Python 程序设计

链家网租房数据处理与分析

简旭锋 @ 2020212895

2022.12

目录

一、需求分析

1.1、项目需求

从链家官网抓取北京市、上海市、广州市、深圳市、厦门市的全部租房数据。

- 比较这五个城市的总体租房情况，包含租金的均价、最高价、最低价、中位数等信息，以及单位面积租金（元/平米）的均价、最高价、最低价、中位数等信息。
- 比较这五个城市一居、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。
- 比较各个城市不同朝向的单位面积租金分布情况。
- 查询各个城市的人均 GDP 和平均工资，分析其和单位面积租金分布的关系。

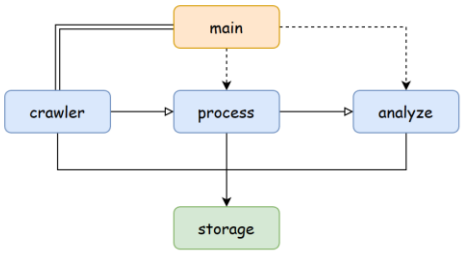
1.2、需求分析

- 获取不同城市的尽量全的租房信息
- 获取并区分各租房信息的城市、板块、户型、价格
- 获取各城市的人均 GDP、平均月工资
- 数据比较及展示分析

二、结构设计

2.1、结构概览

根据项目需求，将项目结构划分为主模块、数据抓取、数据处理、数据展示四个模块。各模块间的主要顺序及关系如下图：



2.2、模块设计

2.2.1、数据抓取

数据抓取模块负责通过链家官网抓取五个城市所有房源的非重复的原始信息。主要依托 scrapy 爬虫模块、Beautiful Soup 4 HTML 解析模块以及 Sqlite3 数据库模块。处理后的原始数据保存在数据库 db 文件中。

2.2.2、数据处理

数据处理模块针对从链家官网获取的原始数据进行预处理。去除无用信息、无效信息以及异常值。主要依托 Pandas 数据处理模块进行。处理后的分析数据通过 csv 文件保存

2.2.3、数据展示

数据展示模块针对经过预处理的数据进行不同类别数据的分析和展示。主要依托 Pyecharts 图像绘制模块。数据展示图表文件通过.jpg 及.html 文件保存。

2.2.4、主模块

主模块通过调用各个数据分析处理模块完成整个项目的运行。也可以单独配置运行需要调用的模块进行特定部分的数据处理，提高项目运行的灵活性和针对性，减少重复操作带来的时间浪费。

三、数据获取

3.1、网站分析

3.1.1、页面结构

对于一二线城市，总体租房数据量一般在万的级别。如厦门市(二线城市)租房数据量显示为 21834 套。但实际初始导航页面 xm.lianjia.com/zufang/中最多只展示 100 页，且每页的数据量级大约为 28-33 个数据项。

因此，需要考虑通过细化分级模式以抓取全部数据。前文提到，在导航模块中有区域的选项，首先考虑通过分区域进行数据抓取。但在查看各个区域的数据量后可以发现，大部分区域的数据量依然超过 5000 条，网页上展示的 100 页无法容纳所有数据。

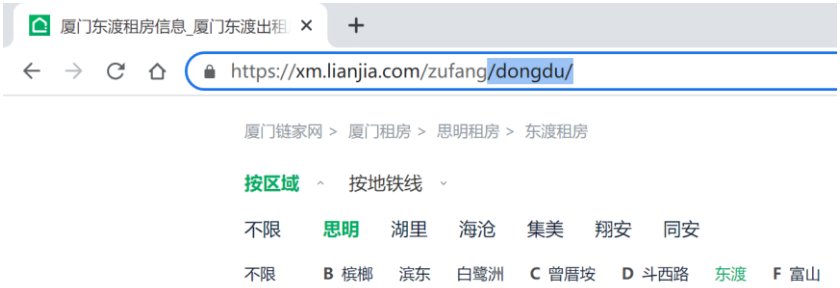
再次细分区域，通过各个 CBD 进行数据抓取。查看各个 CBD 数据后发现，几乎所有 CBD 的房源数据均不超过 3000，即能够在网站的 100 页限制中呈现该 CBD 的所有数据。因此，该项目主要通过不同 CBD 的划分策略进行数据抓取。

3.1.2、划分策略优化

本项目采用以 CBD 为区分的划分策略进行数据抓取。

但通过观察个各城市各区域的 CBD 导航可以发现，在选定不同的区域时，其中的 CBD 导航很可能重复。例如，在厦门市的租房页面上，思明区和湖里区的导航选项中都含有"东渡"板块。

不过，在实际点击 CBD 链接后，都会跳转到同一地址。这说明在服务器端，租房页面 CBD 的划分并不区分区域，而是直接指向 CBD 地址。

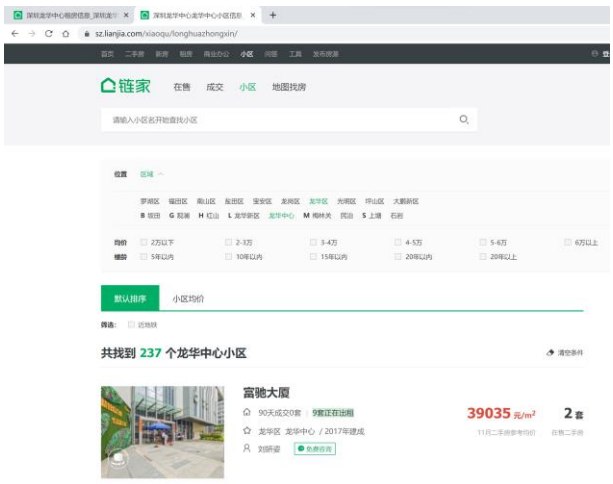


因此，项目考虑提前通过区域地址将所有 CBD 信息进行抓取保存，再通过获取的 CBD 数据进行实际数据项的抓取。

3.1.3、大型 CBD 处理

在部分城市的 CBD 中，实际上也发现了超过 3000 条数据限制的情况，例如深圳市龙华区的龙华中心板块，其租房数据量为 6046。

针对此类大型 CBD，需要通过其他途径获取数据。在链家官网中还有"小区"的大类，可以考虑通过小区进行细化。



各个 CBD 板块都对应有小区的连接，即从原来的/zufang/更改为/xiaoqu/即可访问对应的小区页面。在小区的各个数据项中，可以看到'%d 套在租'的选项，其中页面为各个小区的租房数据项，数据项的大致内容与排布和主页面没有区别。

因此，大型 CBD 板块的内容可以通过结合板块页面和小区页面进行数据抓取。

3.1.3、数据项分析

针对各个数据项，主要信息有名称、板块分布、面积、朝向、户型以及月租金

但由于链家允许其他租房中介品牌的房源信息在链家官网上放置广告，而针对广告，其数据的展示顺序及内容不尽相同。

另一方面，在下方的详细信息展示条中，并没有通过 `div` 的 `id` 或 `class` 进行区分，因此该项的提取需要通过字符串处理。

因此，在数据抓取阶段，仅将完整的原始数据进行抓取，将字段的细分处理调整到数据处理阶段，简化数据抓取模块，提高后续数据处理的灵活性。

3.2、Scrapy 模块

3.2.1、item

前文分析，本次数据抓取模块分为两部分：CBD 信息的抓取以及具体数据项的抓取。因此，为 Scrapy 项目的 item 设计了 CBDItem 以及 RentingItem。

在 CBDItem 中，存储有该 CBD 对应的城市以及区域，同时存储了该 CBD 的总租房数和总页数，便于第二阶段的"翻页"实现和数据比对。

```
class CBDItem(scrapy.Item):
    city = scrapy.Field()       # 城市
    region = scrapy.Field()     # 地区
    name = scrapy.Field()       # 名称
    url = scrapy.Field()        # 链接
    total = scrapy.Field()      # 总租房数
    pages = scrapy.Field()      # 总页数
```

在 RentingItem 中，还存储了房源项的 id(即官网数据源的 house_code 项)，作为唯一性数据判断依据。同时为了区分广告房源，还存储了房源品牌和广告代码。

```
class RentingItem(scrapy.Item):
    id = scrapy.Field()         # 标号
    name = scrapy.Field()       # 名称
    city = scrapy.Field()       # 城市
    cbd = scrapy.Field()        # 板块
    ad_code = scrapy.Field()    # 广告标识
    brand = scrapy.Field()      # 房源品牌
    price = scrapy.Field()      # 总价
    addition = scrapy.Field()   # 附加信息(面积、户型、朝向)
```

3.2.2、pipeline 及去重

- **CBD**

对于抓取的 CBD 信息的处理，由于 CBD 数据量级较小，因此在程序内部实现。通过 python 的 next()函数判断 CBD 名称的重复项。不同城市的 CBD 分开存储，因此不需要考虑不同城市间的名称重复问题。

```
def process_item(self,item,spider):
  node = dict(item)   # 字典化
  if next((x for x in self.data[node['city']] if x['name'] == node['name']),None) == None:
      self.data[node['city']].append(node) # 去重插入
  return item
```

对于抓取到的 CBD 信息，在爬虫结束时统一写入 json 文件，分城市进行存储。

```
def close_spider(self,spider):
    city = spider.city  # 城市标签
    with open(f'data/citys/{city}/cbd.json','w',encoding='utf-8') as f:
        json.dump(self.data[city],f,indent=4,ensure_ascii=False) # json 存储
```

- **房源信息**

对于抓取的具体房源项处理，由于房源信息数据量级很大，在程序内部时间会产生较大的时间消耗和内存消耗。因此，此处调用了 python 内置的 sqlite3 数据库模块进行数据的存储。通过设置房源 id(house_code)为主键以达到自动去重的效果。

数据表的声明如下，id 被设置为主键。

```
sql = f'''create table {city} (
        id text, name text, cbd text,
        ad_code text, brand text,
        price text, addition text,
        primary key(id));'''
db.exec(sql)
```

Pipeline 中对接收到的 item 信息建立 SQL 语句，进行数据库 insert 操作。

```
def process_item(self,item,spider):
        item = dict(item)
        sql = f'''insert into {spider.city} values('{item['id']}',
                '{item['name']}','{item['cbd']}','{item['ad_code']}',
                '{item['brand']}','{item['price']}','{item['addition']}');
        '''
        self.db.exec(sql)
```

