rent'].mean(), x, y]
bj_loc.to_csv('data/bj_loc.csv', index=False)
...
```

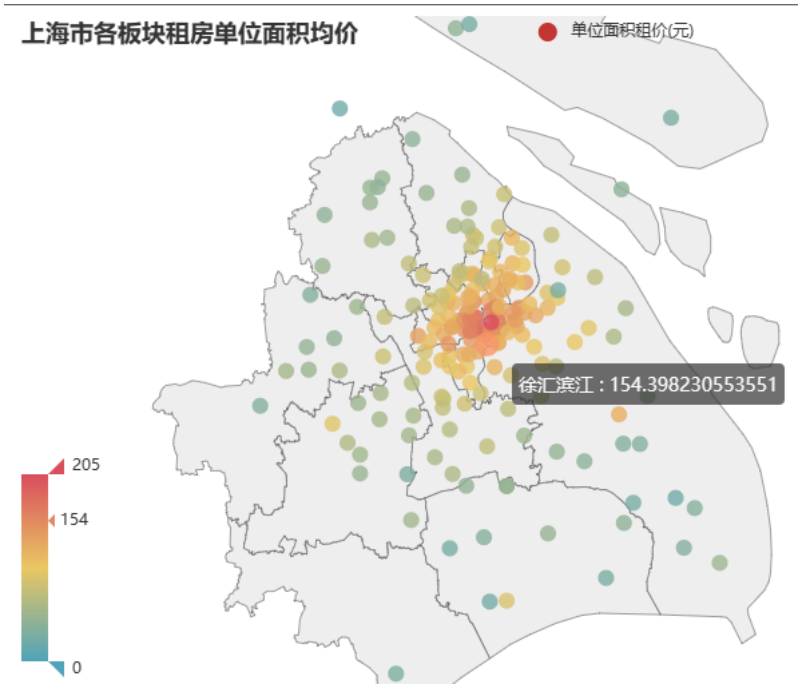
使用 pyecharts 读取相应 csv 文件中的数据，绘制散点热度图：

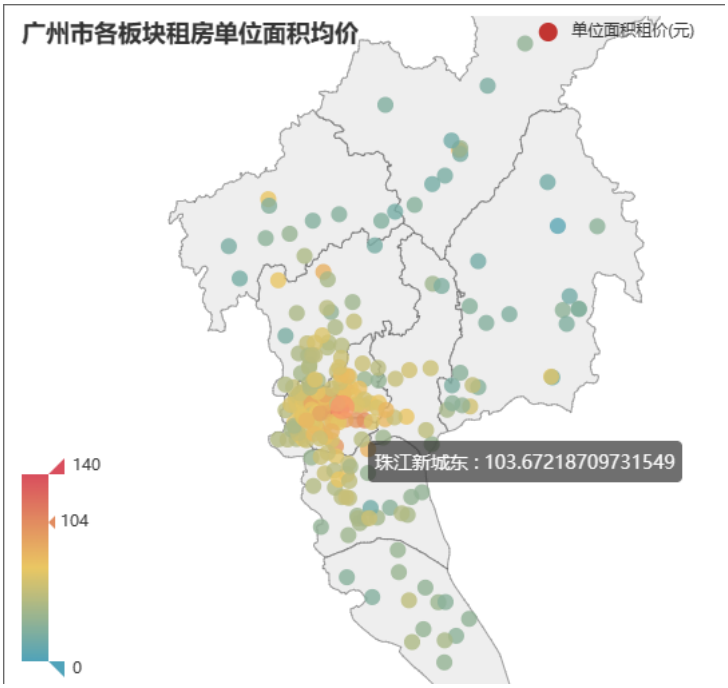
```
bj_loc = pd.read_csv('data/bj_loc.csv')

