2023 Python程序设计大作业：租房数据分析

1. 数据抓取

要求抓取链家官网北上广深4个一线城市，再加上一个离你家乡最近的一个非一线城市/或者你最感兴趣的一个城市的租房数据。

页面分析

通过谷歌浏览器提供的开发者工具，能找到租房信息的网页html中有我们所需要爬取的信息,并且都位于一个div元素下 `<div class="content__list"></div>`

所以我们可以通过这种XPath路径进行爬取，`response.xpath("//*[@id='content']/div[1]/div[1]/*")`

然后通过静态和动态调试，能够发现我们正确获取的所需要的信息，其中一个模块如下

```
    <div
class="content__list--item"
data-el="listItem"
data-house_code="BJ18160095031334010 88"
data-brand_code="200301001000"
data-ad_code="0"
data-bid_version=""
data-c_type="1"
data-position="0"
data-total="66377"
data-fb_expo_id="786707837408985088"
data-t="default"
data-strategy_id=""
data-click_position="0"
data-ad_type="0"
data-distribution_type="203500000001"
data-event_id="21333"
data-event_action="click_position=0"
data-event_position="click_position"
data-event_send="no"
data-rank_score="-1"
data-operation_score="0"
>
<a
class="content__list--item--aside" target="_blank"
href="/zufang/BJ18160095031334010 88.html"
title="整租·乐想汇 1房间 南">
  <img
    alt="整租·乐想汇 1房间 南_乐想汇租房"
    src="https://s1.ljcdn.com/matrix_pc/dist/pc/src/resource/default/250-182.png?
_v=20231122104722f2f"
    data-src="https://s1.ljcdn.com/matrix_pc/dist/pc/src/resource/default/250-
182_1.png?_v=20231122104722f2f"
    class="lazyload">
```

```
    <!-- 是否展示vr图片 -->
            <!-- 是否展示省心租图片 -->
            <!-- 广告标签 -->
        </a>
<div class="content__list--item--main">
<p class="content__list--item--title">
  <a class="twoline" target="_blank" href="/zufang/BJ1816009503133401088.html">
    整租·乐想汇 1房间 南        </a>
                </p>
<p class="content__list--item--des">
          <a target="_blank" href="/zufang/chaoyang/">朝阳</a>-<a
href="/zufang/beiyuan2/" target="_blank">北苑</a>-<a title="乐想汇"
href="/zufang/c1111046126379/" target="_blank">乐想汇</a>
  <i>/</i>
  68.00㎡
  <i>/</i>南        <i>/</i>
    1房间1卫        <span class="hide">
    <i>/</i>
    低楼层                          (18层)
            </span>
</p>
<p class="content__list--item--bottom oneline">
      <i class="content__item__tag--gov_certification">官方核验</i>
      <i class="content__item__tag--is_subway_house">近地铁</i>
      <i class="content__item__tag--central_heating">集中供暖</i>
      <i class="content__item__tag--is_key">随时看房</i>
      </p>
<p class="content__list--item--brand oneline">
            <span class="brand">
      链家          </span>
          <span class="content__list--item--time oneline">2天前维护</span>
</p>
      <span class="content__list--item-price"><em>5100</em> 元/月</span>
</div>

</div>
```

通过谷歌浏览器开发者工具提供的XPath复制和查找功能，我们可以找到我们所需要的信息的XPath路径。

这样创建几个对应的item，就可以将数据存储到对应的item中，然后再通过pipelines.py中的代码，将数据存储到对应的csv文件中。

代码编写

1. 爬虫代码

使用之前的python爬虫框架，重新编写了5个爬虫（分别对应北京，上海，广州，深圳，以及郑州5个城市），并且将爬虫的代码进行了封装，使得代码更加简洁，易于阅读。

下面以bj代码为例，进行说明：

```python
import scrapy

from test1.items import Test1Item  # 从items.py中引入MyItem对象


class LianjiaSpider(scrapy.Spider):
    name = 'bj'
    allowed_domains = ['bj.lianjia.com']
    # https://bj.lianjia.com/zufang/pg2/
    base_url = 'https://bj.lianjia.com/zufang/'


    maxpage = 200
    # 最大爬取页数

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.download_delay = 1
        with open("bj.txt", "r") as f:
            # 读取上次爬取的页数
            self.begin_index = int(f.read())
            self.page_index=self.begin_index
            self.start_urls = [self.base_url + "pg" + str(self.begin_index)]
            print(self.begin_index)
            f.close()
        with open("bj.txt","w") as f:
            # 更新爬取的页数
            f.write(str(self.begin_index+self.maxpage))
            f.close()

    def parse(self, response, **kwargs):
        for each in response.xpath("//*[@id='content']/div[1]/div[1]/*"):
            item = Test1Item()
            item['price'] = each.xpath("div/span/em/text()").extract_first()
            if '-' in item['price']:
                temp = item['price'].split("-")
                item['price'] = (float(temp[0]) + float(temp[1])) / 2
            temp = each.xpath("div/p[1]/a/text()").extract_first()
            # 从temp中提取数据
            temp = temp.split(" ")
            item['name'] = temp[0]
            item["house_type"] = ""
            item["area"] = ""
            item["direct"] = ""
            item["name_chinese"] = "北京"
            for i in temp:
                if '室' in i or '厅' in i or '卫' in i:
                    item['house_type'] = i
                if '东' in i or '南' in i or '西' in i or '北' in i:
                    if len(i) < 5:
                        item['direct'] = i
            temp = each.xpath("div/p[2]/text()").extract()
            for i in temp:
```

```
                if 'm²' in i:
                    i = i.replace("m²", "").replace(" ", "").replace("\n", "")
                    if "-" in i:
                        temp = i.split("-")
                        item['area'] = (float(temp[0]) + float(temp[1])) / 2
                    else:
                        item['area'] = i
            item["block"]=each.xpath("div/p[2]/a[2]/text()").extract_first()
            print(item["block"])
            if item['price'] and item['name'] and item['house_type'] and
item['area'] and item['direct'] and item["block"]:
                yield item
        self.page_index += 1
        if self.page_index > self.maxpage+self.begin_index-1:
            return
        url=self.base_url+"pg"+str(self.page_index)

        print(url)

        yield scrapy.Request(url, callback=self.parse)
        # 递归爬取下一页
```

通过一个bj.txt文件，记录上次爬取的页数，然后通过递归爬取下一页的方式，实现了爬取多页的功能。这样在多次爬取时只需要修改爬虫的名字和pipe中输出的文件名就能实现多个城市的爬取。

2. 数据处理代码

在pipelines.py中，我们对数据进行了处理，将数据存储到对应的csv文件中。

```
import json
import csv
class Test1Pipeline:

    def open_spider(self, spider):
        try:
            self.file = open('zz.csv', "a", encoding="utf-8-sig", newline='')
            self.writer = csv.writer(self.file)
        except Exception as err:
            print(err)

    def process_item(self, item, spider):
        list_item = [item['name_chinese'], item['block'],item['house_type'],
item['direct'], item['area'], item['price']]
        self.writer.writerow(list_item)
        return item
    def close_spider(self, spider):
            self.file.close() #关闭文件
```

3. 运行代码

在main.py中，我们通过调用execute函数，运行爬虫代码。

```
from scrapy import cmdline
cmdline.execute("scrapy crawl zz".split())
```

2. 数据获取

应获取每个城市的全部租房数据（一线城市的数据量应该在万的数量级）。

运行代码

为防止爬取大量数据时，被网站封禁，我们将爬取的数据分为多次爬取，每次爬取300-500页，然后将爬取的数据存储到对应的csv文件中。

发现问题

本来以为爬取的数据量不会很大，但是在爬取的过程中，发现了一些问题：

开始通过chorme浏览器发现https://bj.lianjia.com/zufang/pg1233/这个网页是存在的，但是通过爬虫爬取时，发现这个网页中的数据很不友好，实际上是重复的，这就很让人头疼。

想到的解决方法是重写一下parse函数，对每个区进行遍历，如果数据再不够的话只能进行进一步细分了

同样的，以bj.py 为例，进行说明：

```
import scrapy

from test1.items import Test1Item  # 从items.py中引入MyItem对象


class LianjiaSpider(scrapy.Spider):
    name = 'bj'
    allowed_domains = ['bj.lianjia.com']
    zone_list_chinese=["东城","西城","朝阳","海淀","丰台","石景山","通州","昌平","大
兴","亦庄开发区","顺义","房山","门头沟","平谷","怀柔","密云","延庆"]

    zone_list=
["dongcheng","xicheng","chaoyang","haidian","fengtai","shijingshan","tongzhou","ch
angping","daxing","yizhuangkaifaqu","shunyi","fangshan","mentougou","pinggu","huai
rou","miyun","yanqing"]
    # https://bj.lianjia.com/zufang/pg2/
    zone_list_number=
[100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100]
    base_url = 'https://bj.lianjia.com/zufang/'
    maxpage = 500
    def __init__(self, **kwargs):
        // 读取上次爬取的页数
        super().__init__(**kwargs)
        self.download_delay = 1
        with open("bj.txt", "r") as f:
```

```
            temp=f.read().split("\n")
            self.zone_index=int(temp[0])
            self.page_index =int (temp[1])

            self.start_urls = [self.base_url
+self.zone_list[self.zone_index]+"/pg"  + str(self.page_index)]
            print(self.zone_index,self.page_index)
            f.close()
    def parse(self, response, **kwargs):

        for each in response.xpath("//*[@id='content']/div[1]/div[1]/*"):
            item = Test1Item()
            item['price'] = each.xpath("div/span/em/text()").extract_first()
            if '-' in item['price']:
                temp = item['price'].split("-")
                item['price'] = (float(temp[0]) + float(temp[1])) / 2
            temp = each.xpath("div/p[1]/a/text()").extract_first()
            # 从temp中提取数据
            temp = temp.split(" ")
            item['name'] = temp[0]
            item["house_type"] = ""
            item["area"] = ""
            item["direct"] = ""
            item["name_chinese"] = "北京"
            for i in temp:
                if '室' in i or '厅' in i or '卫' in i:
                    item['house_type'] = i
                if '东' in i or '南' in i or '西' in i or '北' in i:
                    if len(i) < 5:
                        item['direct'] = i
            temp = each.xpath("div/p[2]/text()").extract()
            for i in temp:
                if '㎡' in i:
                    i = i.replace("㎡", "").replace(" ", "").replace("\n", "")
                    if "-" in i:
                        temp = i.split("-")
                        item['area'] = (float(temp[0]) + float(temp[1])) / 2
                    else:
                        item['area'] = i
            item["block"]=each.xpath("div/p[2]/a[2]/text()").extract_first()
            print(item["block"])
            item["zone_name"] = self.zone_list_chinese[self.zone_index]
            if item['price'] and item['name'] and item['house_type'] and
item['area'] and item['direct'] and item["block"]:
                yield item
        if self.maxpage <= 0:
            # 爬取完毕，这样写很丑陋，但是我写在析构函数里面不会执行不知道为什么，也许是
框架本身就不甚严谨
            with open("bj.txt", "w") as f:
                f.write(str(self.zone_index))
                f.write("\n")
                f.write(str(self.page_index))
                f.close()
            return
```

```
        // 通过判断是否有下一页，来决定是否进行下一页的爬取
        if response.xpath("//*[@id='content']/div[1]/div[2]/@data-
totalpage").extract_first() :
            self.zone_list_number[self.zone_index]=int(response.xpath("//*
[@id='content']/div[1]/div[2]/@data-totalpage").extract_first())

print("self.zone_list_number[self.zone_index]",self.zone_list_number[self.zone_ind
ex])
        self.page_index += 1
        if self.zone_index < len(self.zone_list):
            if self.page_index <= self.zone_list_number[self.zone_index]:

url=self.base_url+self.zone_list[self.zone_index]+"/pg"+str(self.page_index)
            else:
                self.page_index = 1
                url = self.base_url + self.zone_list[self.zone_index]
                self.zone_index += 1
        else:
            with open("bj.txt", "w") as f:
                f.write(str(self.zone_index))
                f.write("\n")
                f.write(str(self.page_index))
                f.close()
            return
        self.maxpage -= 1
        yield scrapy.Request(url, callback=self.parse)
```