3.2.3、Spider

- **CBD Spider**

对于 CBD 的 Spider 处理，依托 BS4 模块的 find()函数和 select()函数解析 HTML 内容。

```python
# 地区处理
def parse_region(self, response):
    doc = BS(response.text,features="lxml") # BS4 解析
    params = {}
    for node in doc.find('ul',attrs={'data-target':'area'}).select('.filter__item--level2'):
        if node.attrs.get('data-id') == '0':
            continue
        params['region'] = node.find('a').text
        href = node.find('a').attrs.get('href')
        yield Request(url=f'https://{self.city}.lianjia.com{href}',callback=self.parse_cbd_pre,cb_kwargs=params)
```

针对每个 CBD，提前进入其主页面，提取房源总数和总页数进行保存。

```python
# CBD详细信息处理
def parse_cbd(self, response, item):
    doc = BS(response.text,features="lxml")
    item['total'] = int(doc.find('span',class_='content__title--hl').text) # total数据解析
    if item['total'] > 0:
        item['pages'] = int(doc.find('div',class_='content__pg').attrs.get('data-totalpage'))    # pagebox解析，提取总页数
        yield item
```

- **Renting Spider**

读取详细房源数据项依托 CBD 信息。通过 Json 导入已读取的 CBD 信息，根据网址和总页数遍历抓取每页数据。

```python
def start_requests(self):
    # 读取CBD数据
    with open(f'data/citys/{self.city}/cbd.json','r',encoding='utf-8') as f:
        self.cbd_data = json.load(f)
        f.close()
    for item in self.cbd_data:
        url = item['url']
        pages = item['pages']
```

对于房源数量超过 3000 的大型 CBD，额外采取依照小区信息重读的方式进行数据获取。由于数据通过数据库存储，无需考虑数据重复问题。

```python
if item["total"] > 3000:    # 针对大于3000的CBD，进行小区分类的重读取
    unit_url = str(url).replace('zufang','xiaoqu')
    params = {'cbd':deepcopy(item['name']), 'url':unit_url}
    yield Request(url=unit_url, callback=self.unit_requests,cb_kwargs=params)
```

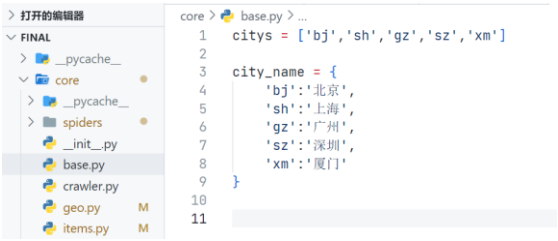
针对每个数据项，通过 BS4 解析其详细信息，通过字符串函数 replace()进行空格、换行符的删除预处理。对没有标注房源品牌的数据，默认指定为链家。

```python
# 处理基本数据项
def parse(self,response,cbd):
    doc = BS(response.text,features="lxml")
    for ele in doc.find('div',class_='content__list').select('.content__list--item'):
        id = ele.attrs.get('data-house_code')
        ad_code = ele.attrs.get('data-ad_code')

        name = ele.find('p',class_='content__list--item--title').text.strip()
        brand = ele.find('span', class_='brand')
        brand = brand.text.strip() if brand != None else '链家'
        price = ele.find('span', class_='content__list--item-price').text.replace('\n','').strip()

        des = ele.find('p',class_='content__list--item--des').text.replace('\n','').replace('  ','')

        item = RentingItem()
        item['id'] = id
        item['name'] = name
        item['city'] = self.city
        item['cbd'] = cbd
        item['ad_code'] = ad_code
        item['brand'] = brand
        item['price'] = price
        item['addition'] = des

        yield item
```

3.3、核心封装

3.3.1、配置

在 core 爬虫核心模块中，通过 base.py 配置需要爬取的城市代码及名称，提供了高度简化的自由度配置。



3.3.2、城市并行化

不同城市间的数据不会相互影响，因此可以通过 CrawlerRunner 和 Reactor 模块并行启动不同城市的 CBD 抓取或租房信息数据抓取。

```
from scrapy.crawler import CrawlerRunner
from twisted.internet import reactor
# CBD 抓取
def cbd_crawler():
    runner = CrawlerRunner()
    for city in citys:
        runner.crawl(CBDSpider, city=city)  # 城市并行化
    runner.join().addBoth(lambda _:reactor.stop())
    reactor.run()
```

通过该项设置，在程序启动后，进行 CBD 信息获取时可以同时启动 5 个城市的爬虫线程，提高程序的运行效率。

3.3.3、CBD、租房信息串行化

根据程序的模块设计，在进行房源信息的获取前需要先获取 CBD 数据。因此，CBD 信息的获取和房源信息的获取是串行的。这就要求实现：

1、CBD、房源串行运行

2、CBD 和房源信息获取内部，各个城市并行。

在程序编写中发现，在一个进程中，reactor 只允许启动一次。因此通过 multiprocessing 多进程模块，在爬虫程序中启动两个进程以解除这个限制。

```
import multiprocessing
# main crwaler api
def catch(recap_cbd = False):
    targets = [cbd_crawler] if recap_cbd else []
    targets.append(renting_crawler)
    for t in targets:
        p = multiprocessing.Process(target=t)
        p.start()
        p.join()
```

上述函数为数据获取模块的最外层封装函数，通过参数 recap_cbd 指定是否需要重新获取 CBD 数据。通过调用该函数即可完成数据获取阶段的所有工作，降低了主模块和爬虫程序的耦合程度，也便于使用在主模块中的自由配置。

四、数据处理

4.1、数据预处理

4.1.1、信息提取与切分

在原始信息中，由于价格、朝向、户型等数据提取的网页原始数据字段，因此需要对这部分信息进行拆分。

	Id	Name	Cbd	Ad_code	Brand	Price	Addition
0	445603	独栋·地派公寓 CCB建融家园·龙翔公寓【SM店】温馨大单间 开间	SM	0	地派公寓	1696-1700 元/月	仅剩2间/31.00㎡/2间在租/1室0厅1卫
1	464924	独栋·地派公寓 CCB建融家园·龙翔公寓【SM店】居家三室两厅 3室2厅	SM	0	地派公寓	6800 元/月	仅剩1间/143.00㎡/1间在租/3室2厅2卫
2	445612	独栋·地派公寓 CCB建融家园·龙翔公寓【SM店】理想中两室一厅 2室1厅	SM	0	地派公寓	4000 元/月	仅剩1间/76.00㎡/1间在租/2室1厅1卫
3	364695	独栋·小鱼公寓 SM-物业小区【现房+可视频看房】 1室1厅	SM	0	小鱼公寓	2599 元/月	仅剩1间/40.00㎡/1间在租/1室1厅1卫
4	364678	独栋·小鱼公寓 SM-物业小区 年前特惠-别墅区旁 1室1厅	SM	0	小鱼公寓	2199 元/月	仅剩1间/43.00㎡/1间在租/1室1厅1卫
5	473401	独栋·海西公寓 乌石浦裕兴花园 小清新 1室1厅	SM	0	海西公寓	1900-3000 元/月	仅剩4间/38.60-39.60㎡/4间在租/1室1厅1卫
6	XM1707950766	整租·裕兴花园 1室1厅 南	SM	0	海西公寓	2300 元/月	湖里-SM-裕兴花园/42.00㎡/南/1室1厅1卫/高楼层（19层）
7	XM1672474484	整租·裕兴花园 1室1厅 东	SM	0	海西公寓	2299 元/月	湖里-SM-裕兴花园/39.84㎡/东/1室1厅1卫/中楼层（18层）
8	XM1697611856	整租·乌石浦花园二期 2室1厅 东南	SM	0	链家	4300 元/月	湖里-SM-乌石浦花园二期/62.41㎡/东南/2室1厅1卫/中楼层（17层）

- **价格**

对于金额，其数据类型为单个整数或两个整数组成范围，加上后缀”元/月”，该部分字段通过正则表达式提取。

```
price = re.match('([0-9\-\.]+) 元/月',input)    # 正则提取
```

由于价格信息可能存在范围，因此通过 python 字符串类型的 split()函数进行切分，通过 numpy 模块的 average()函数计算平均值并取整，以替代范围类型的数据。

```
price = list(map(int,price.group(1).split('-')))    # 对于范围，计算平均值
avg = np.average(np.array(price))
```

- **租房类型**

租房类型对于数据分析可能存在影响，因此将其作为单独的数据列一并提取存储。租房类型信息存储在名称”name”字段中作为前缀，该部分同样可以通过正则表达式提取。对于没有该部分的字段，标注为”未知”。

```
def process_type(input):    # 租房类型判断
    type = re.match('((整租)|(合租)|(独栋)).',input)    # 正则提取
    if type != None:
        if (str(type.group(1))) not in ['整租','合租','独栋']:
            print(type.group(1))
        return str(type.group(1))
    else:
        return "未知"
```

- **面积**

面积数据存储在原始信息的”addition”字段中，用正则表达式提取为。

```
area = re.search('([0-9\-\.]+)㎡',input)    # 面积处理
```

由于面积中同样存在范围，因此对范围类型数据计算均值，保留小数类型。

```
area = list(map(float,str(area.group(1)).split('-')))
avg = np.average(np.array(area))        # 对于范围， 取其均值
```

- **户型**

户型信息存储在原始信息的"addition"字段中，通过正则表达式提取。

```
room = re.search('([0-9]+)室([0-9]+)厅([0-9]+)卫',input)    # 户型处理
```

- **朝向**

朝向信息存储在原始信息的"addition"字段中，由"/"包裹，多个朝向由空格隔开。通过正则表达式可以提取：

```
orien = re.search('/[东西南北]+(\s{1}[东西南北]+)*/',input)     # 朝向处理
```

对于多个朝向的数据，输出时通过"/"进行连接，以便于后续数据分析时筛选。

```
detail['orien'] = '/' + '/'.join(orien.group()[1:-1].split()) + '/'
```