def block_map(title) -> Geo:
    geo = Geo()
    geo.add_schema(maptype="北京")
    for i in bj_loc.itertuples():  # 设置点坐标
        geo.add_coordinate(getattr(i, 'name'), getattr(i, 'x'), getattr(i, 'y'))
```

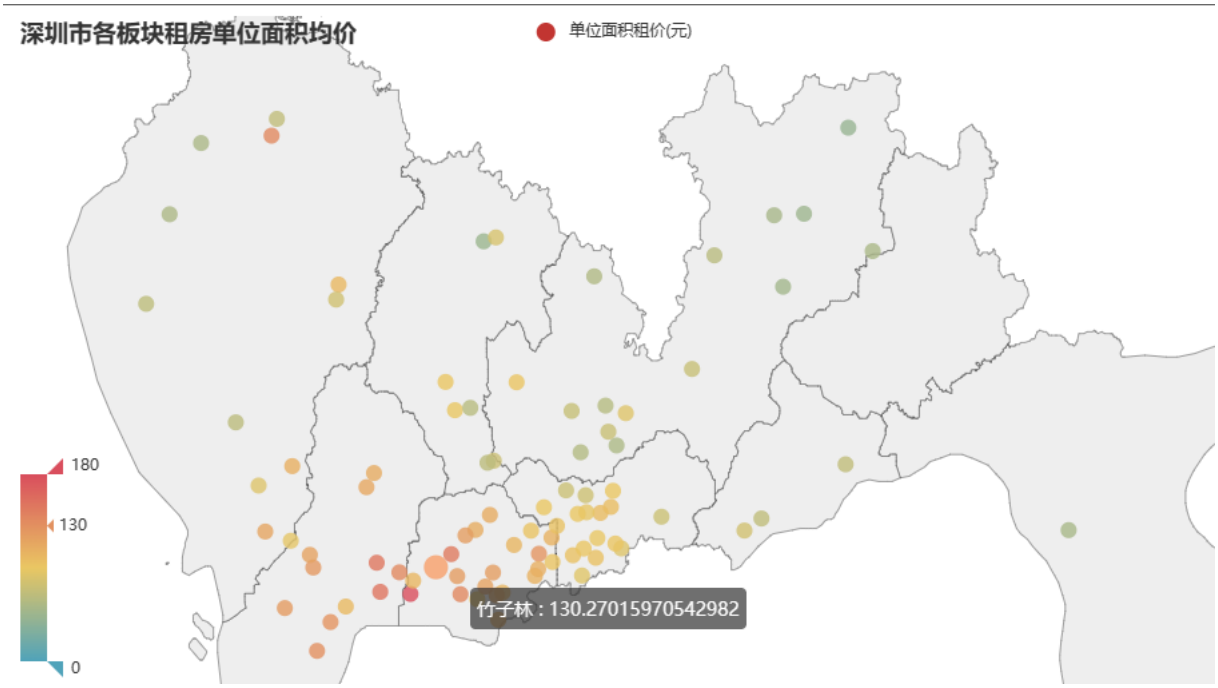
```
    geo.add(  #设置每个点的标签
        title, [tuple(z) for z in zip(bj_loc['name'],
      bj_loc['avg'])],
        label_opts=opts.LabelOpts(is_show=False))
    geo.set_series_opts(label_opts=opts.LabelOpts(is_show=
      False))
    geo.set_global_opts(visualmap_opts=opts.VisualMapOpts(
      max_=np.max(list(bj_loc['avg']))),
                        title_opts=opts.TitleOpts(title="北京市
      各板块租房单位面积均价"))
    return geo