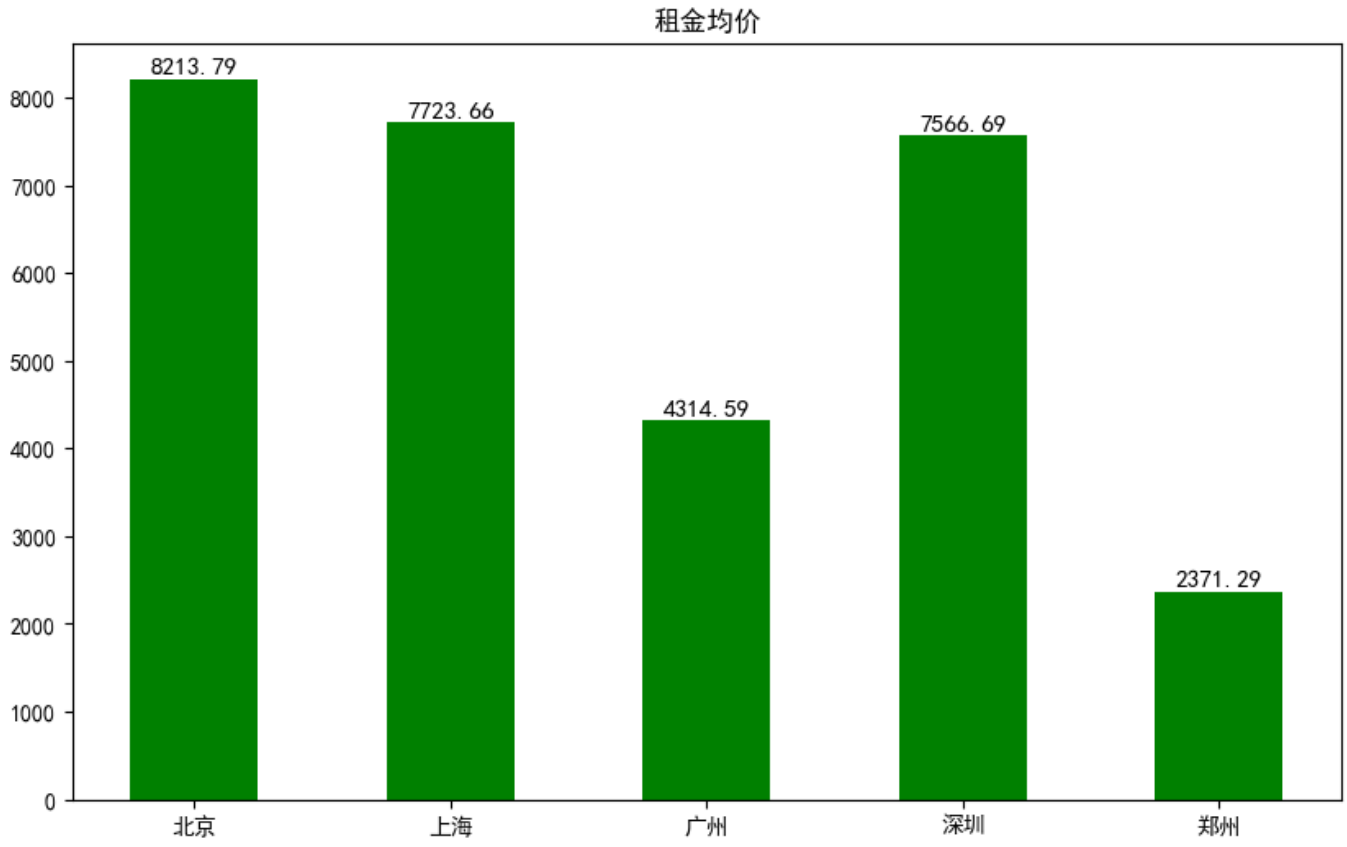
3. 总体房租情况比较

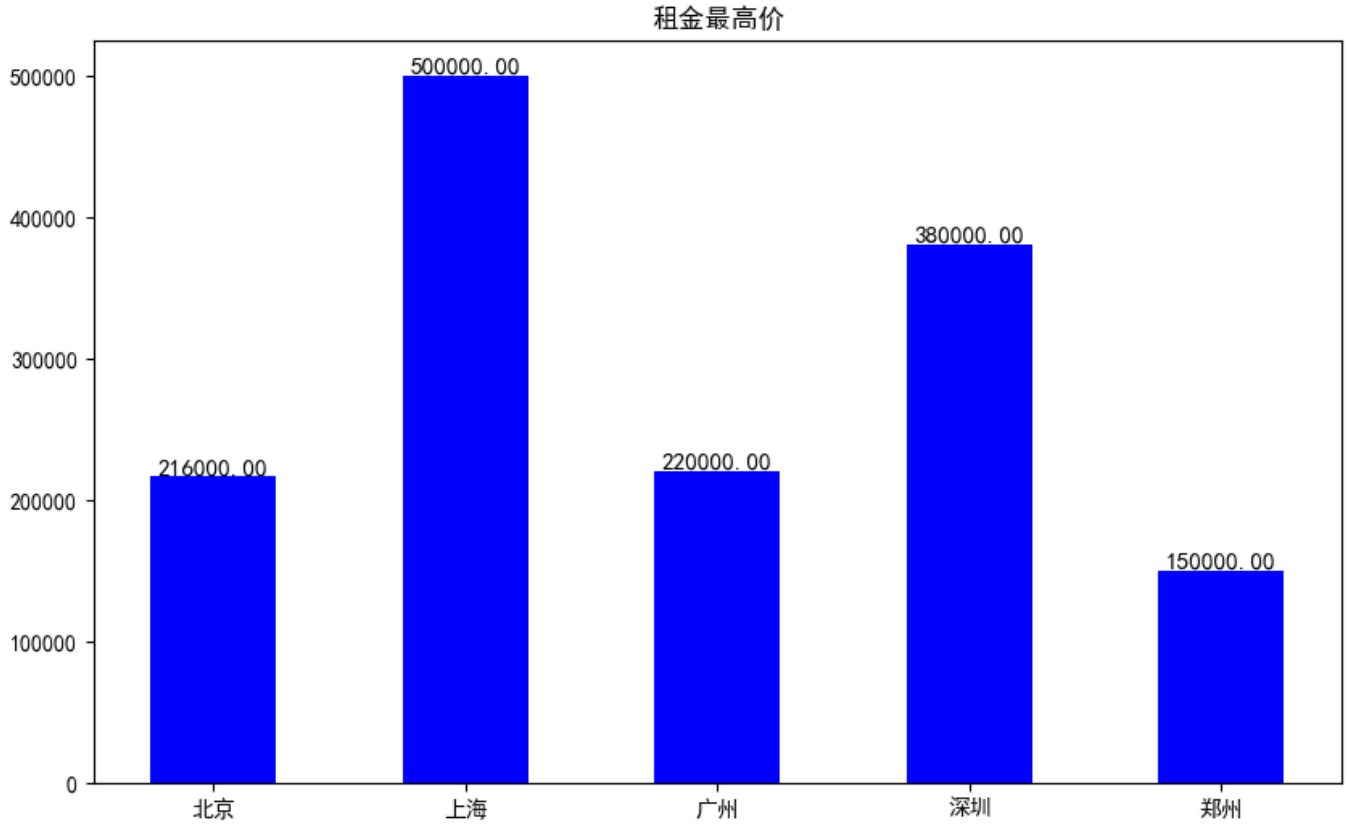
比较5个城市的总体房租情况，包含租金的均价、最高价、最低价、中位数等信息，单位面积租金（元/平米）的均价、最高价、最低价、中位数等信息。

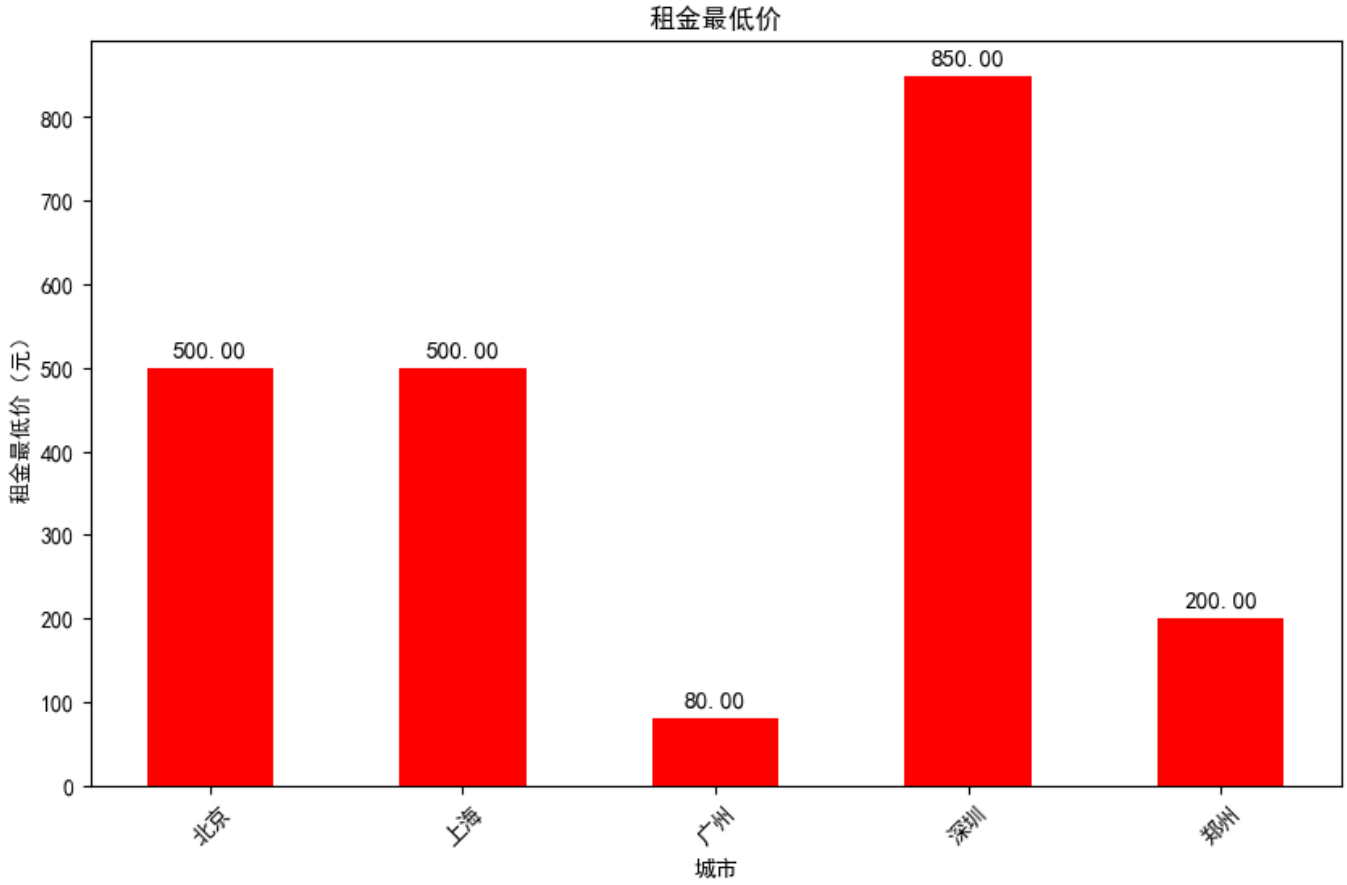
代码编写

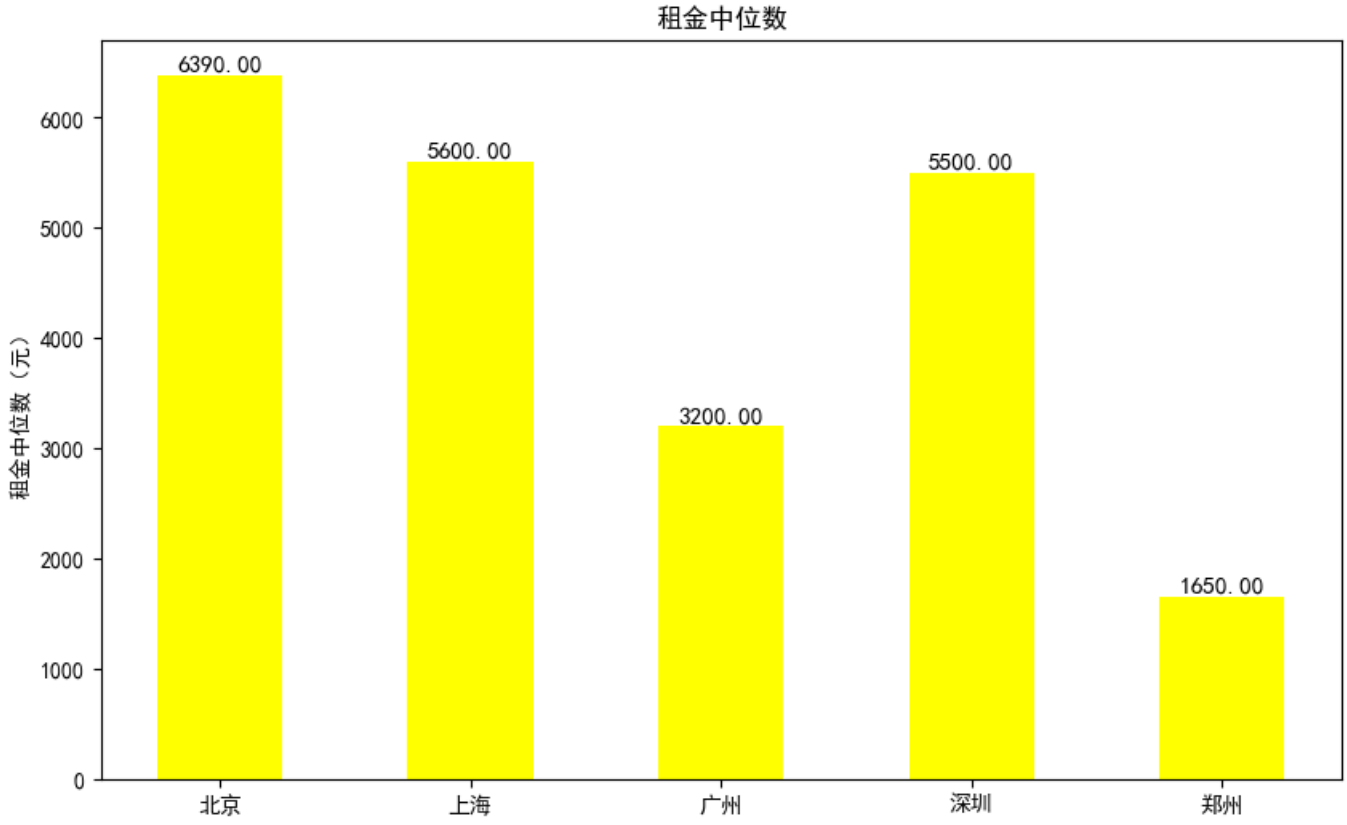
这里使用jupyter进行编写，因为能方便的进行输出，以及进行数据的可视化。看起来会舒服许多许多 不过缺点是有点长了，所以这里仅仅加入一个引用的代码总体和户型