4.1.2、广告信息判断

在链家官网的房源数据中，存在许多广告类型数据，各类广告数据的 HTML 标签并不一致，没有统一的区分字段。因此采取如下策略：

1、id(house_code)不以城市代码(字母)开头的数是广告数据

2、ad_code 不为 0 的数据为广告数据

根据上述两条规则，将广告信息的判断以布尔值(0/1)单独输出为"ad"字段存储。

```
def is_ad(id,ad_code):      # 广告判断
    if re.match('[A-Z]+[0-9]+',id) is None:
        return 1
    return 0 if str(ad_code) == '0' else 1
```

4.1.3、删除无用信息

- **地下室类型房源处理**

在处理信息时发现，在链家官网中有许多地下室类型的租房数据：

该部分类型的数据不便于筛除，但可以发现所有的地下室类型的数据都为 0 卫生间。在调查中发现，虽然也存在部分标注为 0 卫生间的租房数据，但依照大部分租户的使用需求，卫生间这类隐私空间应该在大部分租户的考虑范围之内。因此这里将所有 0 卫生间的房源信息进行筛除，不计入本次统计范围。

```
df_new.drop(df_new[df_new['room'] <= 0].index,inplace=True)
```

- **删除空信息**

在数据表中可以找到许多没有"面积"或"价格"的表项(值为 0 或 NA)，这类数据对于后续数据分析没有帮助，可以视作无效房源，将其删除。

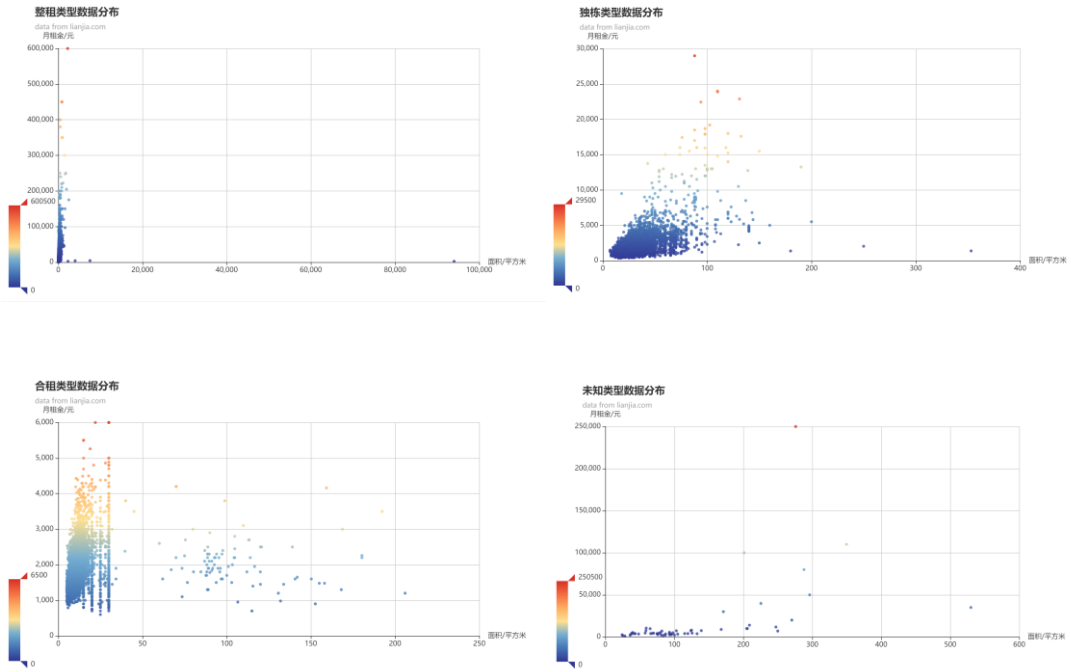
此外，在 Python 中，NA/NaN 类型是唯一一种不等于其自身的数据，因此可以通过这个特性进行 NA/NaN 类型数据的判断。

```
# drop invalid
    df_new.drop(df_new[(df_new['price'] <= 0) | (df_new['price'] != df_new['price'])].index,inplace=True)
    df_new.drop(df_new[(df_new['area'] <= 0) | (df_new['area'] != df_new['area'])].index, inplace=True)
```

4.2、异常数据处理

4.2.1、原始数据图像

该部分提前引用了数据展示模块的内容，以形成数据异常处理策略。数据分类为租房类型(整租、独栋、合租、未知类型)。



4.2.2、离群点分析及处理

在上述图像中，存在许多离群点数据，针对这些数据进行分析处理。

- **删除单位面积价格过低的条目**

在整租类型数据中，存在房源面积为 94036 平方米，而租金仅为 2300 元/月。查看链家官网，发现房型为普通房屋类型，该数据为异常数据，应该删除。

此类房源信息面积异常的情况常反映为单位面积价格过低。因此，考虑将单位面积价格小于 3 元的数据列入异常数据，进行删除。

```
df = df.drop(df[df['unit_price'] < 3].index)    # 删除单位面积价格小于 3 的数据
```

- **删除面积与价格不匹配的条目**

通过数据表格排序，可以看到许多面积大而租金很低的条目。经过 id 搜索，这类条目均属于链家数据源的数据有误。

	id	name	city	cbd	brand	type	area	orien	room	price	unit_price	ad
118479	GZ1715619923604013056	整租·富力金港城南区 3室2厅 北	广州	花东镇	链家	整租	94036	/北/	3	2300	0.02445872	0
64008	XM1713445851772223488	整租·龙山山庄 2室2厅 东南	厦门	莲前	链家	整租	7527	/东南/	2	4300	0.57127674	0
107319	GZ1711645287728021504	整租·珠江新岸公寓 1室0厅 东	广州	滨江东	链家	整租	4000	/东/	1	4000	1	0
92085	GZ1708781286149062656	整租·顺欣广场 3室2厅 东南	广州	新塘北	链家	整租	2300	/东南/	3	2400	1.04347826	0
165242	510770	独栋·微業青年公寓 谭罗华侨32店 阳台单间	深圳	龙华中心	微業青年公	独栋	353		1	1377	3.90084986	1
108194	GZ1698676766089936896	整租·元邦山清水秀 4室2厅 南/北	广州	狮岭镇	链家	整租	312	/南/北/	4	1500	4.80769231	0
145274	SZ1686045969772183552	整租·莲花山庄五村 1室1厅 西南	深圳	布吉街	链家	整租	280	/西南/	1	1400	5	0
87147	GZ1699646045635477504	整租·合盛秀丽庄园 4室2厅 复式 东南	广州	小楼镇	链家	整租	197	/东南/	4	1000	5.07614213	0
108006	GZ1582303774272126976	整租·致岭假日花园 4室2厅 南	广州	狮岭镇	链家	整租	193	/南/	4	1000	5.18134715	0
55428	XM1651817679557754880	合租·大洲城市花园 7居室 南卧	厦门	富山	链家	合租	205.95	/南/	7	1200	5.82665696	0

因此，考虑将租房面积超过 200 平方米，而租金小于 1500 元/月的数据列入异常数据，进行删除。

```
df = df.drop(df[(df['area'] >= 200) & (df['price'] <= 1500)].index)
```

- **合租类型数据处理**

结合合租类型的数据分布图和总体数据表格，可以看到有许多合租类型的条目标注的是房屋总面积而非单租户实际使用面积。

	id	name	city	cbd	brand	type	area	orien	roon	price	unit_price	ad
55428	XM1651817679557754880	合租·大洲城市花园 7居室 南卧	厦门	富山	链家	合租	205.95	/南/	7	1200	5.82665696	0
60184	XM1659852432139616256	合租·大永固花园 6居室 东南卧	厦门	海沧生活区	链家	合租	152.64	/东南/	6	900	5.89622642	0
59809	XM1607733808432414720	合租·天源 6居室 东南/南卧	厦门	海沧体育馆	链家	合租	115	/东南/南/	6	700	6.08695652	0
59961	XM1567519747933732864	合租·天湖城天湖 3居室 东南卧	厦门	海沧区政府	链家	合租	132	/东南/	3	980	7.42424242	0
148649	SZ1635169831458177024	合租·合正园 7居室 南卧	深圳	梅林	链家	合租	168.09	/南/	7	1300	7.73395205	0
167176	SZ1625353352097300480	合租·东方明珠城 5居室 南卧	深圳	龙岗中心城	链家	合租	106.61	/南/	5	950	8.91098396	0
145353	SZ2868879382511099904	合租·阳光花园二期 5居室 东卧	深圳	布吉街	链家	合租	130.69	/东/	5	1200	9.18203382	0
141442	SZ1621830996536066048	合租·春华四季园二期 4居室 南卧	深圳	坂田	链家	合租	158.09	/南/	4	1480	9.36175596	0
166199	SZ2558896596016439296	合租·君悦龙庭三期 4居室 南卧	深圳	龙岗中心城	链家	合租	155	/南/	4	1480	9.5483871	0
161067	SZ1620015939791618048	合租·富通蟠龙居 6居室 西南卧	深圳	西乡	链家	合租	150.26	/西南/	6	1600	10.6482098	0
59980	XM1639560040895479808	合租·天成广场 3居室 南卧	厦门	海沧区政府	链家	合租	133.83	/南/	3	1450	10.834641	0
67398	XM1580872922543685632	合租·融侨观邸 4居室 南卧	厦门	马銮湾东	链家	合租	140.63	/南/	4	1600	11.3773732	0
150023	SZ2785615006140735488	合租·滢水山庄 4居室 北卧	深圳	民治	链家	合租	141.89	/北/	4	1650	11.6287265	0
154262	SZ1698364467949076480	合租·碧海富通城三期 5居室 南卧	深圳	碧海	链家	合租	120.06	/南/	5	1450	12.0772947	0
135509	SZ1684052065707360256	合租·左庭右院北区 3居室 南卧	深圳	丹竹头	链家	合租	115.57	/南/	3	1400	12.1138704	0
143993	SZ1576792203672420352	合租·深航假日名居 5居室 东北卧	深圳	宝安中心	链家	合租	180.33	/东北/	5	2200	12.1998558	0
143649	SZ1576793612635602944	合租·深航假日名居 5居室 西南卧	深圳	宝安中心	链家	合租	180.33	/西南/	5	2260	12.5325792	0
60218	XM1686001530420854784	合租·荣发楼 2居室 西南卧	厦门	海沧生活区	链家	合租	103	/西南/	2	1500	14.5631068	0