block_map('单位面积租价(元)').render('assets/bj_block.html')
```

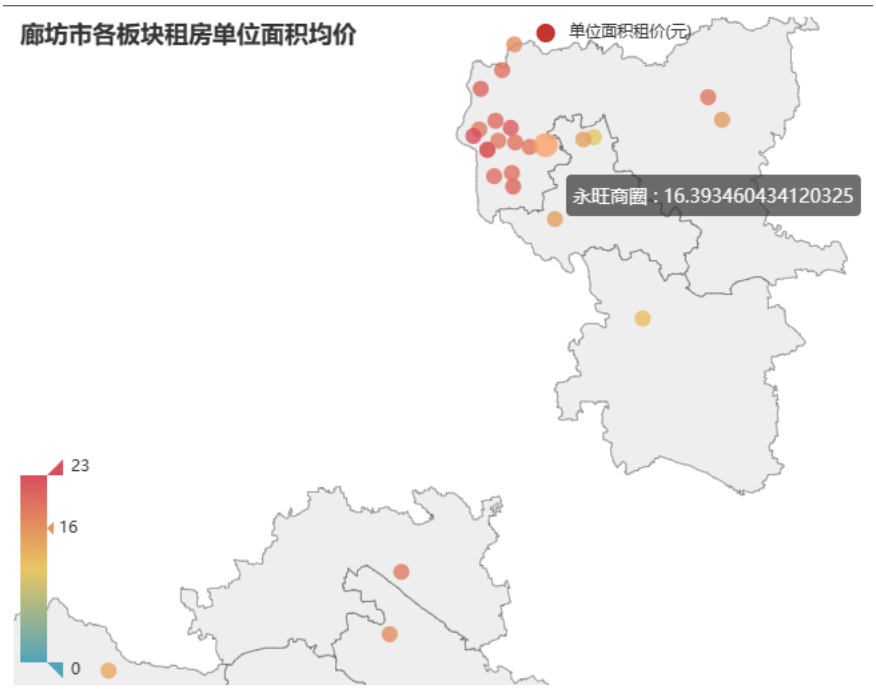
最后展示的图像如下所示：




上海市各板块租房单位面积均价
单位面积租价(元)
徐汇滨江：154.398230553551
205
154
0


广州市各板块租房单位面积均价
单位面积租价(元)
珠江新城东：103.67218709731549
140
104
0





3.3.5 不同朝向租金情况

先将朝向字段中的东南，东北，西南，西北替换为 a，b，c，d，方便稍后运用字符串判断不同朝向的条目：

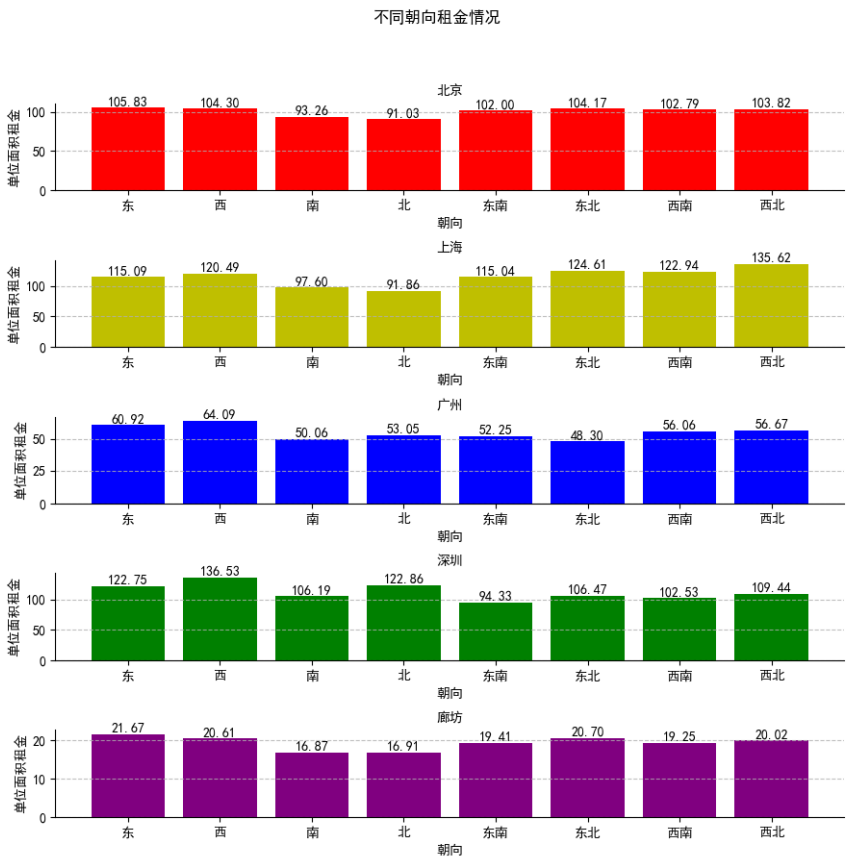
```
faces = ['东', '西', '南', '北', 'a', 'b', 'c', 'd']
f_str = ['东', '西', '南', '北', '东南', '东北', '西南', '西北'
        ]
for i in range(4, 8):
```

```
    bj_data['朝向'] = bj_data['朝向'].replace(f_str[i],
        faces[i])
    ...
```

然后分别计算出每个城市不同朝向的单位面积均价，并绘制柱状图。在判断朝向时如果朝向为空，则不归纳入任何朝向：