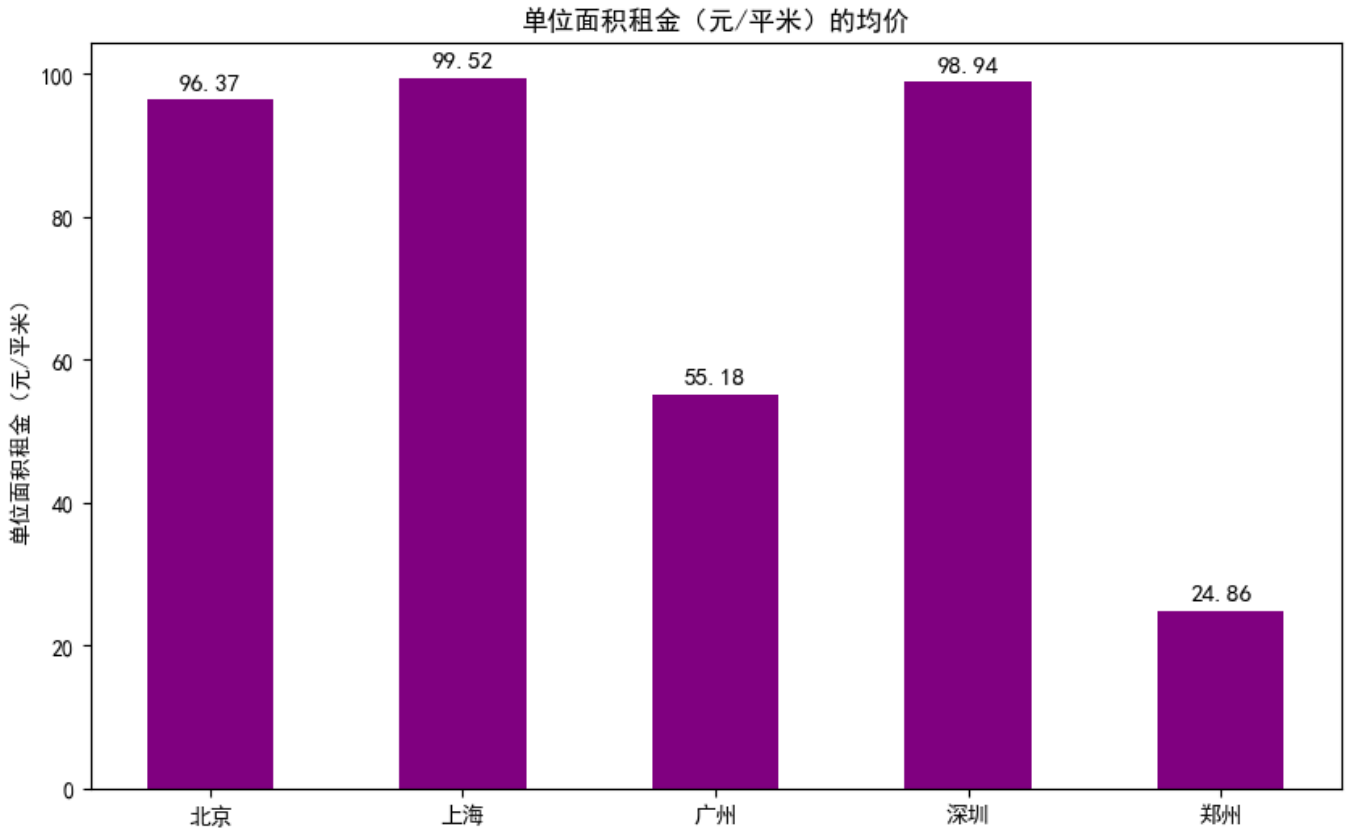
结果展示

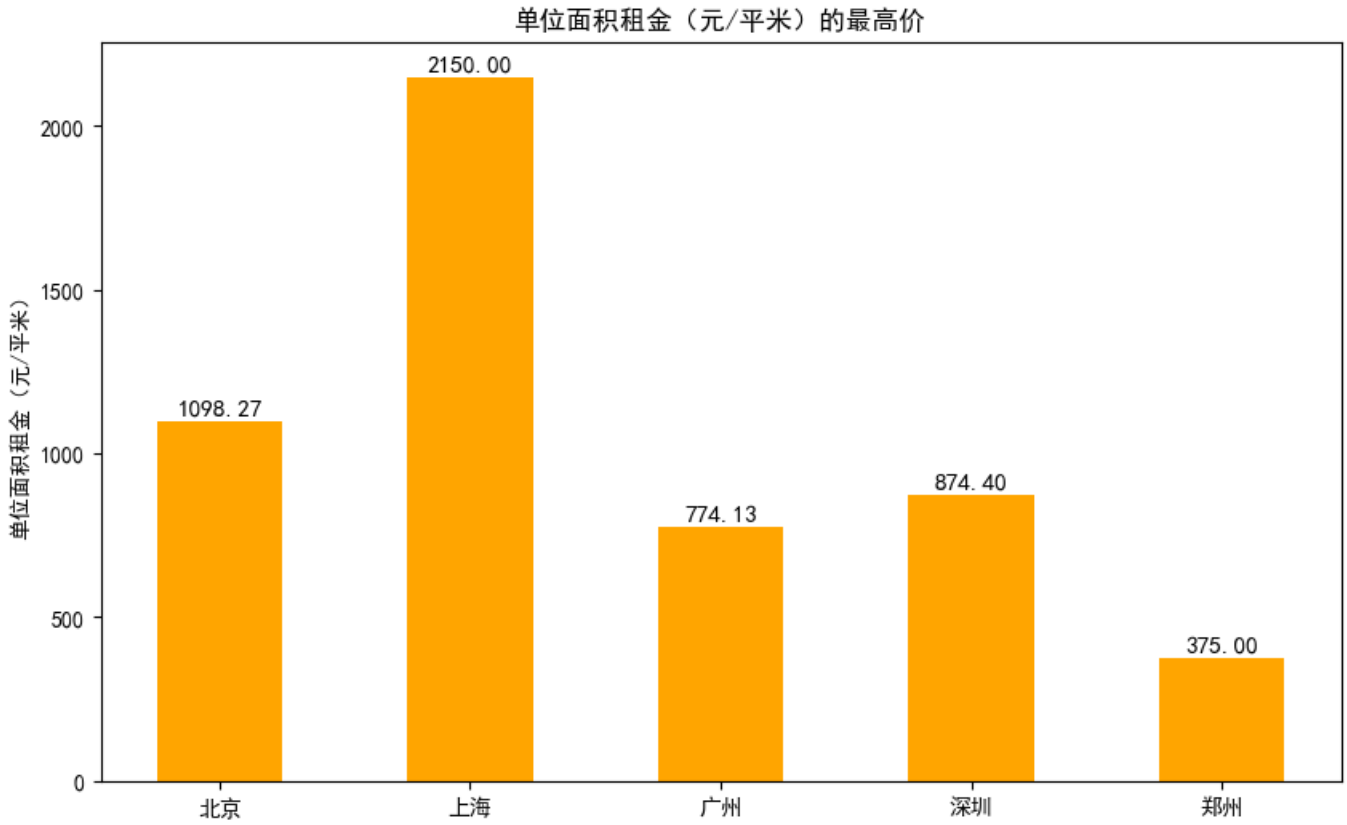

租金均价
8213.79
7723.66
4314.59
7566.69
2371.29
0
1000
2000
3000
4000
5000
6000
7000
8000
北京
上海
广州
深圳
郑州

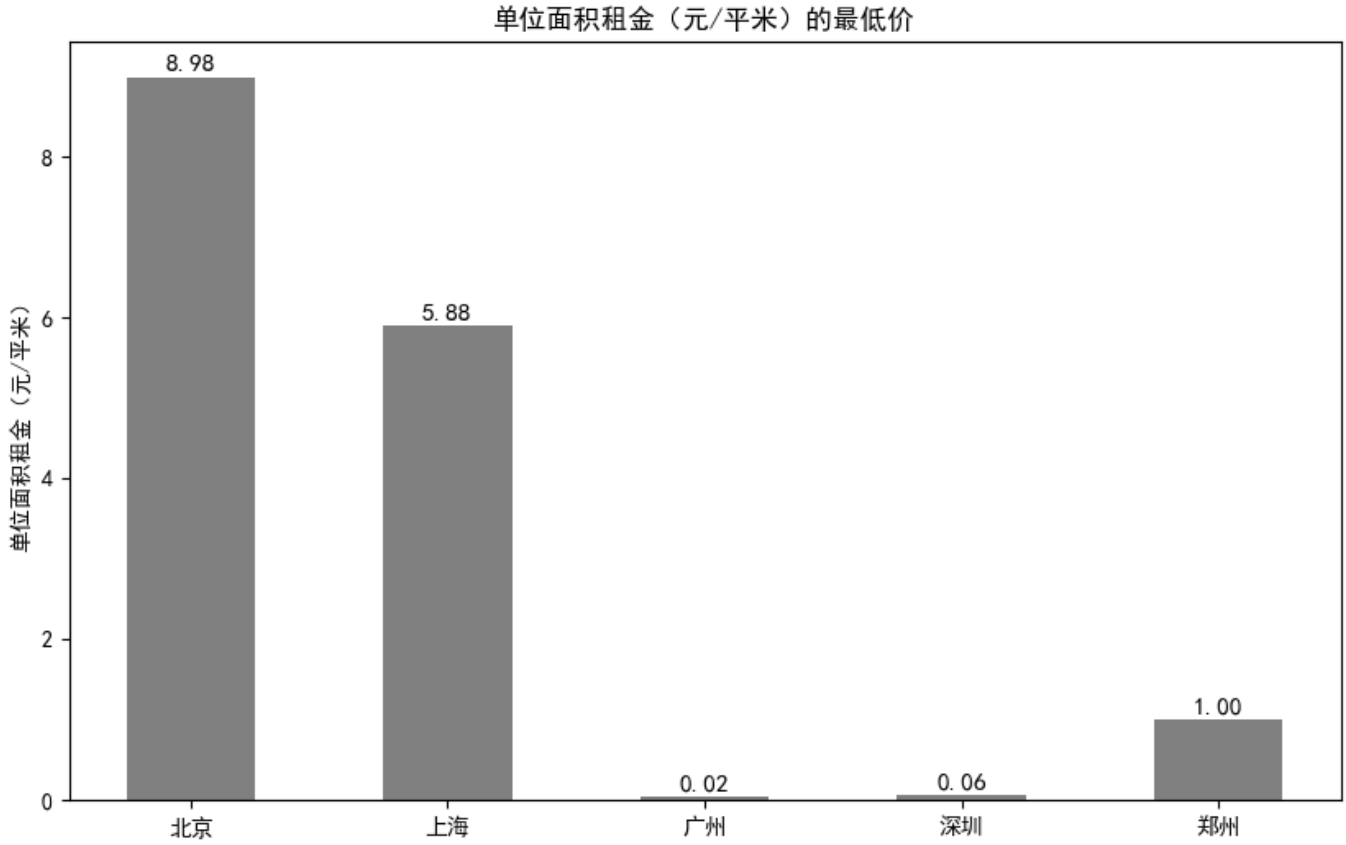

租金最高价
500000.00
380000.00
216000.00
220000.00
150000.00
0
100000
200000
300000
400000
500000
北京
上海
广州
深圳
郑州

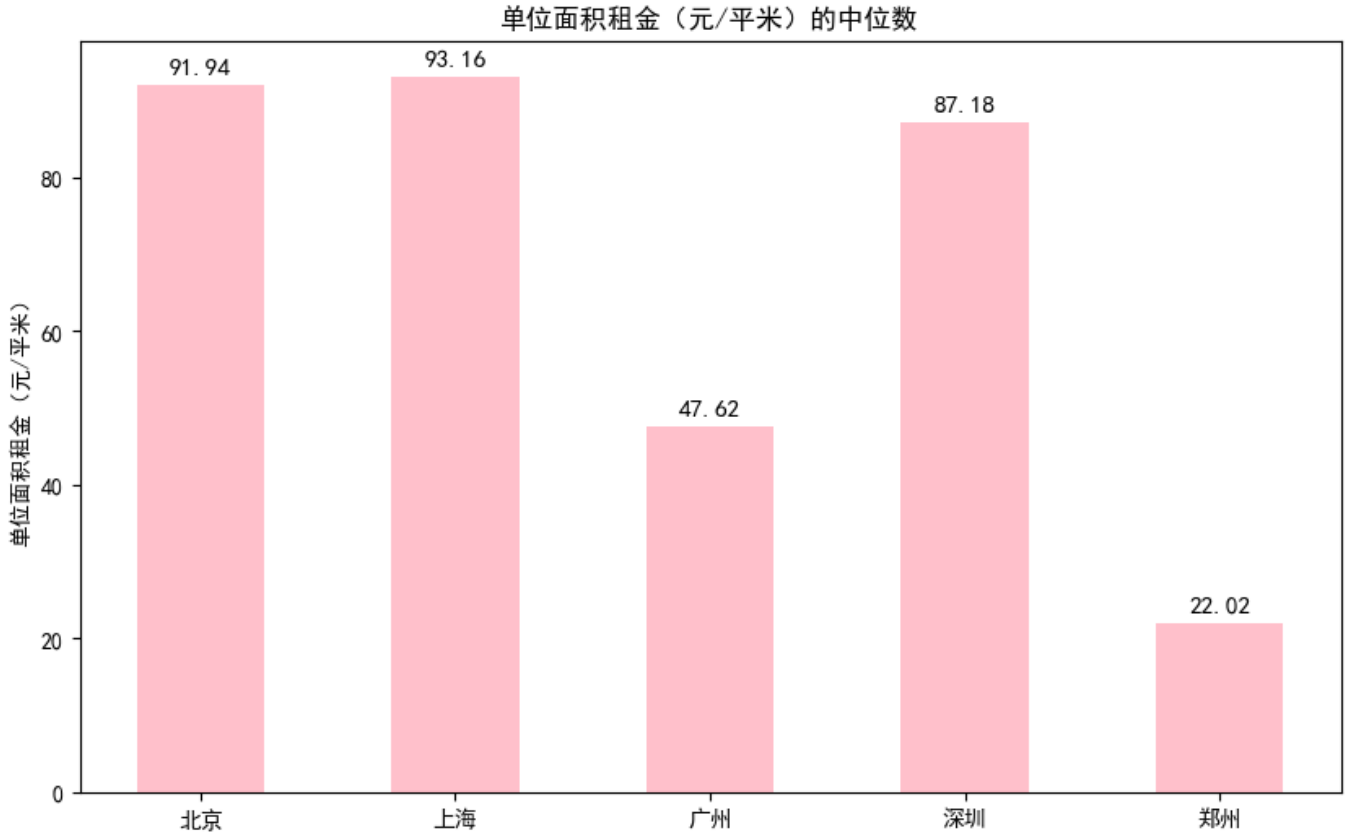

租金最低价
850.00
500.00
500.00
80.00
200.00
800
700
600
500
400
300
200
100
0
租金最低价（元）
北京
上海
广州
深圳
郑州
城市


租金中位数
6390.00
5600.00
3200.00
5500.00
1650.00
6000
5000
4000
3000
2000
1000
0
租金中位数（元）
北京
上海
广州
深圳
郑州


单位面积租金（元/平米）的均价
单位面积租金（元/平米）
0
20
40
60
80
100
96.37
99.52
55.18
98.94
24.86
北京
上海
广州
深圳
郑州


单位面积租金（元/平米）的最高价
单位面积租金（元/平米）
0
500
1000
1500
2000
1098.27
2150.00
774.13
874.40
375.00
北京
上海
广州
深圳
郑州