由于数据处理时无法判断租户所付的真实租金和所租的真实面积，因此针对合租类型数据采取如下策略：

1、保留合租类型数据条目，参与月租金统计分布；

2、在含有单位面积月租金的统计项目中，去除合租类型数据。

4.3、核心代码

4.3.1、大型表项的 apply()使用

由于总体数据表的条目数量很大，通过 for 循环或迭代器的方式进行循环处理效率较低，因此采用 Pandas DataFrame 的 apply()方法进行大型数据表的多列迭代处理。

针对前文所述的数据拆分处理，核心代码大致为：

```
df['type'],df['price'],df['area'],df['orien'],df['room'],df['unit_price'],df['ad']
= zip(*df.apply(lambda x:ext_function(x),axis=1))
 # 数据表连接
df_new = pd.concat([df_new,df],join='inner', ignore_index=True)
```

在 apply()函数中，自定义数据切分的函数 ext_function(row)，并通过 zip()函数将多个返回类型迭代器进行打包，返回给原数据表。

```
# 数据切分 apply()函数
def ext_function(row):
    type = process_type(row['name'])           # 租房类型
    add = process_addition(row['addition'])     # 附加数据
    price = process_price(row['price'])         # 价格
    unit_price = price/add['area'] if add['area'] > 0 else 0    # 单位面积价格
    return type,price,add['area'],add['orien'],add['room'],unit_price,is_ad(row['id'],row['ad_code'])
```

五、数据展示

5.1、绘图模块

5.1.1、绘图模块简介

本项目主要采用 pyecharts 绘图库进行数据展示图像的绘制。pyecharts 是一个用于生成 Echarts 图表的类库。Echarts 是一个由百度开源的数据可视化，拥有良好的交互性和精巧的图表设计。pyecharts 是 Echarts 与 Python 的对接。使用 pyecharts 可以生成独立的网页。

5.1.2、绘图模块对比

在绘制功能上，使用广泛的 matplotlib 和 pyecharts 都能绘制高级的数据展示图像，但 pyecharts 可以生成 html 展示文件，并且有更好的交互性，使用者可以动态选取不同类型元素进行展示。

因此，本项目选择以 pyecharts 进行数据展示，其良好的交互特性能够更好的为数据分析部分提供帮助。

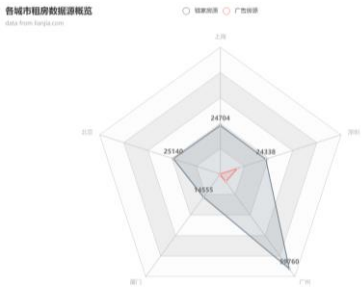
5.2、图像绘制

5.2.1、总体数据情况

总体数据情况展示各个城市的所有租房数据，采用 pyecharts.charts.Radar 雷达图进行展示。租房数据分为链家房源和广告房源，叠加在同一张图中。

```
df_lj = df[df['ad'] == 0].groupby(['city']).count()['id'].reset_index(name='count')      # 链家房源
df_ad = df.groupby(['city'])['ad'].apply(lambda x:(x==1).sum()).reset_index(name='count')# 广告房源
```

图像快照如下：



5.2.2、租房类型情况

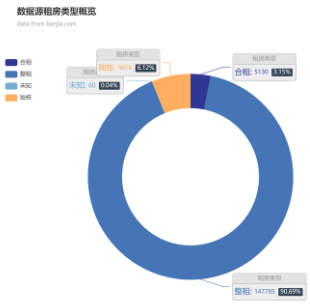
- **类型分类统计**

租房类型分类统计展示整租、独栋、合租类型的数据在所有数据中的分布情况，采用 pyecharts.charts.Pie 饼图进行绘制，可以更好的展示数据占比。

通过对数据中"type"字段进行分类聚合(groupby()函数)以及count()函数进行数据分类和计算。

```
res = df.groupby(['type'])['id'].count().reset_index(name='count')
for index,row in res.iterrows():
    data.append([row['type'],row['count']])
```

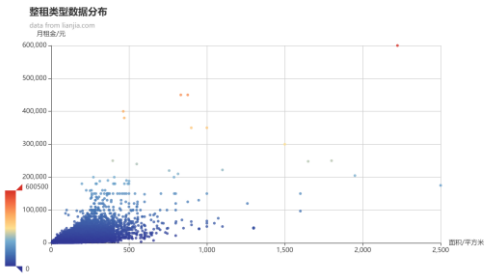
图像快照如下：



- **不同类型价格情况**

不同租房类型价格情况展示各类型租房数据的月租金价格分布，采用 pyecharts.charts.Scatter 散点图进行绘制。数据通过"type"字段进行筛选。

图像快照如下：



5.2.3、广告情况

广告情况统计各个城市出现的所有广告品牌及其投放的广告房源的数量，仅作辅助展示，采用 pyecharts.charts.WordCloud 词云图进行展示。

数据处理部分同样采用 groupby()函数进行分类以及 count()函数进行统计计算。

```
res = df.groupby(['brand'])['id'].count().reset_index(name='count')
```

图像快照如下：

5.2.4、价格分布对比

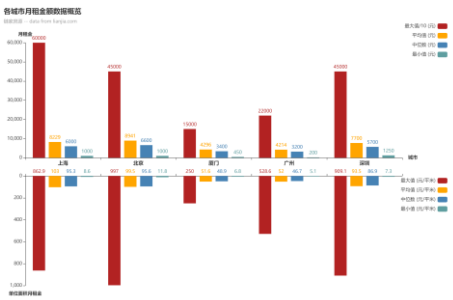
价格分布比较 5 个城市的租金和单位面积租金的均价、最高价、最低价以及中位数等信息。采用 pyecharts.echarts.Bar 条形图进行绘制，并通过 Grid 模块进行组合。

数据分析部分采用 groupby()分城市统计，通过 agg()函数进行平均值、中位数等的聚合运算。

```
res = df.groupby('city')[col].agg([np.average,np.max,np.min,np.median]).reset_index()
```

值得注意的是，由于月最高租金部分差异过大，容易使得图像比例失调，因此最大数据部分的单位为/10 元。

图像快照如下：



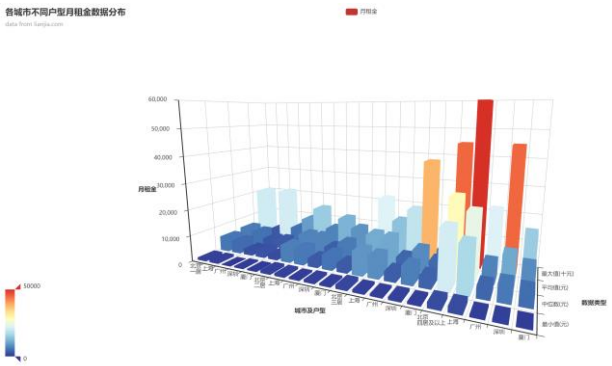
5.2.5、户型分布对比

户型分布对比五个城市一居、二居、三居及四居以上的价格情况，包括均价、最大值、最小值、中位数等。采用 pyecharts.charts.Bar3D 立体条形图进行绘制。

数据分析部分同样采用 agg()函数进行聚合运算。

```
res1 = df[df['room'] == 1]['price'].agg(func).reset_index()
res2 = df[df['room'] == 2]['price'].agg(func).reset_index()
res3 = df[df['room'] == 3]['price'].agg(func).reset_index()
res4 = df[df['room'] >= 4]['price'].agg(func).reset_index()
```

图像快照如下：



5.2.6、板块分布对比

板块均价情况对比不同城市各个板块的均价。在本项目中发现，各个城市的板块数量庞大，各类图像的绘制都不够直观。另一方面，直接对比各个板块的情况对数据分析的意义较小，因此考虑将板块信息投射到城市地图中进行分析对比。

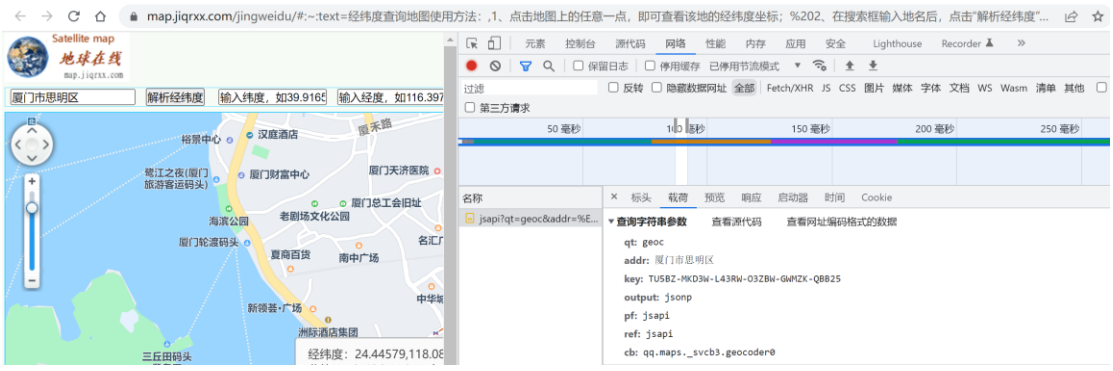
城市地图采用 pyecharts.charts.Geo 地理图像绘制。然而，pyecharts 中并没有各个板块的经纬度信息。因此，项目在此处通过 python 的 Requests 模块额外构造了简答的小型爬虫模块，用于获取不同城市各个 CBD 商圈板块的经纬度信息。