```
fig, ax = plt.subplots(5, 1)
for j in range(5):
    avg_price = []
    if j == 0:
        for i in range(8):
            avg_price.append(bj_data.loc[bj_data['朝向'].str
        .contains(faces[i], na=False), 'rent'].mean())
        ax[j].bar(f_str, avg_price, label='北京', color=
        colors[j])
```

最后展示的图像如下所示：



分析数据发现，房屋朝向对单位面积租金的影响不大。综合来看，房屋朝向南和北的单位面积租金较低；朝向西的单位面积租金较高。不同城市之间的相似点体现在朝南

方向的房子价格普遍较低，经查阅资料发现，这可能是因为：

1. 影响早上睡眠：朝南的房子早晨阳光直射；
2. 厨房卫生间容易受潮：一般朝南的房子厨房卫生间常年得不到光照，容易受潮；
3. 除此之外，还可能有通风不足，餐厅位置不协调等因素。

而不同城市之间的差异点体现在每个城市高价的朝向各异，北京和廊坊朝东租金最高，上海朝西北租金最高，深圳和广州朝西租金最高。体现出南方城市普遍朝西租金较高，北方城市朝东租金较高。

3.3.6 GDP 与租金情况分析

网上调查得到 2021 年人均 GDP 为：

```
# 单位：万元
# 数据来源：中经数据
bj_gdp = 18.4
sh_gdp = 17.36
gz_gdp = 15.04
sz_gdp = 17.37
lf_gdp= 6.45
```

计算出每个城市的人均 GDP/平均单位面积租金的比例：

```
mean = rent.mean().to_list()
...
y = [gdp[i] / mean[i] for i in range(5)]
```

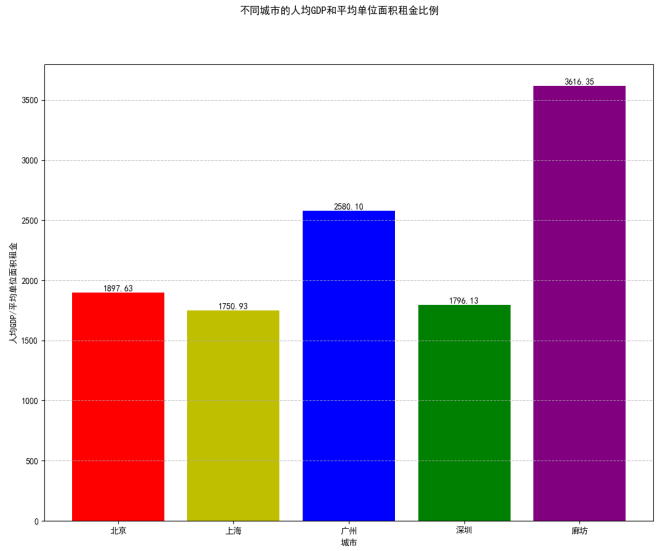
绘制柱状图如下：

读图可以发现，在廊坊市租房性价比最高。

3.3.7 人均可支配收入与租金情况分析

网上调查得到 2021 年人均可支配收入为：

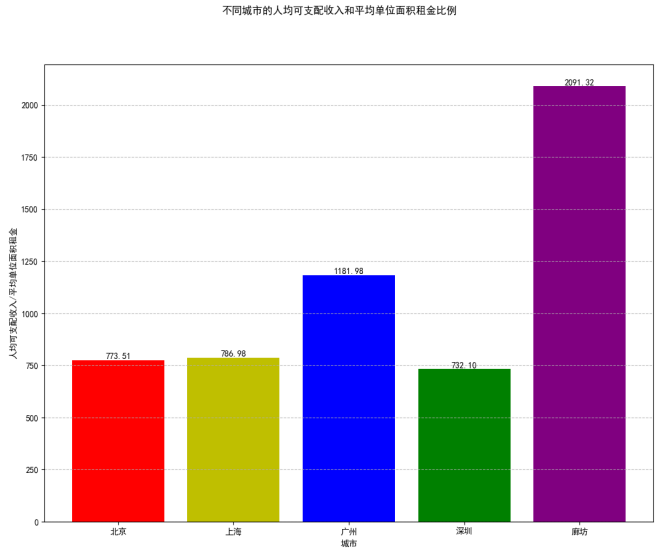
```
# 单位：元
# 数据来源：国家统计局各市调查大队
bj_income = 75002
sh_income = 78027
gz_income = 68900
sz_income = 70800
lf_income= 37300
```



计算出每个城市的人均可支配收入/平均单位面积租金的比例：