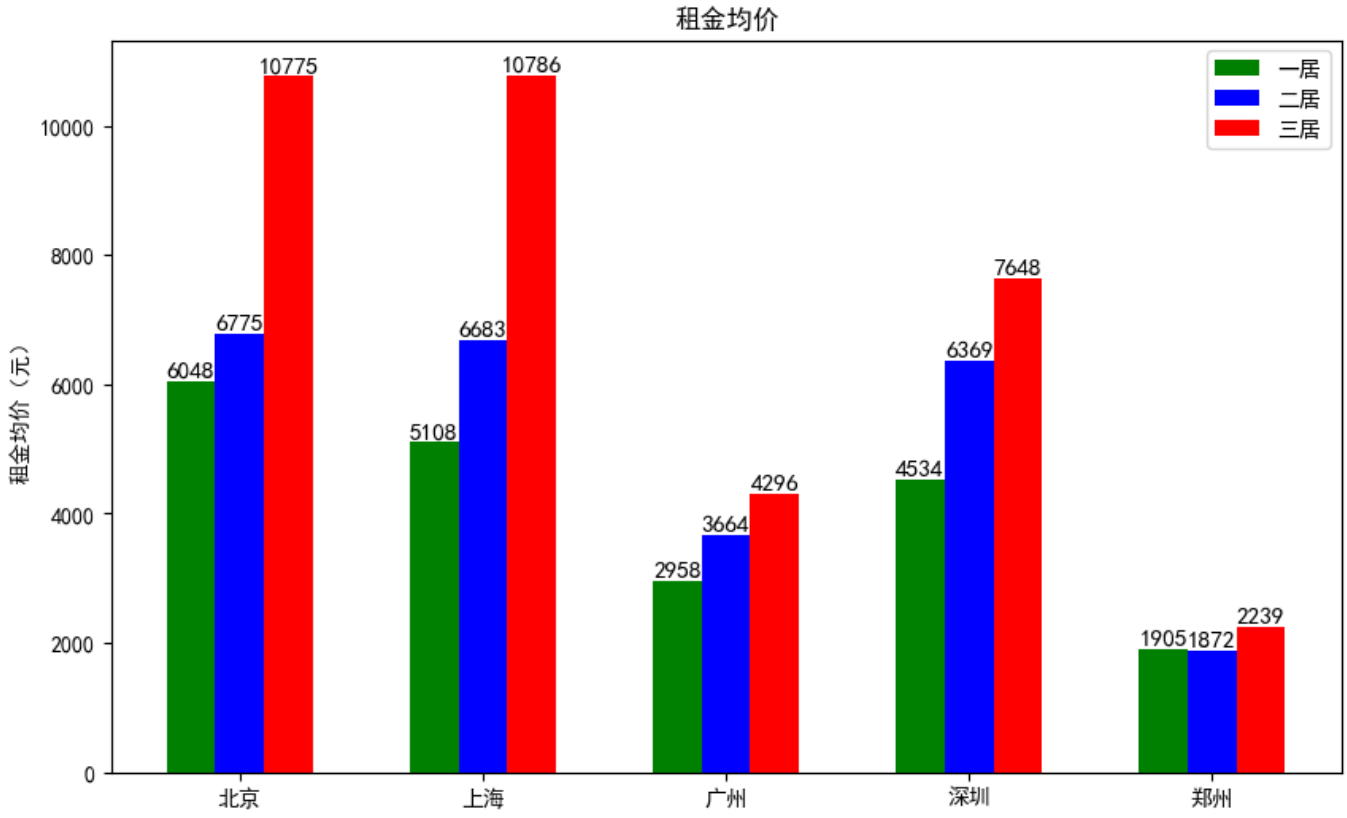
4. 不同户型比较

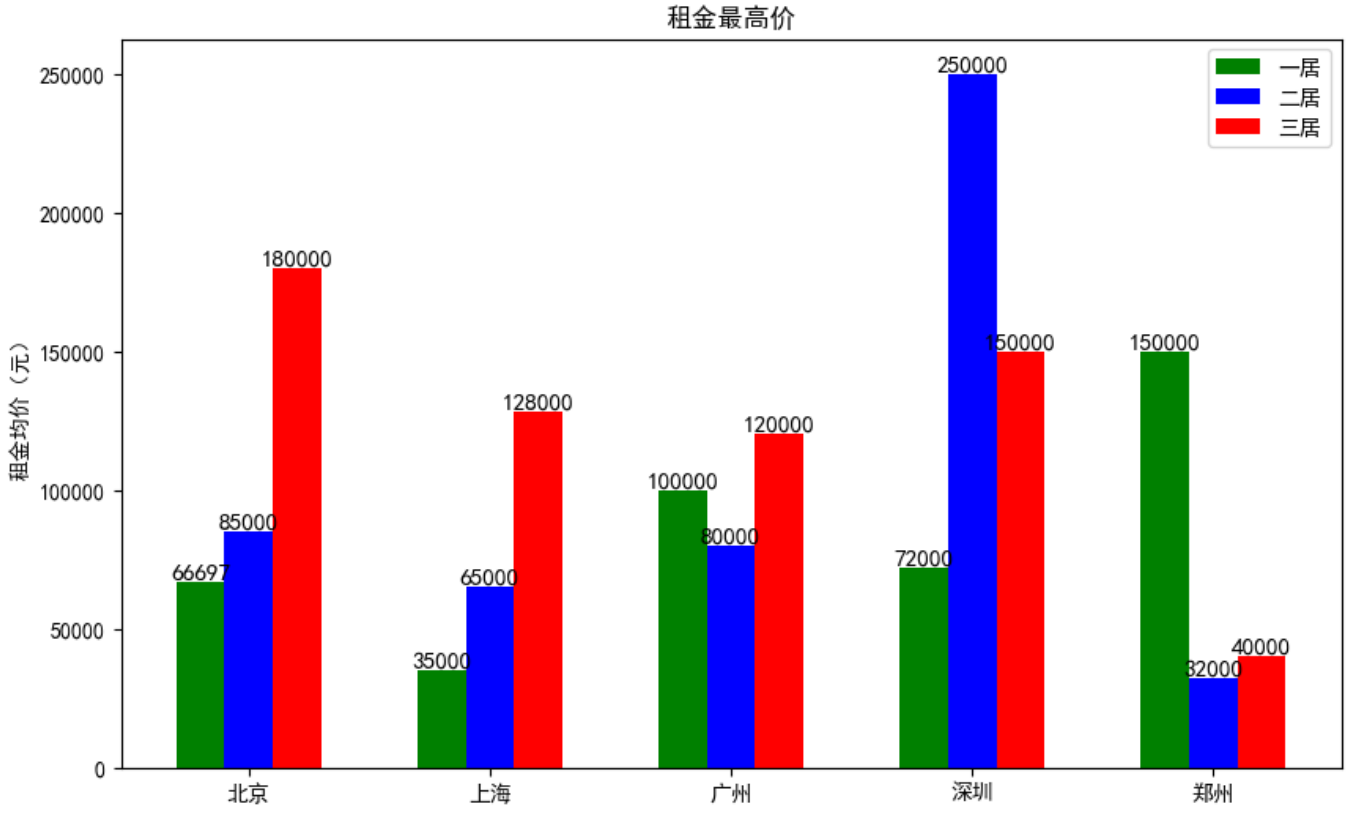
比较5个城市一居、二居、三居的情况，包含均价、最高价、最低价、中位数等信息，采用合适的图或表形式进行展示。