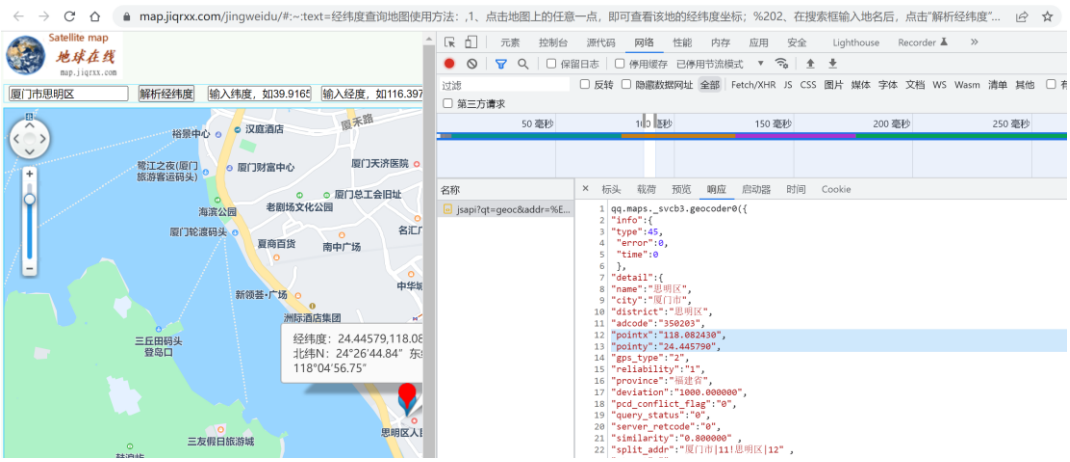
- **经纬度信息获取**

在数据获取阶段，已经分别存储了每个城市的详细 CBD 数据，因此，该小型爬虫程序根据已有的 CBD Json 文件，遍历获取其经纬度信息，并额外存储在 Position.json 文件中。

在经纬度信息查询网站 https://map.jiqrxx.com/jingweidu/#中，通过浏览器调试模式获取到数据接口和数据格式：

接口：https://apis.map.qq.com/jsapi

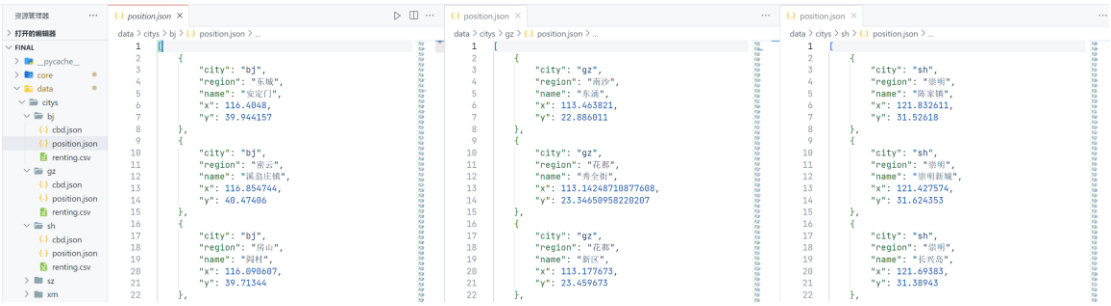




根据上述信息，构造爬虫程序的请求提交和响应解析部分，将获取到的经纬度信息通过 Json 进行保存。同时返回未查询到的 CBD 板块信息，该部分需要人工进行搜索查询。

```python
import requests
import json
import time
from core.base import citys, city_name
headers = {"User-Agent":'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36'}

def get_point(node):
    session = requests.session()
    city = city_name[node['city']]
    region = node['region']
    name = node['name']
    query = city + region + name
    url='https://apis.map.qq.com/jsapi'
    params = {
        'qt': 'geoc',
        'addr':query,
        'key':'UGMBZ-CINWR-DDRW5-W52AK-D3ENK-ZEBRC',
        'output':'jsonp',
        'pf': 'jsapi',
        'ref':'jsapi',
        'cb':'qq.maps._svcb3.geocoder0'
    }
    response = session.get(url=url, headers=headers, params=params)
    pos = response.text.find('"detail"')
    data = response.text[pos+9:-3]
    data = json.loads(data)
    try:
        return {'x':float(data['pointx']),'y':float(data['pointy'])}
    except Exception:
        print(f'not found: {query}')
        return {'x':-1,'y':-1}
```

最终数据保存结果为：

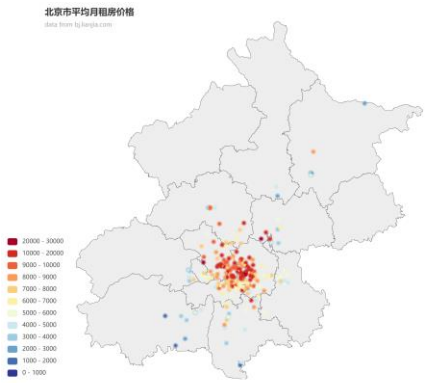


- **城市地图绘制**

获取到 CBD 经纬度信息后，通过 `Geo.add_coordinate()`加入自定义的点(即获取的 CBD 位置数据)。

数据分析部分，通过 `groupby()`函数和 `agg()`函数对每个 CBD 进行分类聚集计算，再根据 `Geo.add('',data_pair,typeGeoType.EFFECT_SCATTER)`将各个数据点以散点的形式加入到地图中。

图像快照如下：



5.2.7、朝向分布对比

朝向分布对比各个城市不同朝向的单位面积租金分布情况。为更直观展示不同朝向的价格分布，采用 pyecharts.echarts.HeatMap 热力图进行绘制。

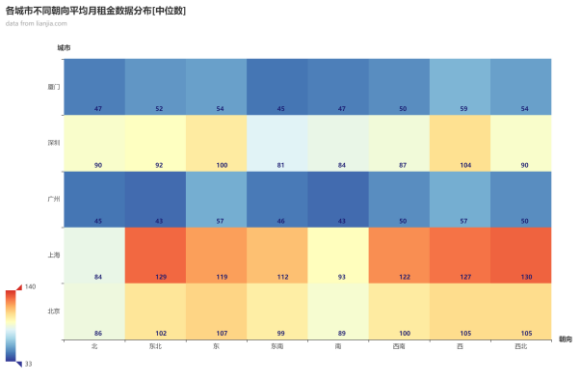
数据分析部分，根据前文叙述，只统计租房类型为整租和独栋的类型。由于房租最大值相比大部分数据有较高的跨度，对平均数可能产生影响，因此采用中位数进行数据展示。

```
df = df[(df['type'] == '整租') | (df['type'] == '独栋')]
res = df['unit_price'].median(axis=0)
```

对于一条数据含有多个朝向的情况，采取如下策略：如果一条数据含有某个朝向，则在统计对应朝向时可以统计该条数据。即一条数据可以被统计多次。该策略通过 python 中的 str.contains()方法实现。

```
df = df[df['orien'].str.contains(f'/{pos}/')].copy()
```

图像快照如下：



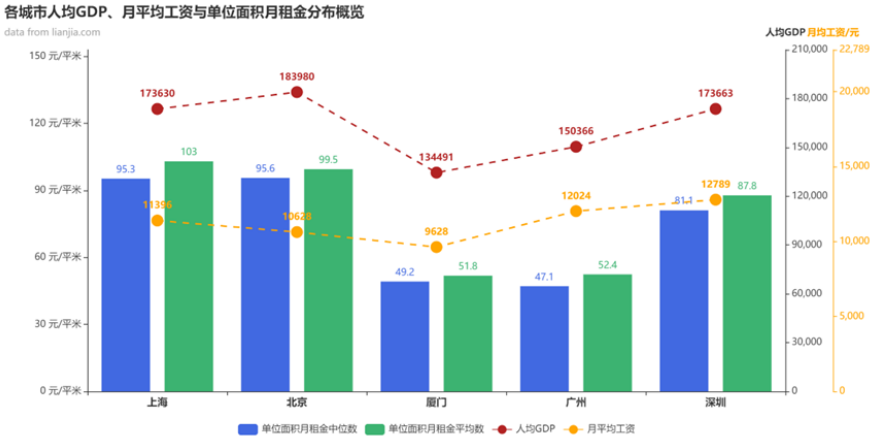
5.2.8、人均 GDP/工资及租金对比

各城市的人均 GDP 和平均工资通过互联网查询获得。对比二者和单位面积租金的分布情况，采用 pyecharts.echarts.Bar 条形图和 pyecharts.echarts.Line 折线图叠加进行展示。

单位面积租金分布数据根据上文价格分布中的数据解析部分取得，提取其中的平均数和中位数进行展示，通过 Bar.overlap()叠加 GDP 和工资的折线图。

```
b.overlap(l)    # 叠加 GDP
b.overlap(l2)   # 叠加平均工资
```

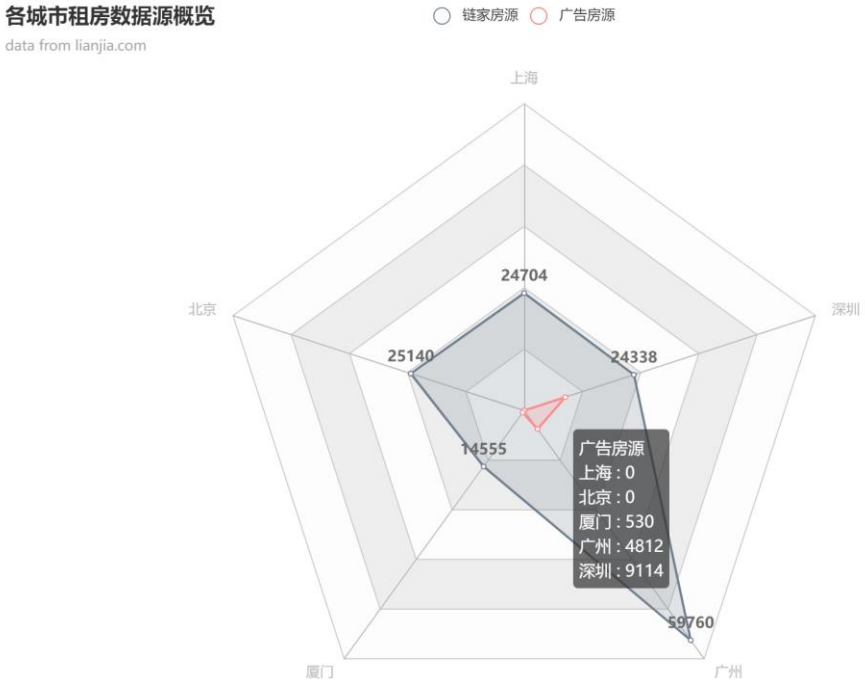
图像快照如下：



六、数据分析

6.1、总体数据分析

6.1.1、数据统计分析



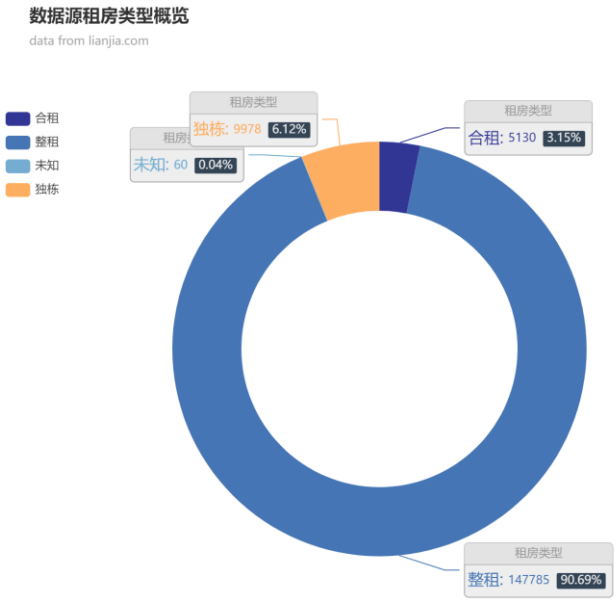
根据数据概览，本次项目中参与的数据分析房源数据总量为 162593 条，其中广州房源数据占比最大，达到 59760 条。对于广告房源，上海和北京没有广告数据，而深圳的广告数据条目最多，达到 9114 条，占深圳总房源数量的 27.25%。