```
mean = rent.mean().to_list()
...
y = [income[i] / mean[i] for i in range(5)]
```

绘制柱状图如下：



读图可以发现，在深圳市租房负担最重。

3.4 自主探索

3.4.1 题目设计

补充爬取北京在租房屋的楼层信息，分析楼层高度对房屋单位面积租金的影响。由于每次爬取一个房屋的楼层信息都需要访问一次页面，工作量较大，所以只爬取北京的楼层信息。

3.4.2 数据获取

因为每个房屋的楼层信息都在房屋本身的主页上，所以需要进入每个房屋的首页爬取，改造爬虫程序，使其按照已经爬取到的房屋 ID 访问每套房屋的主页，获取到其楼层，并根据面积和租金算出单位面积租金。

```
# 在floor.py中
class MySpider(scrapy.spiders.Spider):
    name = 'floor'
    allowed_domains = ['bj.lianjia.com']
    # 用已经获取到的ID构造url
    start_urls =  ["https://bj.lianjia.com/zufang/" + ID +
       '.html' for ID in pd.read_csv('../data/bj.csv')['ID'
       ]]
    def parse(self, response):
        data = pd.read_csv('../data/bj.csv')
        item = FloorItem()
        item['ID'] = response.request.url.split('.')[-2].
      split('/')[-1]
        item['rent'] = float(data.loc[data.ID == item['ID'
      ], 'rent'])  # 直接借用之前计算好的单位租金
        item['floor'] = eval(response.xpath('//li[@class="
      floor"]/span[2]/text()')[0].extract()
            .split('/')[-1].split('层')[0])  # 获取楼层信息
        yield item

# 在pipelines.py中将信息保存至csv
class FloorPipeline:
    def open_spider(self, spider):
        self.info = pd.DataFrame(columns=['ID', 'floor', '
      rent'])
    def process_item(self, item, spider):
        dict_item = dict(item)
```

```
22          self.info.loc[len(self.info)] = [dict_item['ID'],
        dict_item['floor'], dict_item['rent']]
23          return item
24      def close_spider(self, spider):
25          self.info.to_csv('../data/bj_floor.csv', index=
        False)
```

3.4.3 数据分析

使用hean()和tail()方法查看数据的顶部和尾部几条：

```
1 data = pd.read_csv('./data/floor.csv')
2 print(data.head())
3 print(data.tail())
```

输出结果如下：