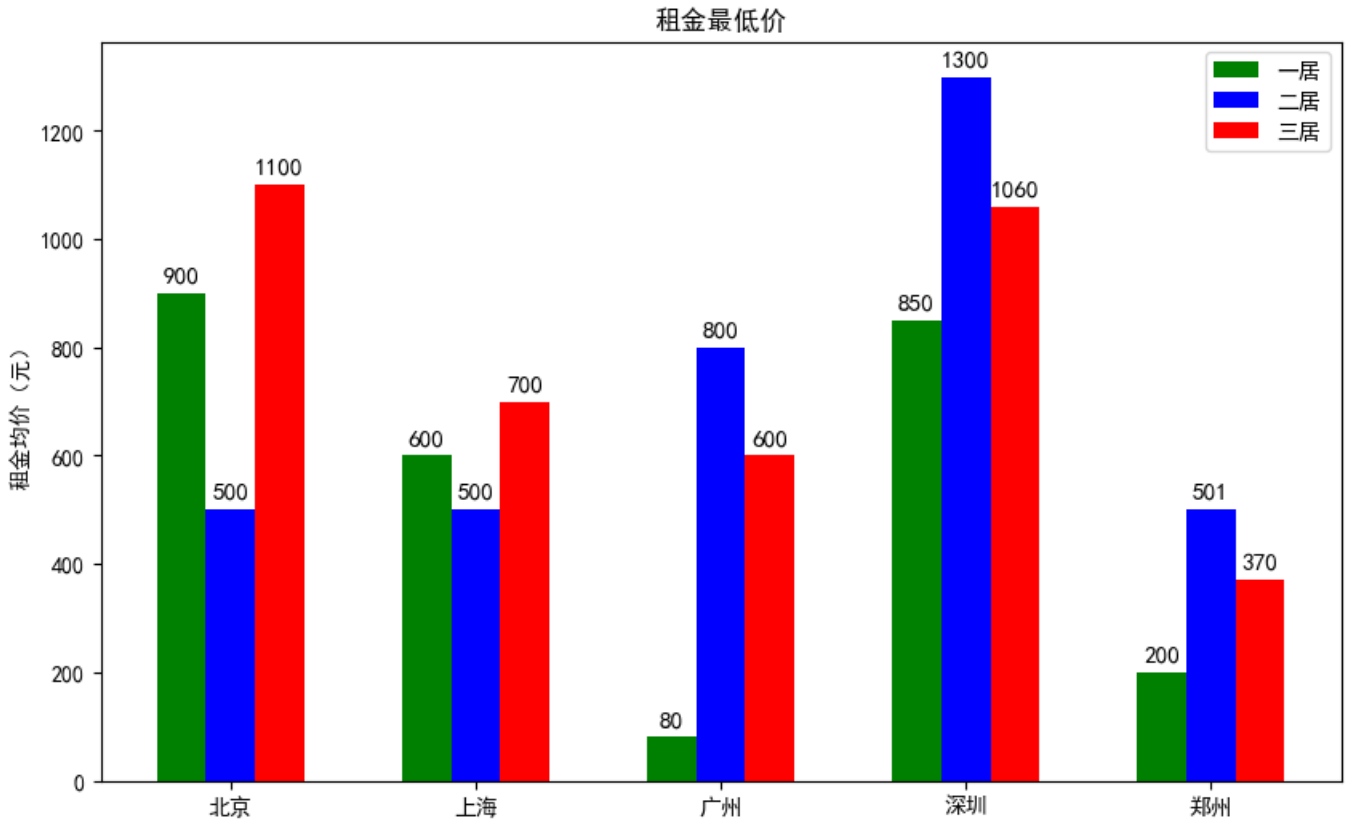
代码编写

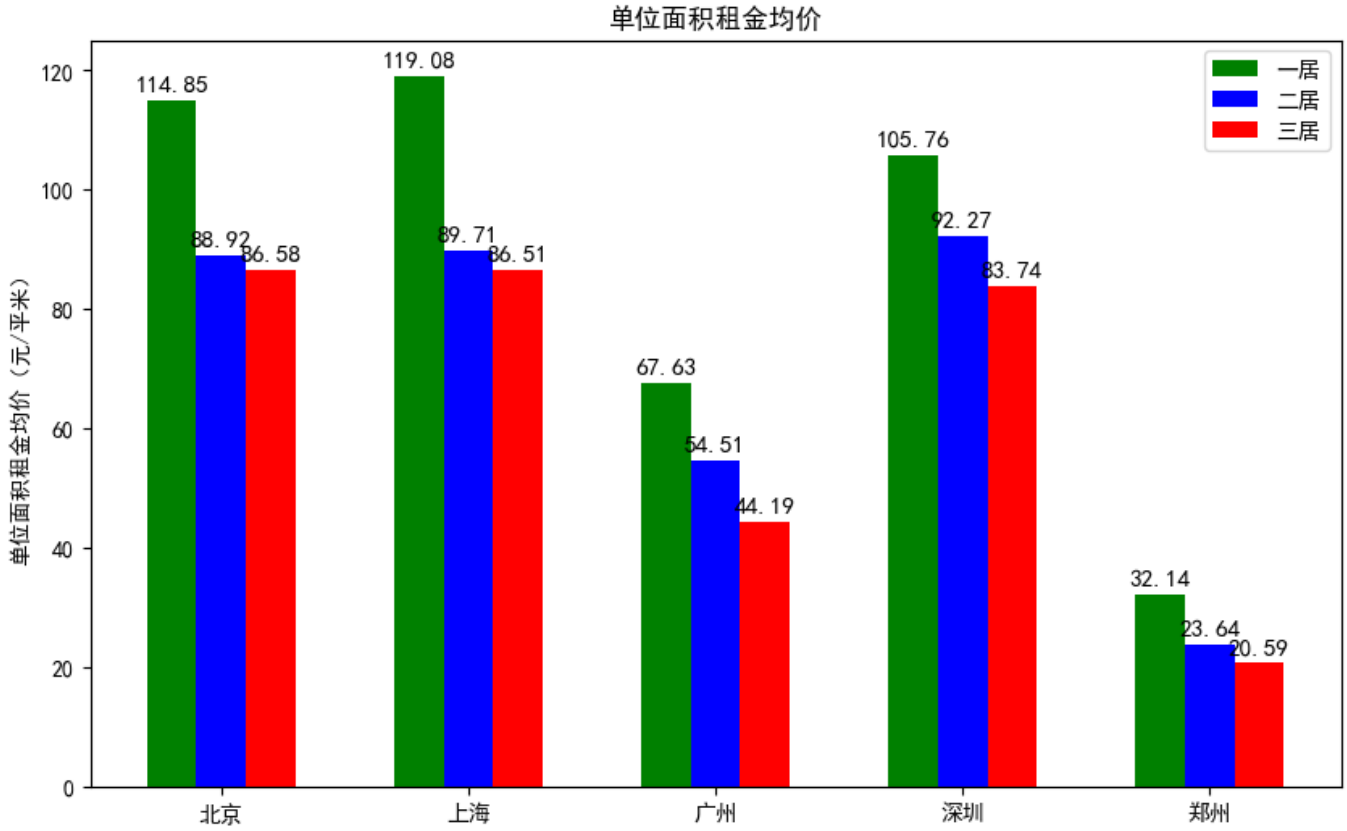
与上面的代码类似，这里仅仅加入一个引用的代码总体和户型

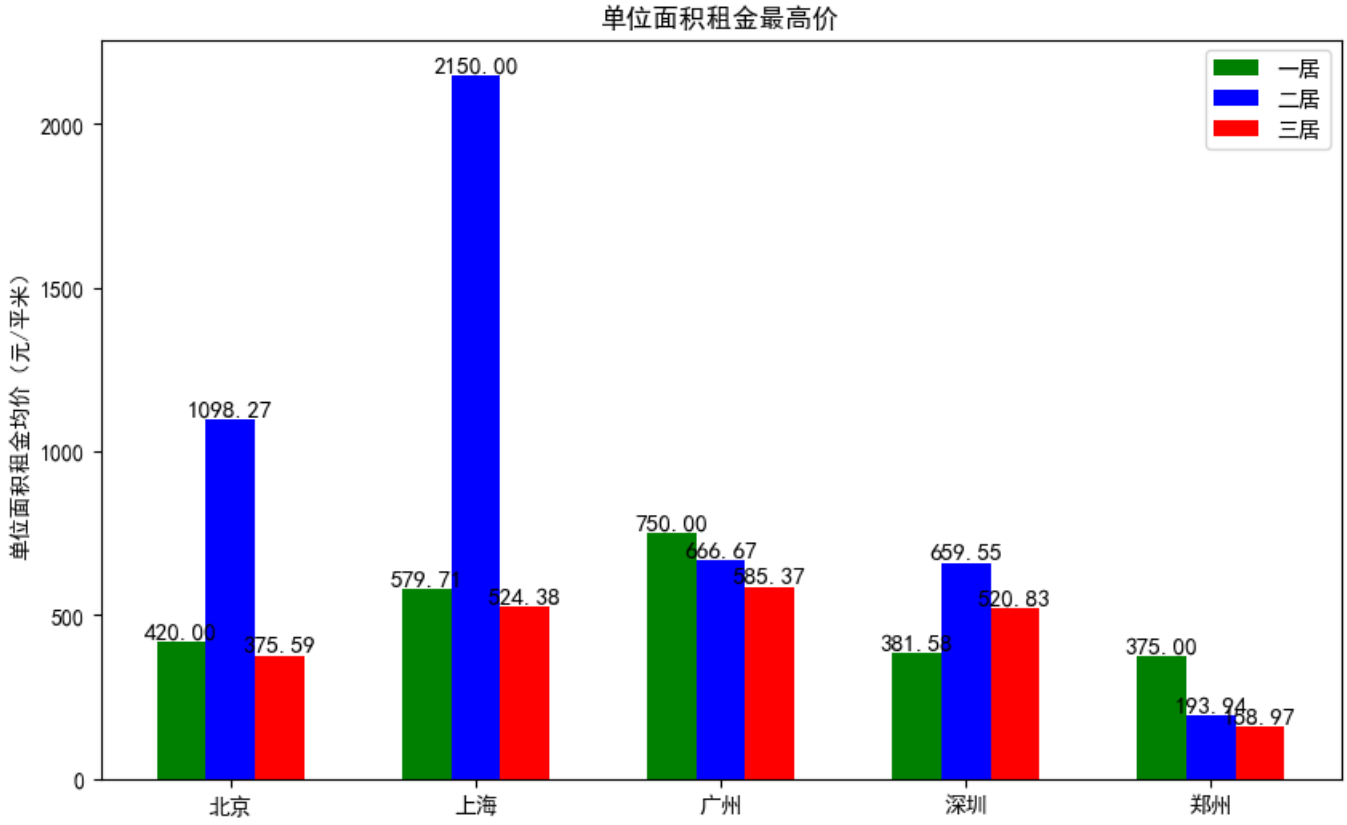
结果展示

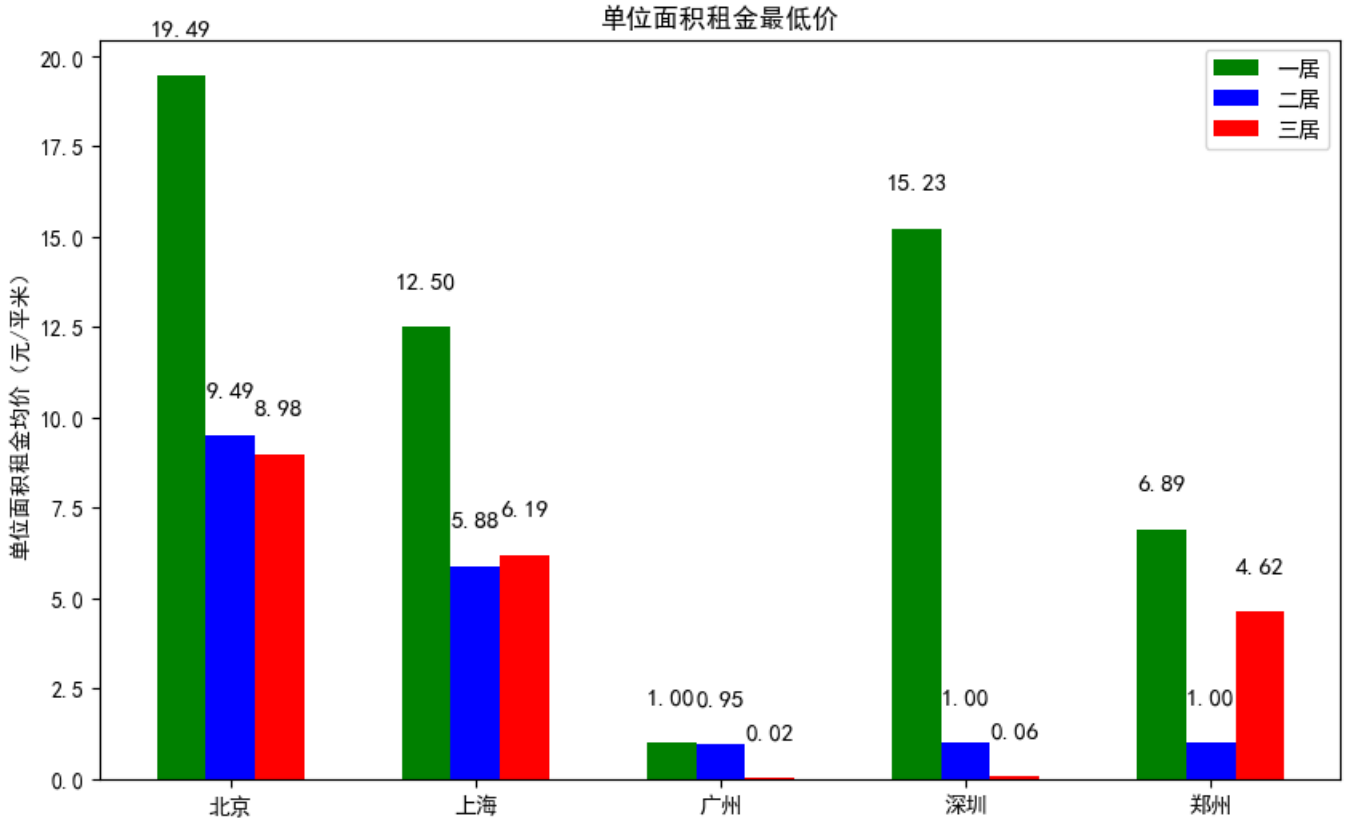


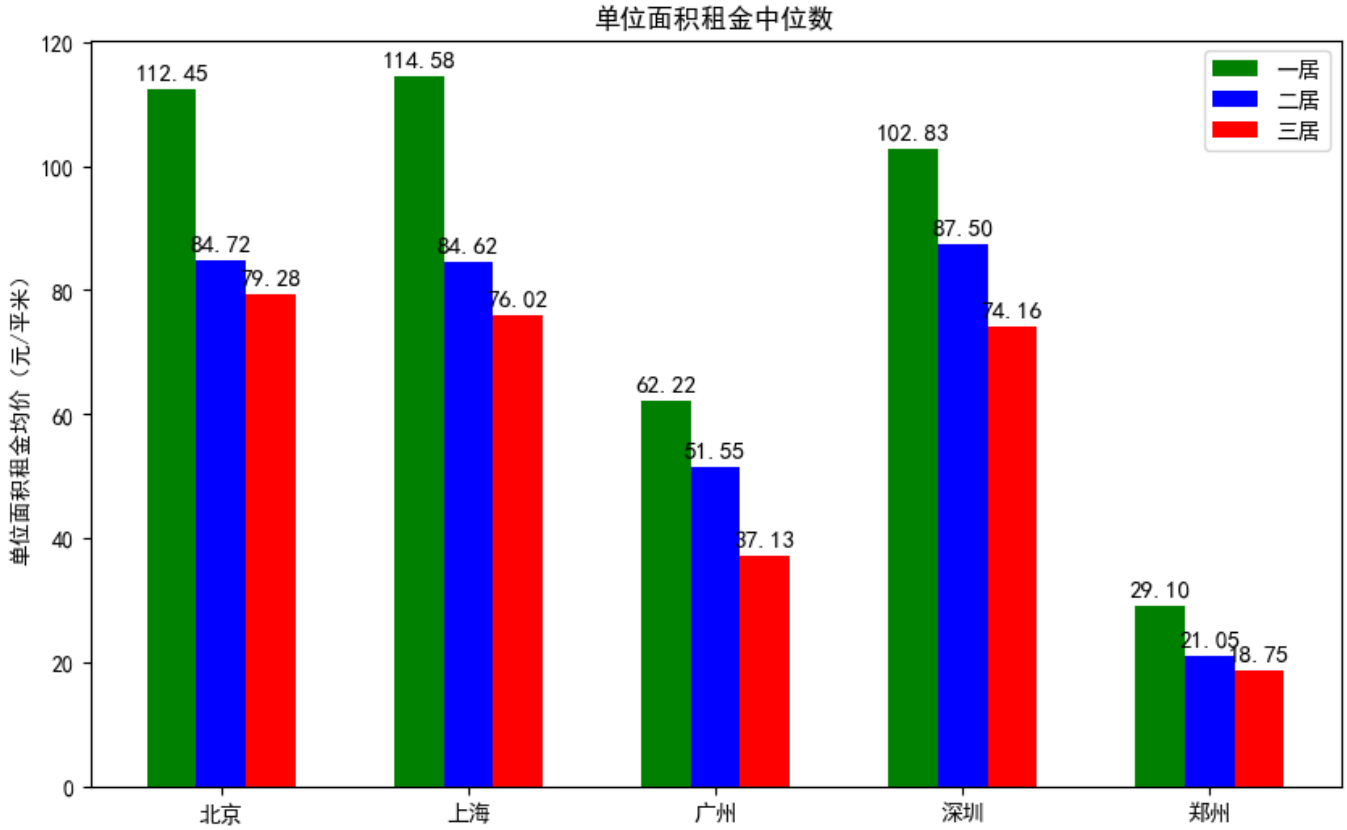



租金最低价
一居
二居
三居
租金均价（元）
0
200
400
600
800
1000
1200
900
500
1100
600
500
700
80
800
600
850
1300
1060
200
501
370
北京
上海
广州
深圳
郑州


单位面积租金均价
一居
二居
三居
单位面积租金均价（元/平米）
0
20
40
60
80
100
120
114.85
88.92
86.58
119.08
89.71
86.51
67.63
54.51
44.19
105.76
92.27
83.74
32.14
23.64
20.59
北京
上海
广州
深圳
郑州


单位面积租金最高价
一居
二居
三居
单位面积租金均价（元/平米）
0
500
1000
1500
2000
420.00
1098.27
375.59
579.71
2150.00
524.38
750.00
666.67
585.37
381.58
659.55
520.83
375.00
193.94
168.97
北京
上海
广州
深圳
郑州


单位面积租金最低价
一居
二居
三居
单位面积租金均价（元/平米）
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
19.49
9.49
8.98
12.50
5.88
6.19
1.00
0.95
0.02
15.23
1.00
0.06
6.89
1.00
4.62
北京
上海
广州
深圳
郑州