由于本次项目的目的是分析各城市的租房数据，因此广告数据的参与并不影响该城市的总体租房情况，所以在后续的数据分析中，若无特别说明，广告数据都包含在内。

6.1.2、广告数据源品牌分析

根据上述图表，除北京、上海无广告数据，广州、深圳以及厦门的广告数据来源都非常丰富。需要额外说明的是，在数据中标注品牌为"贝壳优选"的数据并不算完全意义上的广告数据。因为"贝壳优选"是托管给链家的，称为"轻托管"模式，因此在数据统计时将"贝壳优选"品牌列入非广告数据。

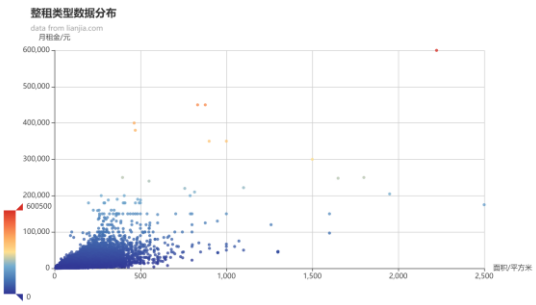
6.1.3、租房类型占比分析

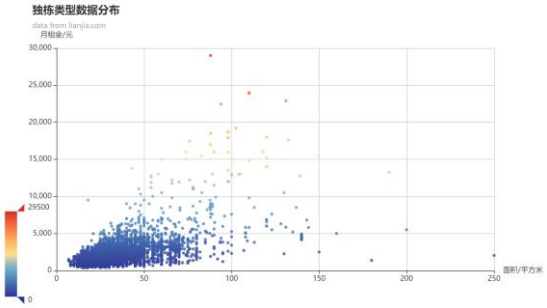


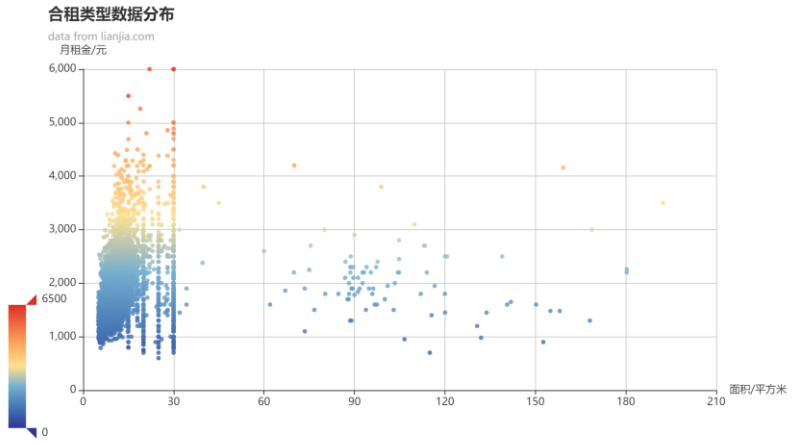
从上表中可以看出，绝大部分房源信息都是整租类型，小部分为独栋及合租类型。由于合租类型仅占 3.15%，按照前文所述的合租类型数据处理策略，在涉及单位面积租金的分析中忽略合租类型和未知类型并不影响整体的数据分布走向。

6.1.4、不同租房类型价格分布

在前文中展示了异常处理前的不同类型价格分布图标，这里针对异常处理后的主要数据进行展示分析。





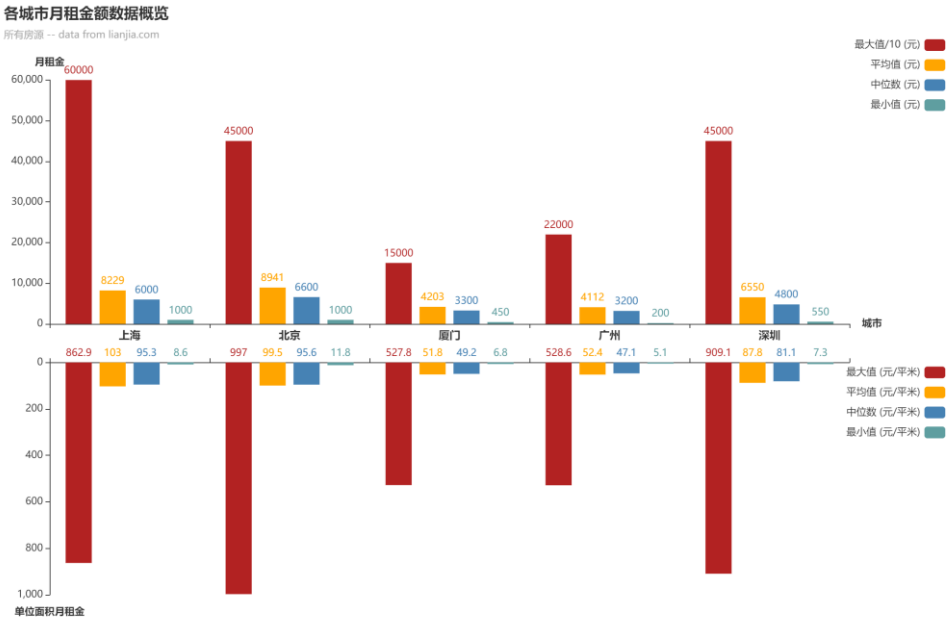


根据上述图表可以看到，整租类型和独栋类型的数据分布较为相似。在面积 400 平方米以下、月租金 5000 元以下的部分较为集中。

对于合租类型，可以看到大部分标注的数据为租房者实际使用面积，部分分散数据标注的是房屋总面积。

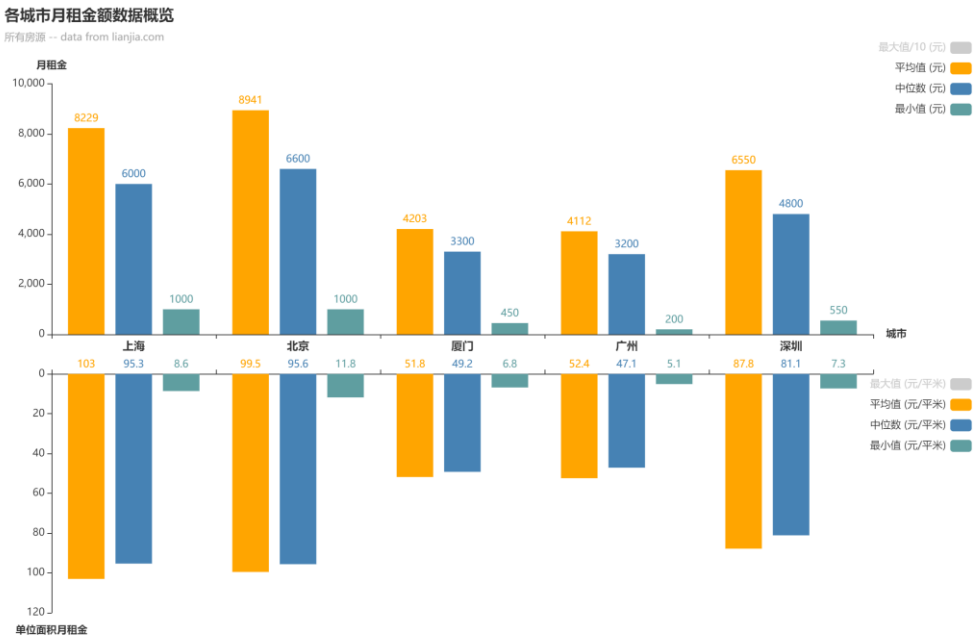
6.2、不同城市数据对比分析

6.2.1、总体租房情况分析

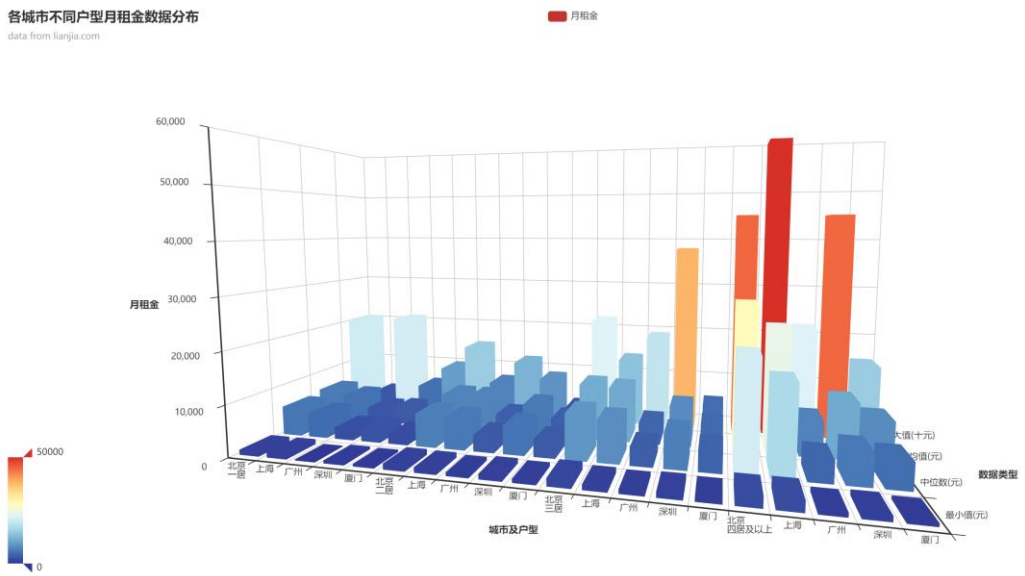


在上述表格中可以看到，上海市的最高月租金最高，达到了 600000 元/月，北京市的最高单位面积月租金最高，为 997 元/平米。

- **忽略差异较大的最高月租金**，可以看到，对于月租金和单位面积月租金，无论是平均值、中位数还是最小值，北京市的数据都是最高的，其次是上海市和深圳市，厦门市和广州市的数据差异不大。