```
1  ID  floor        rent
2  0  BJ1712378196185317376       6  35.526316
3  1  BJ2898562730338557952       2  97.770825
4  2  BJ2897190668550553600       3  26.181209
5  3  BJ1613203985907318784      11  60.000000
6  4  BJ1608332984685953024       6  60.606061
7  ID  floor        rent
8  25474  BJ1682010233485918208      20  58.939096
9  25475  BJ1712666789751554048      15  57.777778
10 25476  BJ1717050456749899776       6  48.000000
11 25477  BJ1709192823682629632      24  46.428571
12 25478  BJ1716806114034581504       6  56.395488
```

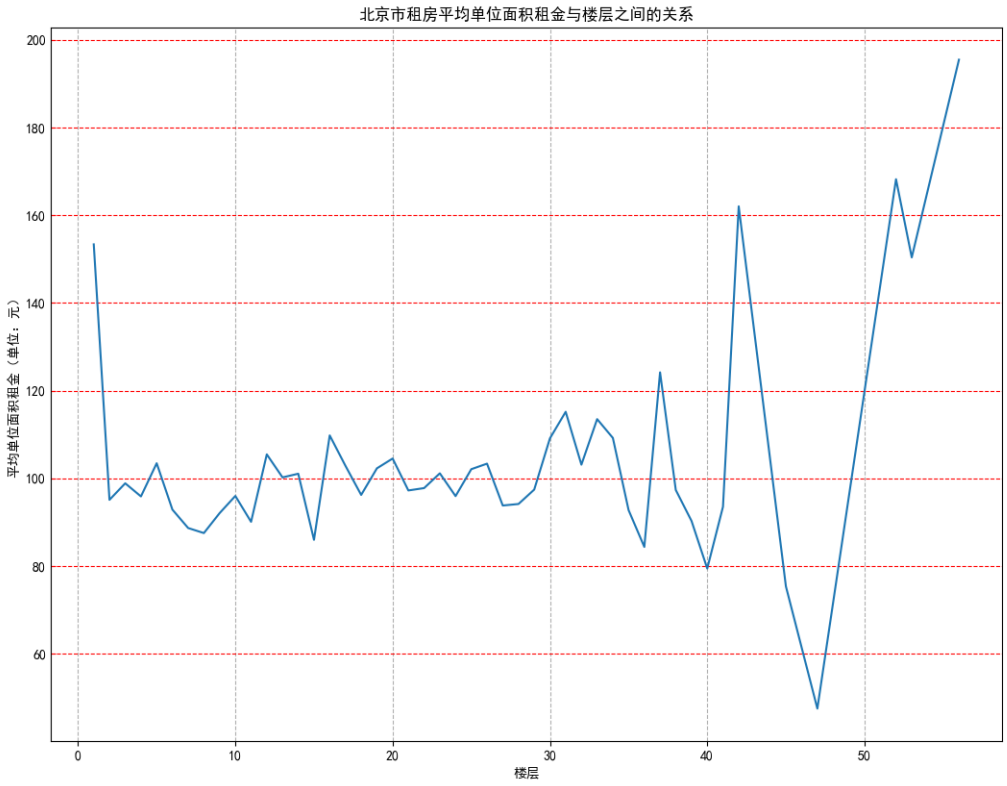
3.4.4 数据展示

使用折线图反映价格随楼层增高的变化：

```
1 floors = data.floor.unique()
2 floors.sort()
3 f_price = pd.DataFrame(columns=['x', 'y'])
4 for i in range(len(floors)):  # 计算出不同楼层的平均单位面积租金
5     f_price.loc[i] = [floors[i], data.loc[data.floor ==
        floors[i], 'rent'].mean()]
6 plt.title('北京市租房平均单位面积租金与楼层之间的关系')
```

```
plt.grid(linestyle='--')
plt.xlabel('楼层')
plt.ylabel('平均单位面积租金（单位：元）')
plt.tick_params(axis='y', direction='in', color='r',
        grid_color='r')
plt.plot(f_price.x, f_price.y)
plt.show()
```

最后展示的图像如下所示：



第四部分　实验总结

本次大作业花费了我 5 天的时间完成，过程中的收获满满。

在数据获取阶段，一开始我并没有使用 scrapy 框架，而是使用了 requests+lxml 的方式单线程爬取数据，尽管也可以得到结果，但是效率过于缓慢。每次在发现爬取到的数据有误时都需要重新爬取，浪费了很多时间。后来使用了 scrapy 进行了改进，同时调整了每次爬取信息的筛选条件后效率大大提升。

在设置筛选条件时，一开始我设置的是价格 + 房型栅格式筛查，后来发现这样做不但麻烦，而且难以获取到某些特殊情况下房屋的板块信息。后来采用板块进行筛查，不但条件变得更加简洁了，而且还可以补充一定的板块信息。

在数据预处理阶段，一开始由于没有爬取房屋 ID 的信息，导致去重的时候不知道是房屋就是有本身信息重复还是把同一个房屋爬取了多次，后来仔细观察每一个房屋对象，找到了他们的房屋 ID 之后就可以放心的去重了。

在数据展示与分析阶段，我学习了如何使用百度的地图 API 进行板块的经纬度查询，并使用 pyecharts 绘制了地图版本的板块与租金分布图。

总体而言，本次大作业给我的收获并不仅仅是一些 Python 库和接口的知识，更是提高了我解决问题的能力。我在克服一个个困难的同时磨练了自己的思维，每一次从不同的角度解决问题时都让我感到十足的成就感。虽然这次大作业花费了我预料之外的许多精力，但是我相信它带给我的收获会陪伴我走上更长久的进步之路。