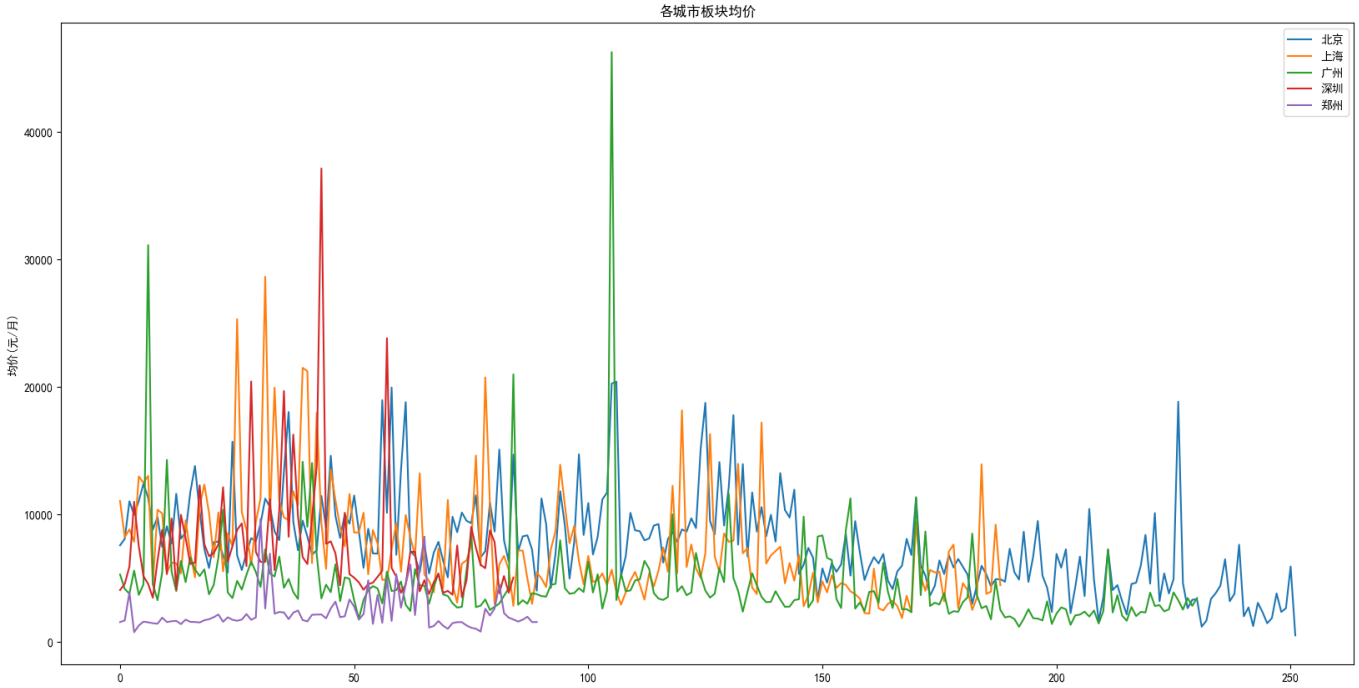
5. 板块均价比较

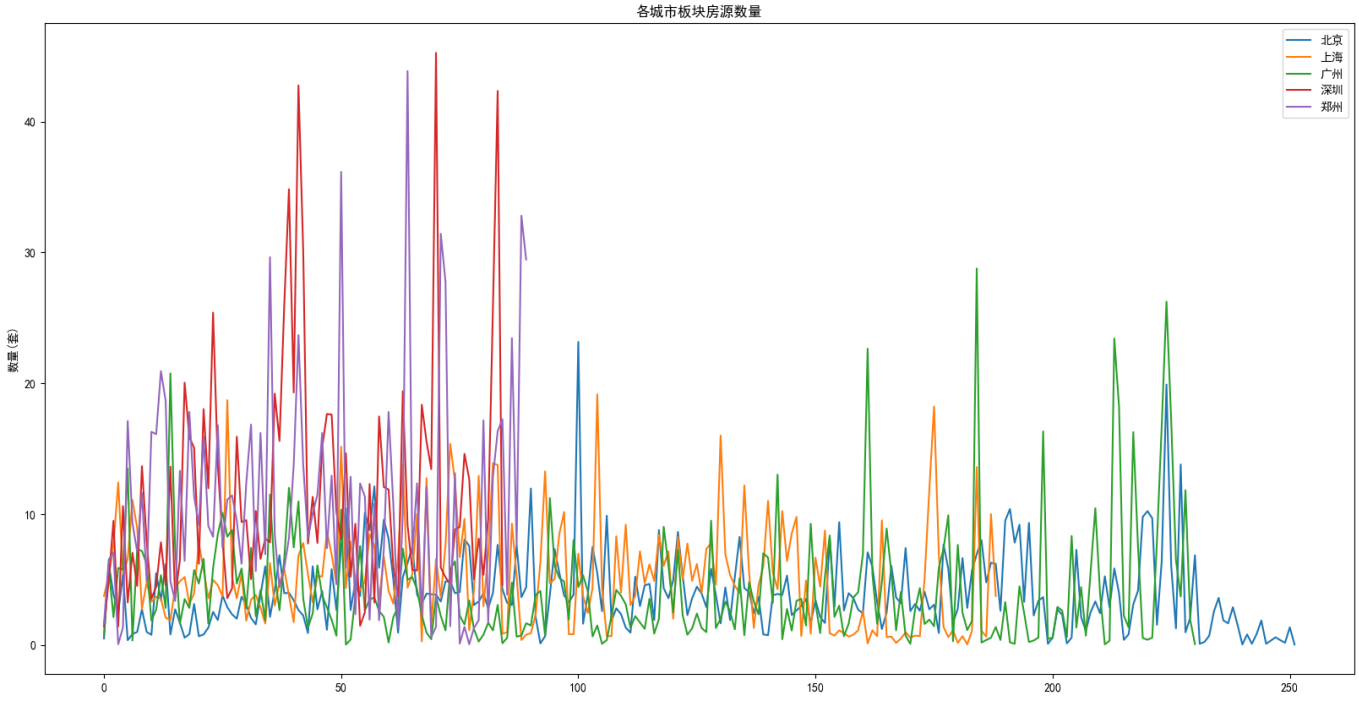
计算和分析每个城市不同板块的均价情况，并采用合适的图或表形式进行展示。例如上图中的"海淀-四季青-五福玲珑居北区"，"四季青"即为板块名称。

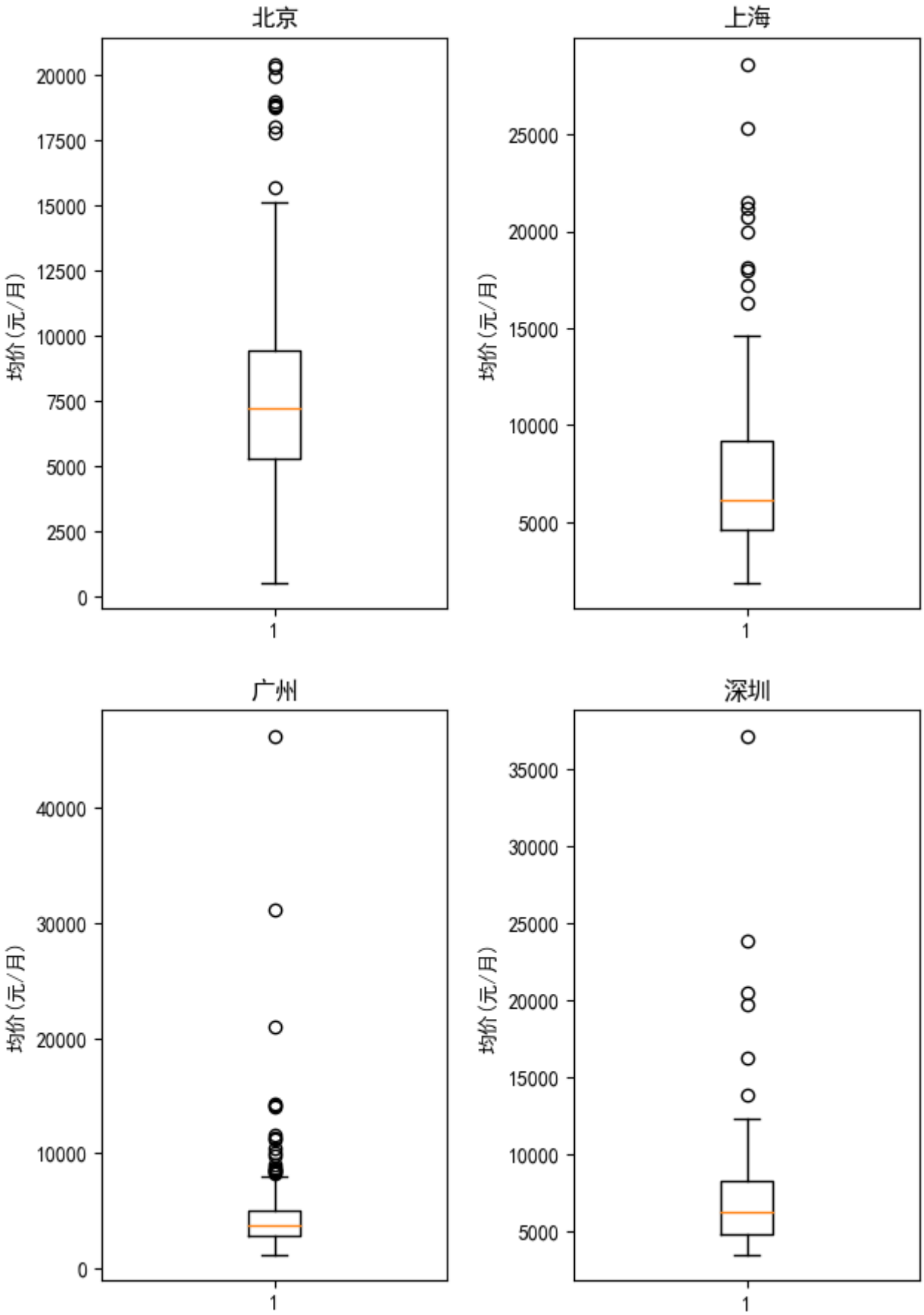
代码编写

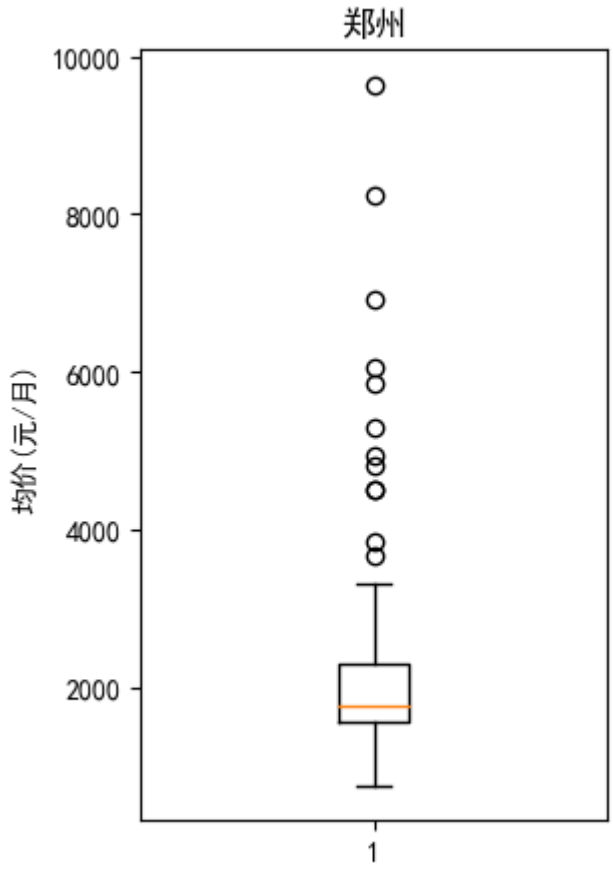
与上面的代码类似,板块均价

结果展示


各城市板块均价
北京
上海
广州
深圳
郑州
均价(元/月)
0
10000
20000
30000
40000
0
50
100
150
200
250


各城市板块房源数量
北京
上海
广州
深圳
郑州
数量(套)
0
10
20
30
40
0
50
100
150
200
250


北京
上海
广州
深圳
均价(元/月)
1



6. 朝向租金分布比较

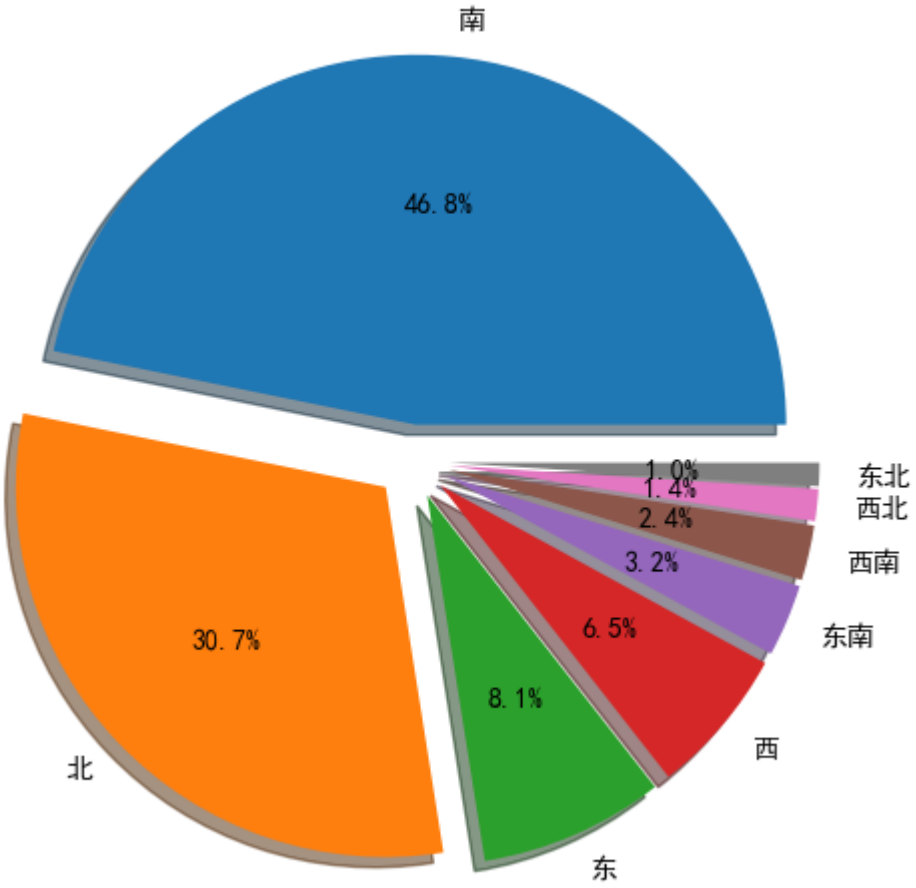
比较各个城市不同朝向的单位面积租金分布情况，采用合适的图或表形式进行展示。哪个方向最高，哪个方向最低？各个城市是否一致？如果不一致，你认为原因是什么？