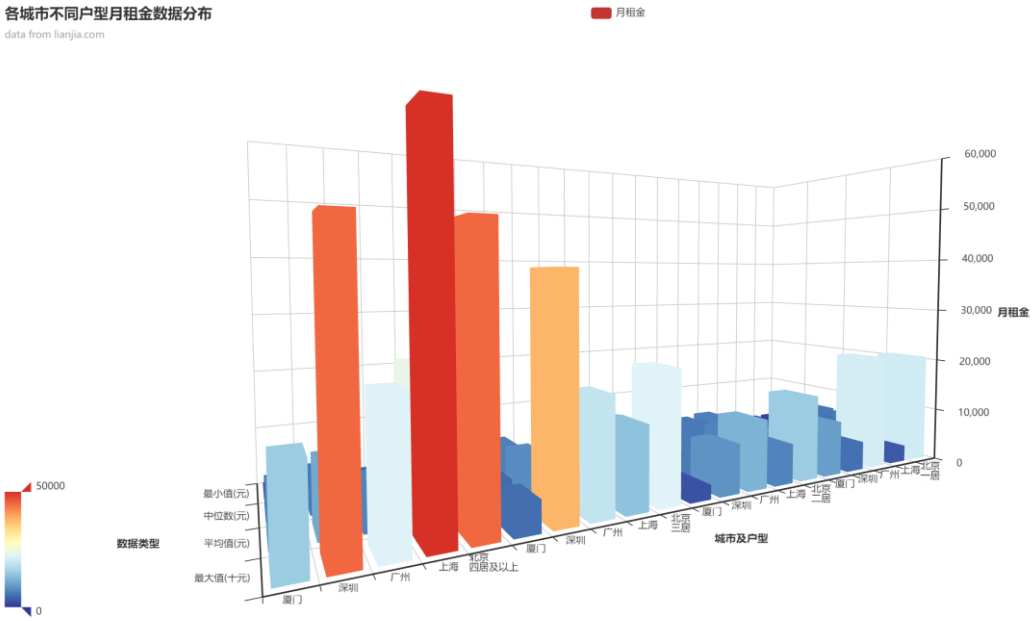


6.2.2、不同户型对比分析

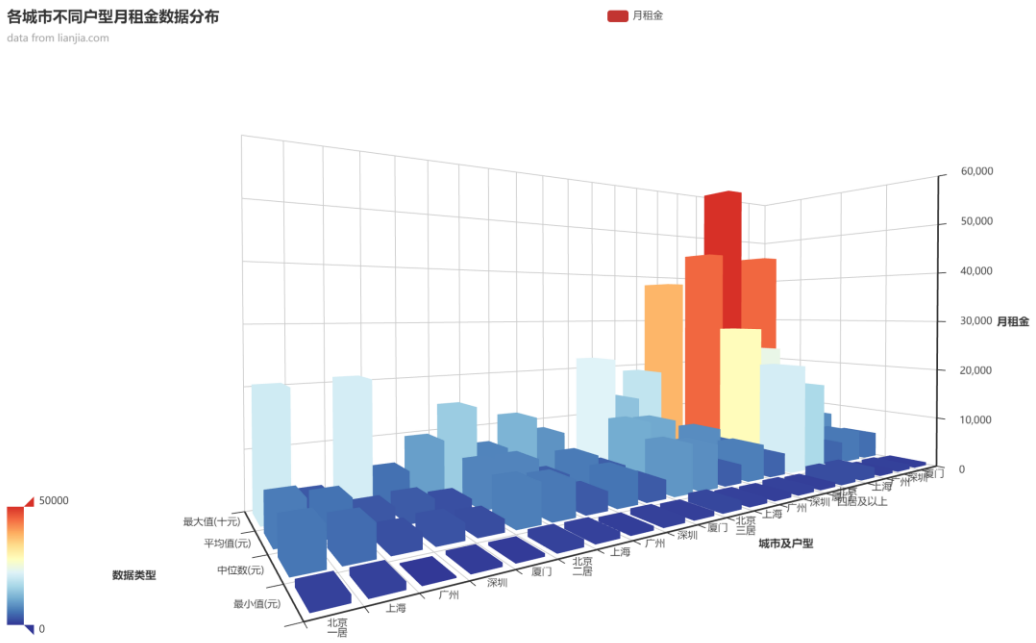


从图中可以看到，四居以上类型的平均月租金要显著高于其他户型。

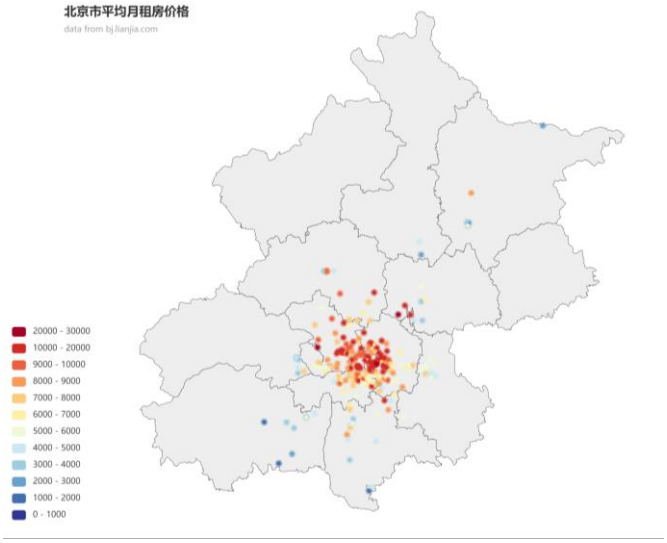
- **分析最高月租金**，该部分数据没有明显的户型趋或地域趋势。猜测是因为最高值的影响因素较多，包含建筑的面积、装修程度、地理位置等。也有可能存在部分链家官网数据源数据错误的情况。

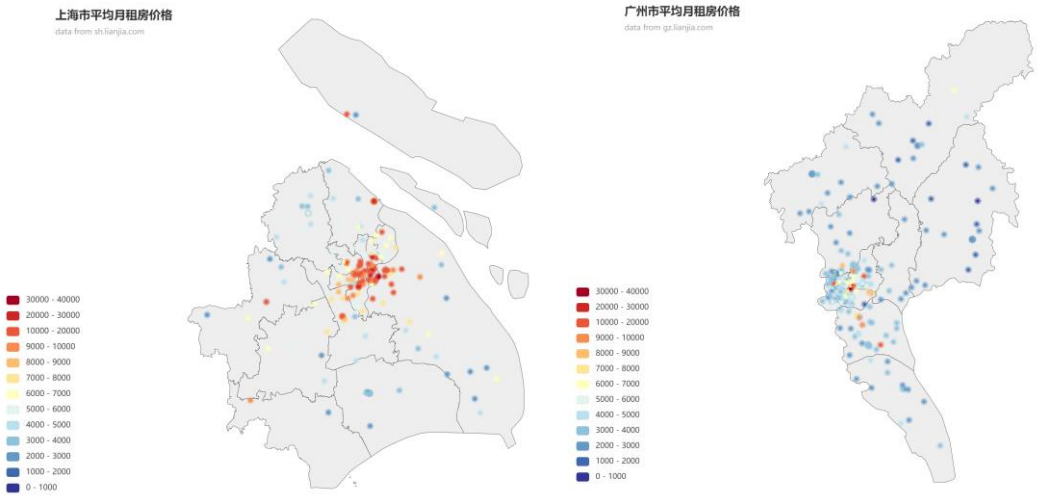


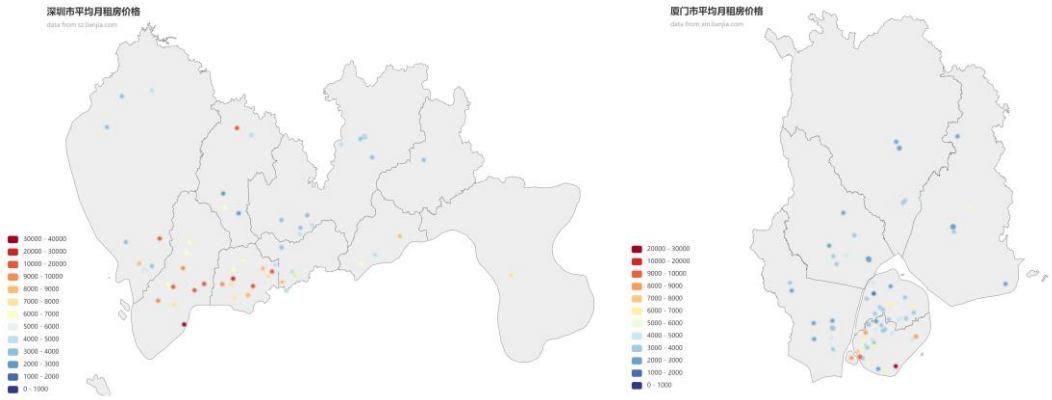
- **分析平均值和中位数，针对单一户型，**北京市、上海市的月租金都要高于其他城市。而一线城市广州和二线城市厦门的月租金差异不大。
- **分析平均值和中位数，针对单一城市**，一居、二居、三居的月租金均呈现递增趋势。居室数量越多，平均租金也会越高。



6.2.3、CBD 板块与均价分布



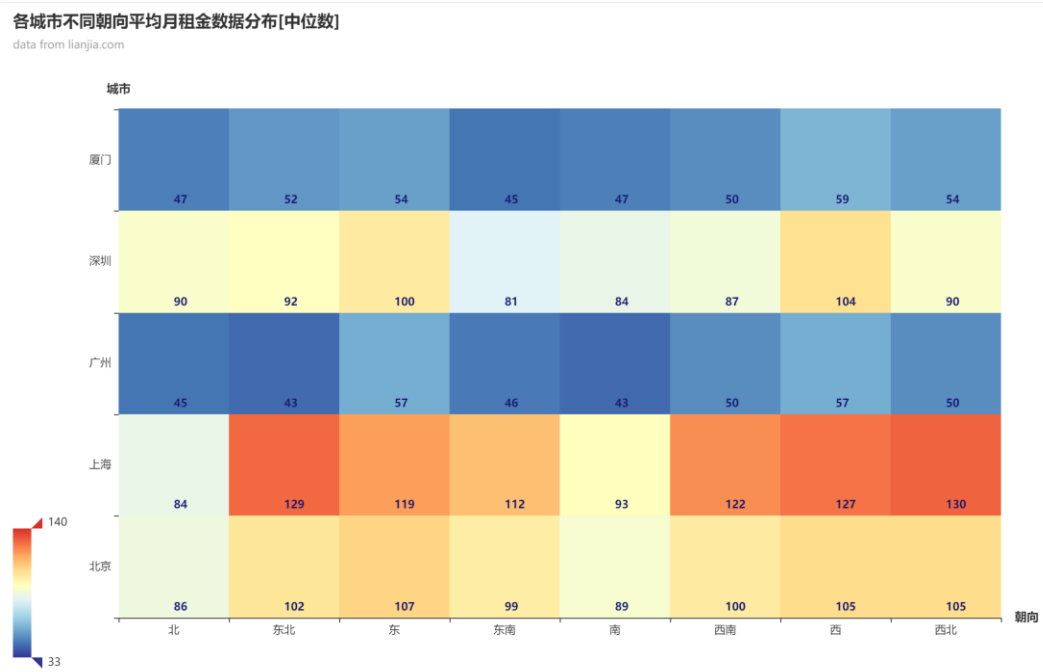




根据 CBD 商圈/板块分布及平均月租金的散点分布图，可以明显的看出，对于绝大部分房源数据，靠近市中心的房源的平均月租房价格要高于市区边缘的房源的平均月租房价格。且平均月租金呈现出由靠近市中心向远离市中心的递减趋势。此外，在靠近市中心的部分，板块分布也更加密集。

可以合理的猜测，越靠近市中心，各类政企商业及服务的分布更加密集，租房需求也要高于原理市中心的部分。在远离市中心的部分，由于交通到市中心的时间成本更高，其租房平均月租金也要相对较低。

6.2.4、不同朝向对比分析



从上图中可以看出，不同城市租房数据的方向趋向并不一致。

- **北京：**东方向最高，北方向最低
- **上海：**西北方向最高，北方向最低
- **广州：**东/西方向最高，南/东北方向最低
- **深圳：**西方向最高，东南方向最低
- **厦门：**西方向最高，东南方向最低

在互联网上查询**房屋朝向的选择倾向性**，得知：

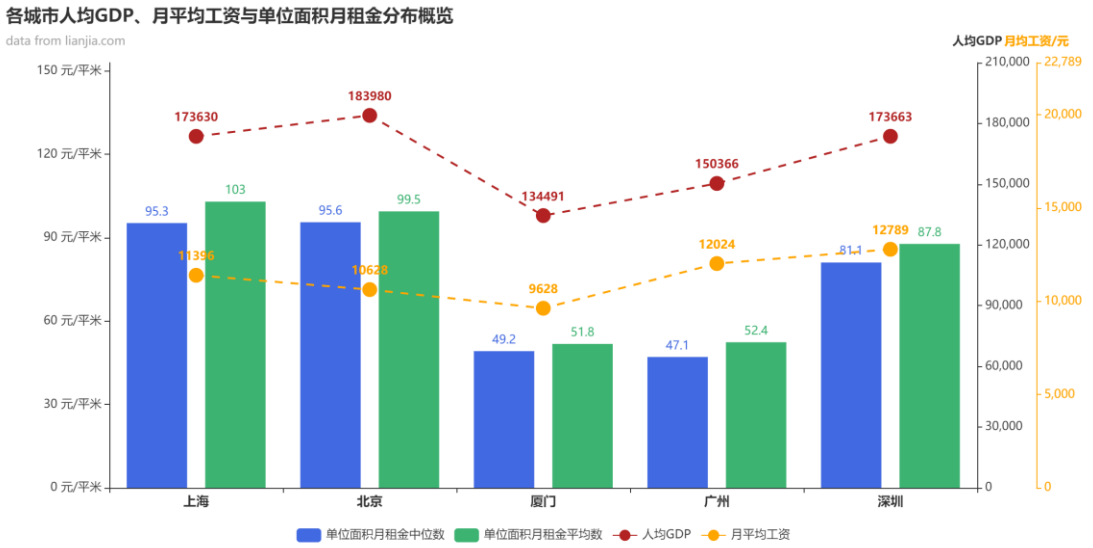
- **北：**采光差，夏天凉快但冬天阴冷
- **南：**阳光充足，冬暖夏凉
- **西：**下午采光充足
- **东：**上午采光充足

在各个城市的数据倾向中可以大致看出：靠北的城市(北京/上海)其北方向房屋均价较低；靠南的城市(广州/深圳/厦门)其南方向房屋均价较低。猜测可能是气候的主导作用。

实际上，在各个城市租房的月租金的影响因素中，地理位置分布(CBD 商圈分布)的影响要远远大于其房屋朝向的影响，这也能够解释为什么在房屋朝向分布中并没有比较明确的分布趋势。

6.2.5、人均 GDP/平均工资及单位面积租金分析

*各城市人均 GDP 及月平均工资来源于各城市 2022 统计年鉴和中国 2022 统计年鉴



从上图知，各城市人均 GDP、月平均工资和单位面积月租金总体上呈现正相关的趋势。

- **分析单位面积月租金与人均 GDP 的关系**

可以看到北京、上海的单位面积月租金较其他三个城市都要高。因此在上海、北京的租房性价比较其他三个城市低。

对于人均 GDP 相近的上海和深圳，深圳的单位面积月租金明显低于上海，因此在深圳租房的性价比要高于上海。

对比广州和深圳，广州的单位面积月租金相交深圳减少约 40%，而人均 GDP 相差 2 万元，因此广州的租房性价比要高于深圳。

对于单位面积月租金相近的厦门和广州，广州的人均 GDP 更高，因此在广州租房的性价比要高于厦门。

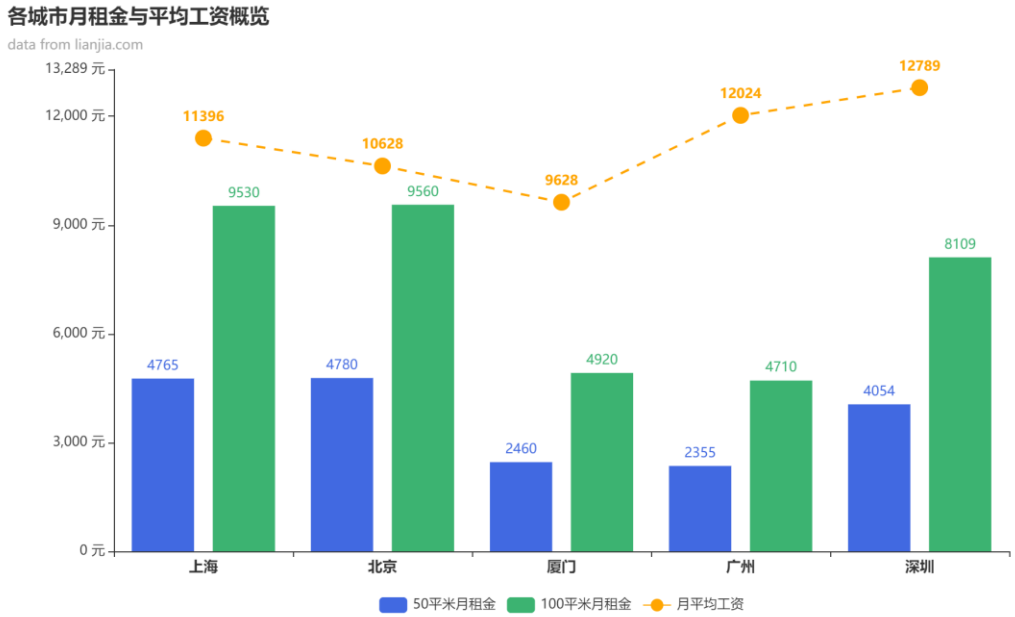
总结上述分析可得，在五个城市中，租房性价比由高到低大致为：

广州 > 厦门/深圳 > 上海 > 北京

- **分析单位面积月租金与平均工资的关系**

以单位面积月租金中位数为基础数据，展示租 50 平米和 100 平米的月租金和月均工

资的关系如下：



可以很明显的看出，在租房后，广州、深圳、厦门的盈余资金要明显高于上海和北京。这也意味着，在租房指出相近时，在上海和北京租房面积要小于广州和深圳。

上图非常明确的展示出，北京的租房负担时五个城市中最重的。其他城市的租房负担大致可以排序为：

北京 > 上海 > 深圳 > 厦门 > 广州

七、总结

经过对链家官网(lianjia.com)中北京市、上海市、广州市、深圳市、厦门市五个城市共 162593 条有效租房数据的获取和分析，大致可以得到以下结论：

1、(单位面积)月平均租金分布城市顺序：**北京 > 上海 > 深圳 > 厦门 > 广州**
2、(单位面积)月平均租金分布户型顺序：**三居 > 二居 > 一居**
3、**靠近**市中心的 CBD，平均月租金 > **远离**市中心的 CBD 的平均月租金
4、靠北的城市(北京/上海)北方向房屋均价较低；靠南的城市(广州/深圳/厦门)南方向房屋均价较低。但总体上依旧取决于地理位置分布(CBD 商圈分布)的影响
5、租房性价比由高到低依次为：**广州 > 厦门/深圳 > 上海 > 北京**
6、租房负担由大到小依次为：**北京 > 上海 > 深圳 > 厦门 > 广州**