代码编写

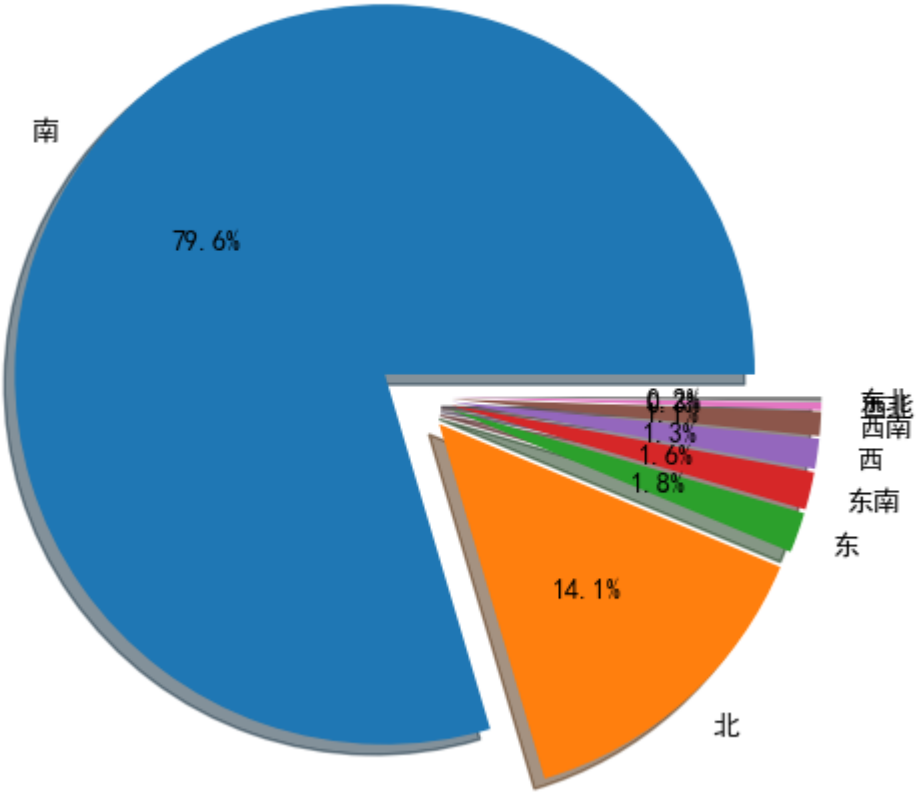
与上面的代码类似,朝向租金分布

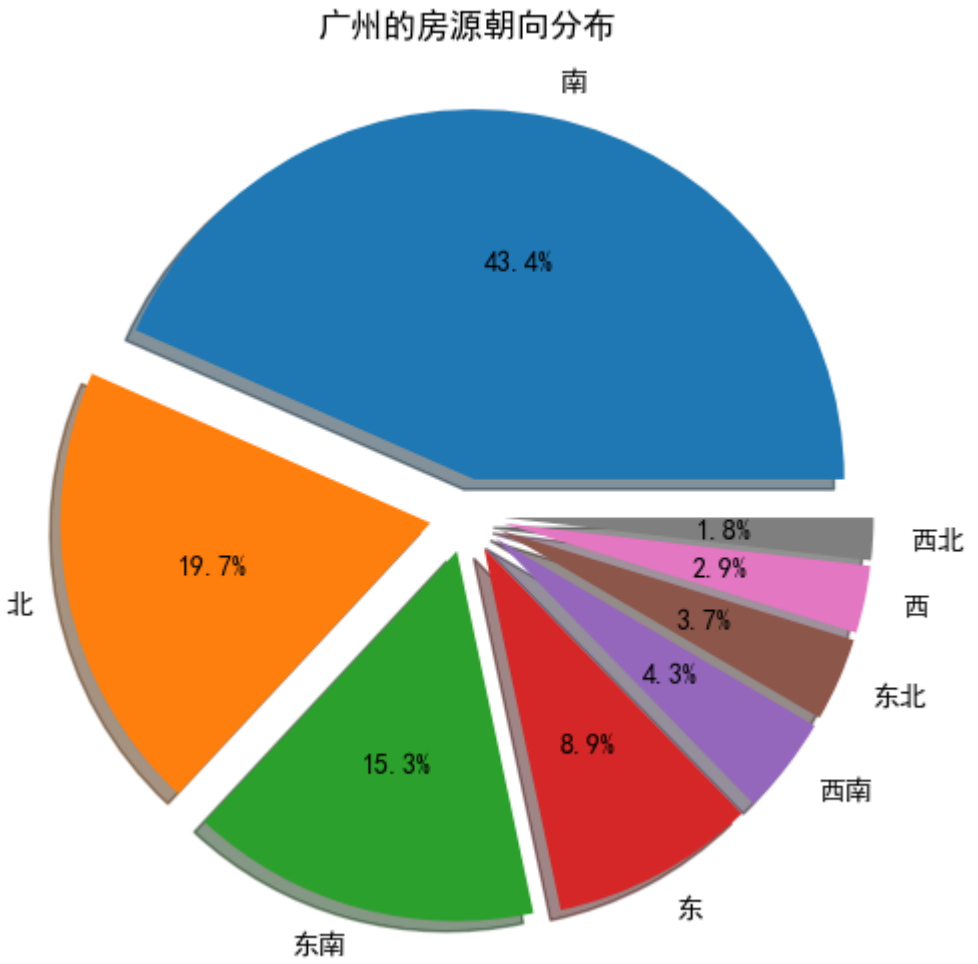
结果展示

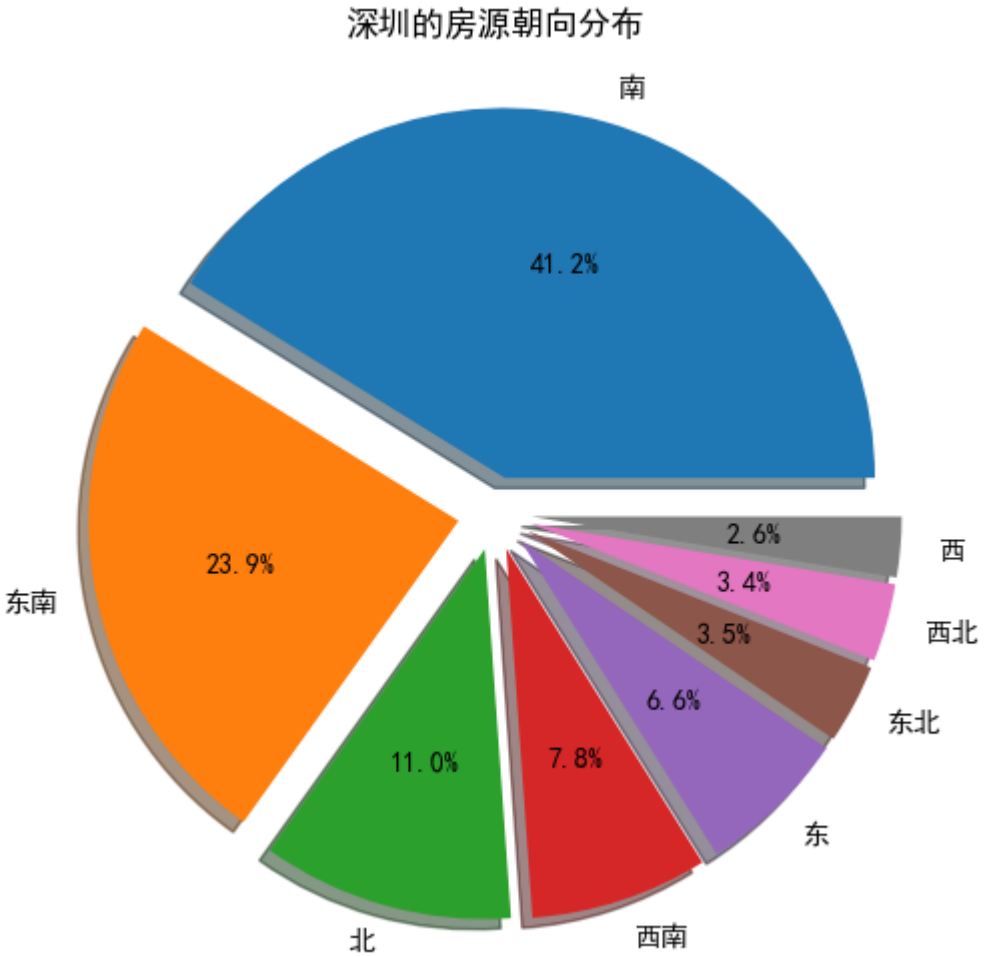
北京的房源朝向分布

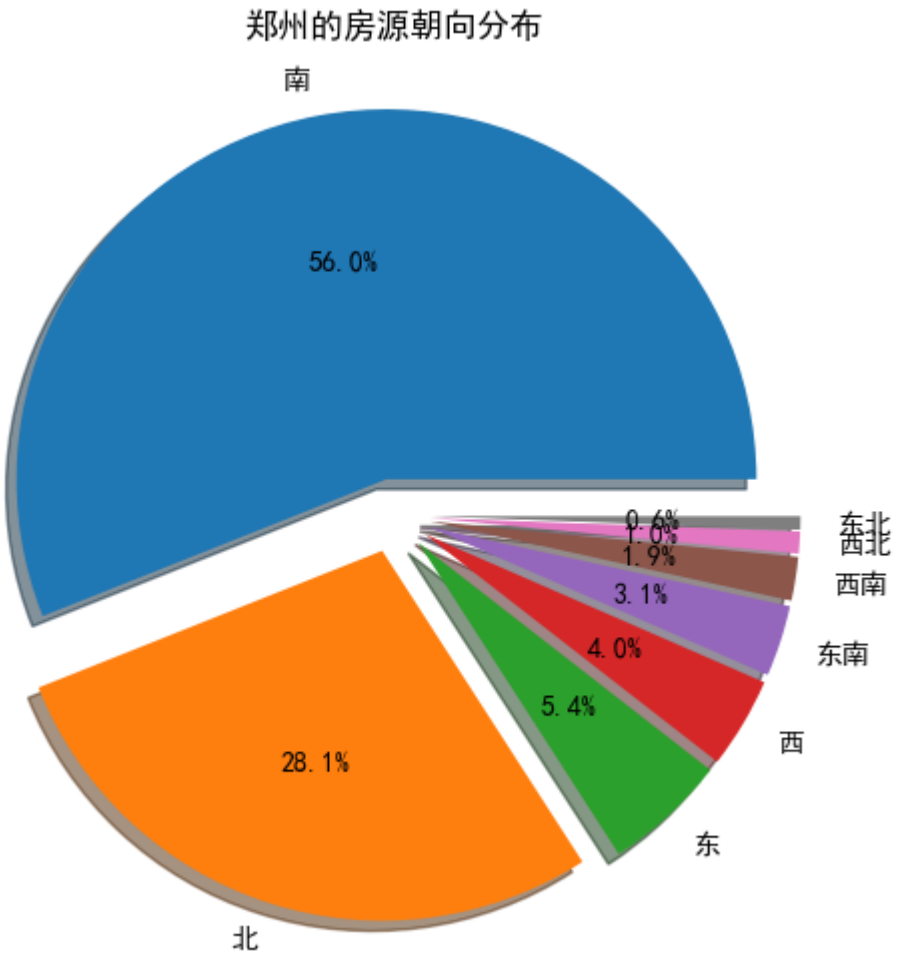


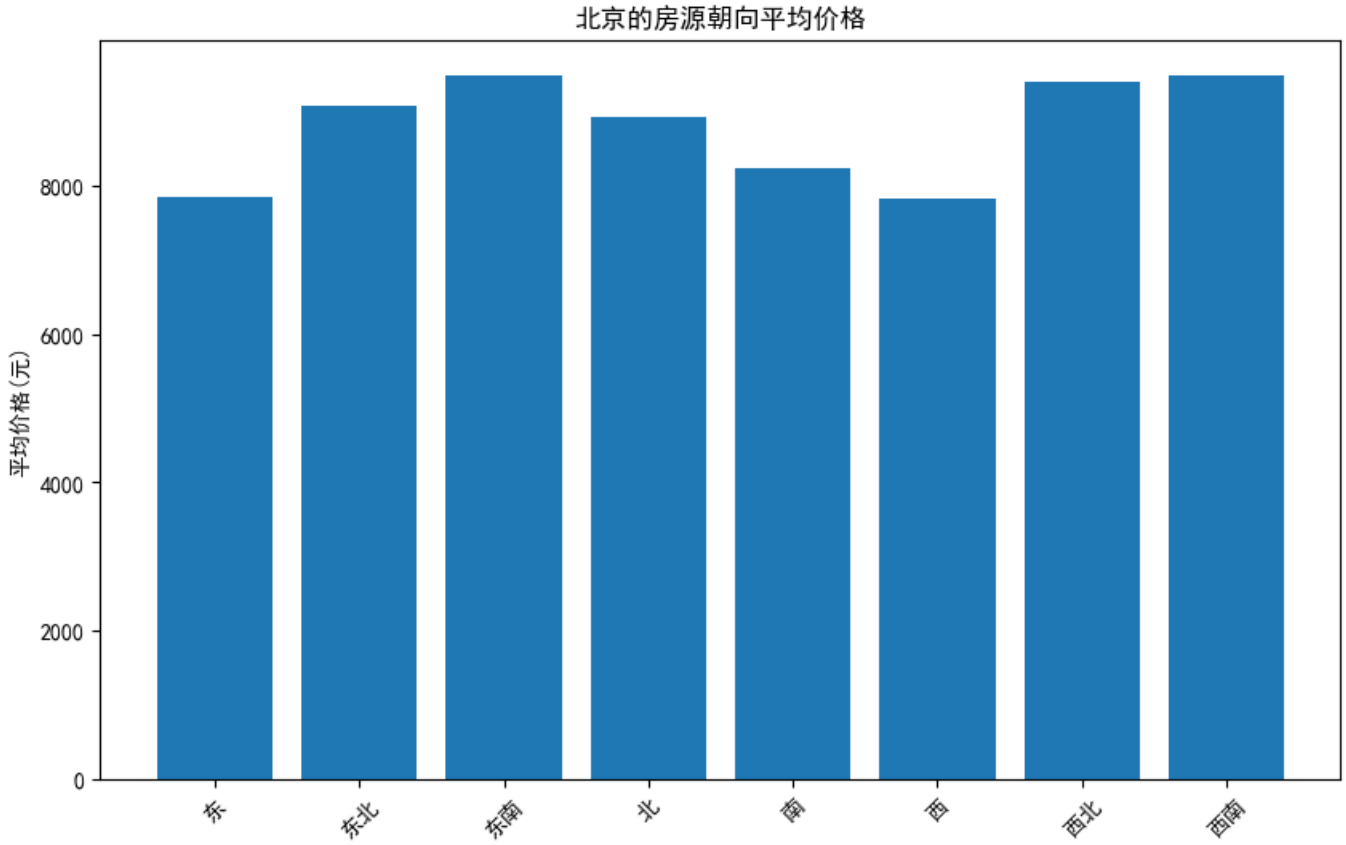
上海的房源朝向分布

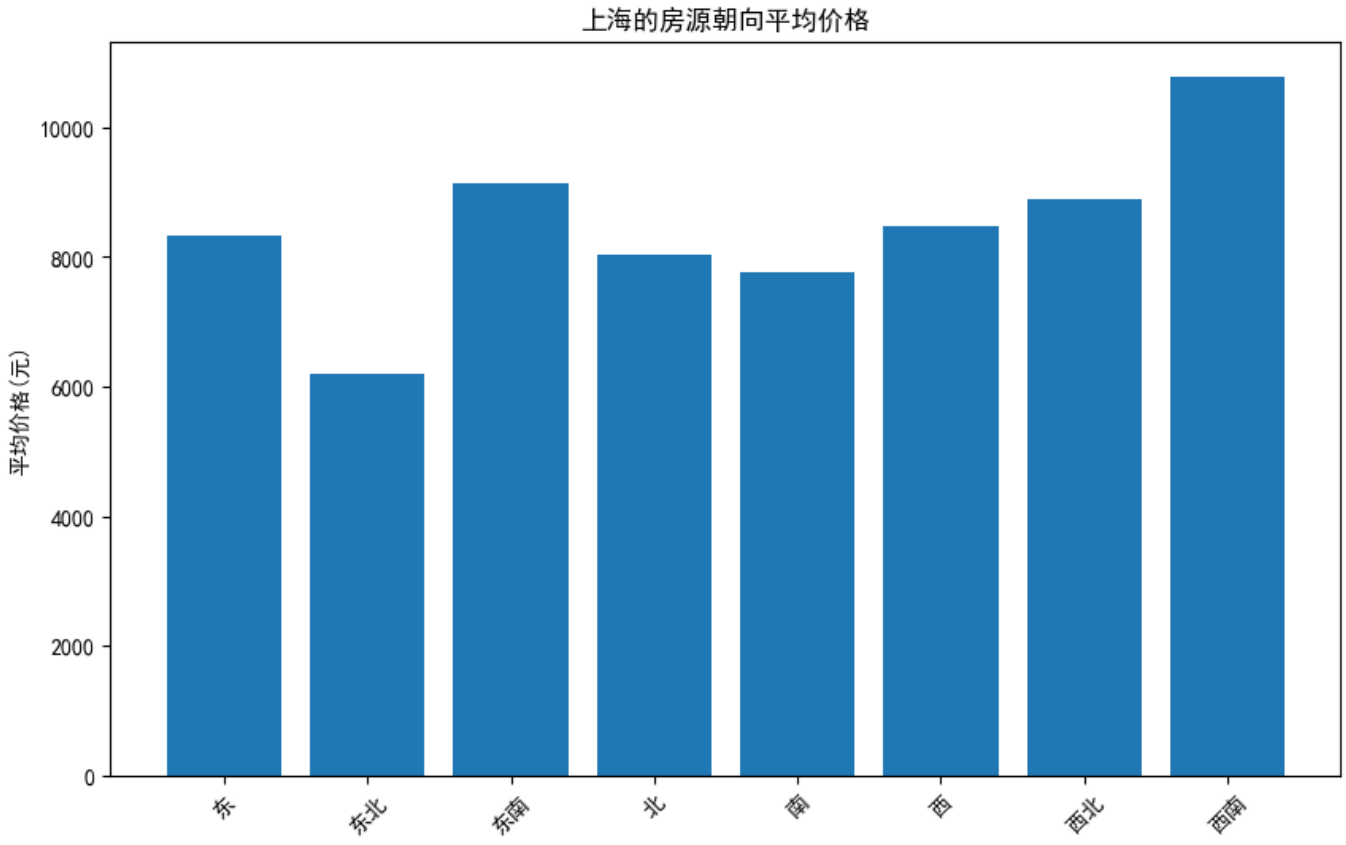


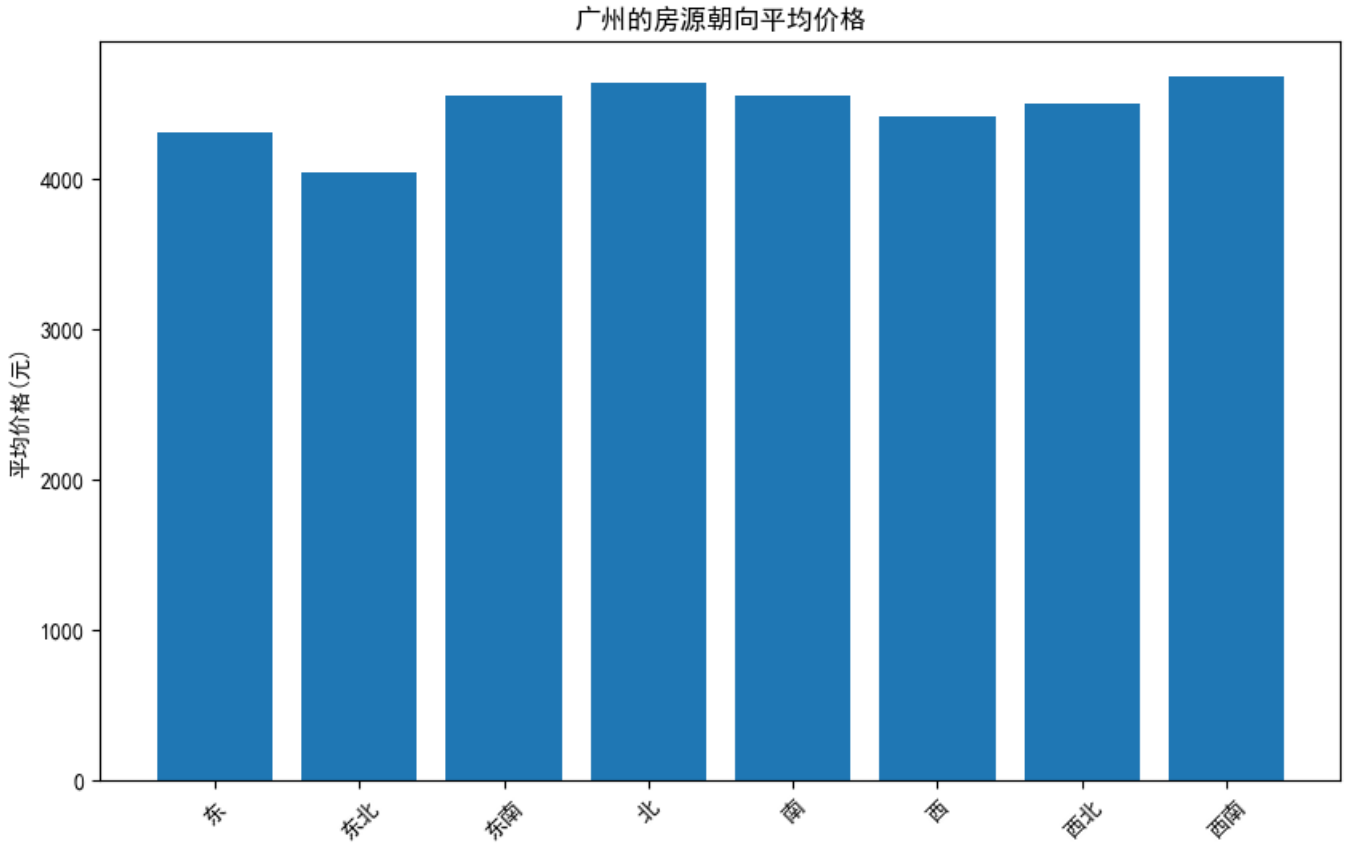

广州的房源朝向分布
南
43.4%
北
19.7%
东南
15.3%
东
8.9%
西南
4.3%
东北
3.7%
西
2.9%
西北
1.8%

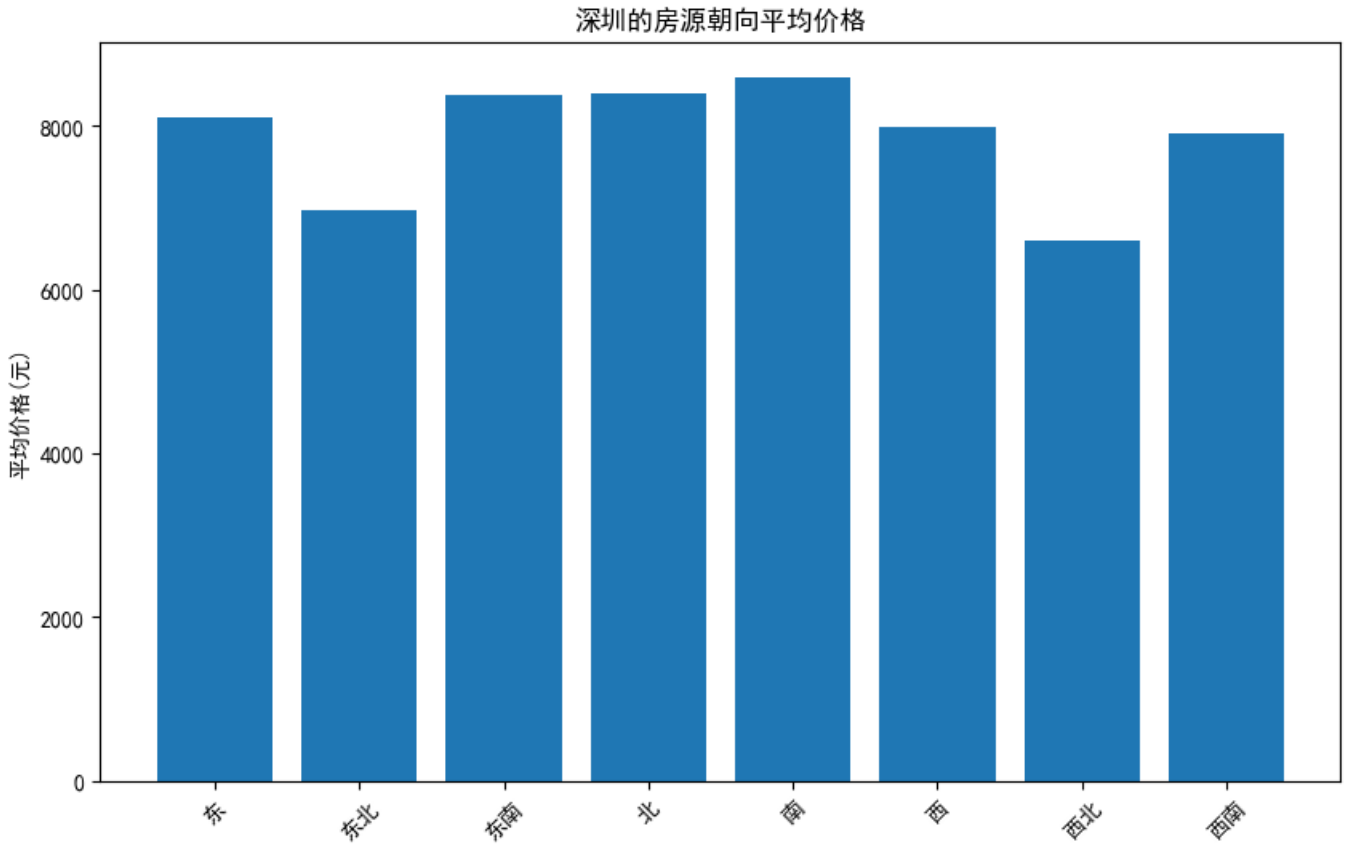

深圳的房源朝向分布
南
41.2%
东南
23.9%
北
11.0%
西南
7.8%
东
6.6%
东北
3.5%
西北
3.4%
西
2.6%

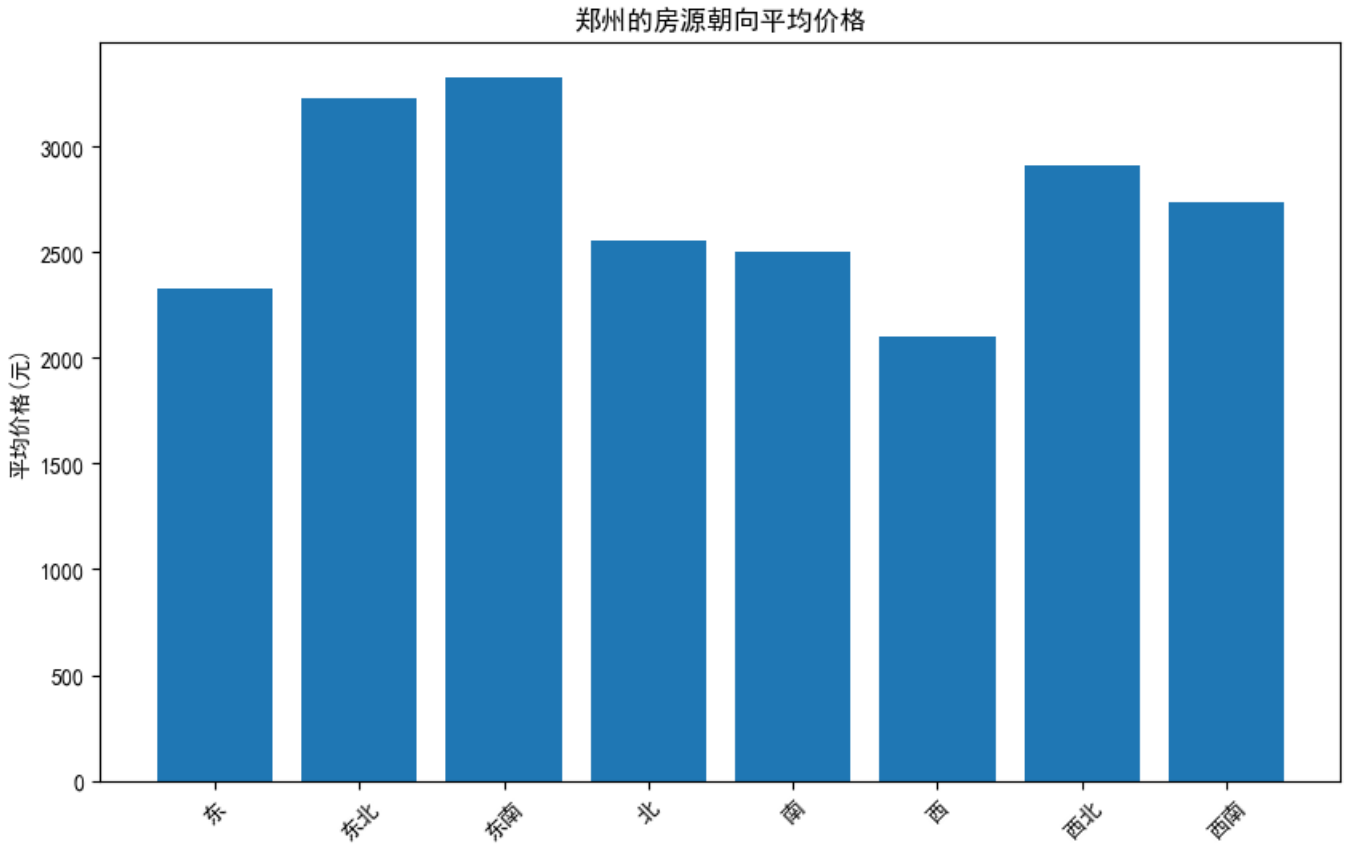

郑州的房源朝向分布
南
56.0%
28.1%
北
5.4%
东
4.0%
西
3.1%
东南
1.9%
西南
1.0%
西北
0.6%
东北


北京的房源朝向平均价格
平均价格(元)
0
2000
4000
6000
8000
东
东北
东南
北
南
西
西北
西南


上海的房源朝向平均价格
平均价格(元)
0
2000
4000
6000
8000
10000
东
东北
东南
北
南
西
西北
西南


广州的房源朝向平均价格
平均价格(元)
0
1000
2000
3000
4000
东
东北
东南
北
南
西
西北
西南





7. 城市平均工资与租金分布关系

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。比较一下在哪个城市租房的负担最重？
根据各地区人力资源和社会保障局所提供的数据

北京,135567,11297 数据来源

8. 与2022年数据对比

与2022年的租房数据进行对比（只比较北上广深4个城市，原始数据会给出），总结你观察到的变化情况，并用图、表、文字等支撑你得到的结论。

代码编写

因为2022年的租房数据格式是json，所以需要一个程序进行转化

```
import json
import csv
import os
file_list_chinese=['北京','上海','广州','深圳']
filename_list=
["BeijingHouseInfo.json","ShanghaiHouseInfo.json","GuangzhouHouseInfo.json","Shenz
henHouseInfo.json"]
file_list=["bj","sh","gz","sz"]
for i in range(4):
    with open(filename_list[i],'r',encoding='utf-8') as f:
        with open(file_list[i]+'.csv','w',encoding='utf-8',newline='') as f1:
            writer=csv.writer(f1)

writer.writerow(['name_chinese','block','house_type','direct','area','price'])
            for line in f.readlines():
                dic=json.loads(line)

writer.writerow([file_list_chinese[i],dic['district'],dic['layout'],dic['direction
'].split(" ")[0],dic['area'],dic['total_price']])
```

10. 提交要求

以pdf格式提交到教学云平台上，文件名为学号，总页数不超过30页。

11. 截止时间

截止时间为2024年1月6日23:59。