python 期末大作业实验报告

数据爬取

本实验爬取北上广深四个一线城市, 以及驻马店一个非一线城市, 爬取字段说明如下:

```
class RentHouseItem(scrapy.Item):
    city = scrapy.Field()  # 市，如 北京
    name = scrapy.Field()  # 房子名称，如 整租·忠实里 1室1厅 东
    district = scrapy.Field()  # 行政区，如 海淀
    county = scrapy.Field()  # 县级板块，如 四季青
    town = scrapy.Field()  # 最小一级，如 五福玲珑居北区
    orientation = scrapy.Field()  # 房屋朝向
    room_type = scrapy.Field()  # 房屋类型，如 1 表示 1居室
    price = scrapy.Field()  # 价格，单位 元/月
    area = scrapy.Field()  # 面积，单位 平方米
    url = scrapy.Field()  # 房子信息链接
```

> 某些房屋可能会本身就不存在一些字段, 比如 "独栋" 型房屋没有地区板块和朝向属性

爬取步骤

因为每个城市数据过多, 链家网站一次只会展示 3000 条 *(30 条/页*100 页)*, 所以我采用 **分价格分行政区** 的方式进行爬取,每个城市按照行政区分别爬取, 各个城市通过设置合理的价格区间, 使得每次爬取的数量都控制在 3000 条以内,
以保证能够爬取到所有数据.

由于要爬取五个城市的数据, 每个城市网页的页面提取一致, 为了代码复用, 所以编写 `Crawler` 类执行实际的爬取. `Crawler`
类中包含有两个函数, 分别展示如下:

`process_url` (对 `url` 进行进一步处理, 比如加上页码)

```
def process_url(self, response: HtmlResponse, **kwargs):
    """
    确定爬取的页数，进一步处理url
    """
    # 判断如果查询记录为0，则直接返回
    total_nums = response.xpath(
        '/html/body/div[@class="wrapper"]/div[1]/div[@id="content"]/div[@class="content__article
        '/p[@class="content__title"]/span[1]/text()').extract_first()
    if total_nums is not None and int(total_nums) == 0:
        yield

    # 获取将要爬取的页数
    total_page = response.xpath(
        '/html/body/div[@class="wrapper"]/div[1]/div[@id="content"]/div[@class="content__article
        '/div[@class="content__pg"]/@data-totalpage').extract_first()
    total_page = int(total_page) if total_page is not None else 100

    url_parts = response.url.split('brp')
    for page in range(1, total_page + 1, 1):
        # 组装url，进行实际的爬取
        url = url_parts[0] + 'pg' + str(page) + 'brp' + url_parts[1]
        yield Request(url=url, callback=self.get_info, dont_filter=True,
                      cb_kwargs={'city': kwargs['city'],
                                 'low_price': kwargs['low_price'], 'high_price': kwargs['high_pr
                                 'prefix': kwargs['prefix']})
```

get_info (对网页内容进行爬取, 真正的爬虫代码)

```
def get_info(self, response: HtmlResponse, **kwargs):
    """
    真正的爬虫函数
    """
    # 再次判断是否是0条记录
    total_nums = response.xpath(
        '/html/body/div[@class="wrapper"]/div[1]/div[@id="content"]/div[@class="content__article
        '/p[@class="content__title"]/span[1]/text()').extract_first()
    if total_nums is not None and int(total_nums) == 0:
        yield

    low_price, high_price = int(kwargs['low_price']), int(kwargs['high_price'])
    prefix = str(kwargs['prefix'])

    list_items = response.xpath(
        '/html/body/div[@class="wrapper"]/div[1]/div[@id="content"]/div[@class="content__article
        '/div[1]/div[@class="content__list--item"]')
    for list_item in list_items:
        rent_house_item = RentHouseItem()
        rent_house_item['city'] = kwargs['city']
        rent_house_item['name'] = list_item.xpath('./a[@class="content__list--item--aside"]/@tit
        rent_house_item['url'] = prefix + list_item.xpath('./a[1]/@href').extract_first().strip(

        list_item = list_item.xpath('./div[@class="content__list--item--main"]')

        # 记录最低价格，在当前价格范围内的进行存储，避免数据重复
        price = list_item.xpath('./span[@class="content__list--item-price"]/em/text()').extract_
            .split('-')[0]
        if not (low_price <= int(price) < high_price):
            continue
        rent_house_item['price'] = price

        # 获取面积，户型，朝向，位置信息
        desc = list_item.xpath('string(./p[@class="content__list--item--des"])').extract_first()
        for d in desc:
            if d.count('㎡') != 0:
                rent_house_item['area'] = d[:-1] if d.count('-') == 0 else d.split('-')[0]
            elif d.count('室') != 0 or d.count('房') != 0:
                rent_house_item['room_type'] = d[:(
                    int(d.index('室')) if d.count('室') != 0 else int(d.index('房')))]
            elif d.count('-') == 2:
                region = d.split('-')
                rent_house_item['district'] = region[0]
```

```
                rent_house_item['county'] = region[1]
                rent_house_item['town'] = region[2]
            elif d.count('东') != 0 or d.count('南') != 0 or d.count('西') != 0 or d.count('北')
                rent_house_item['orientation'] = "".join(d.strip().replace(" ", "").split("/"))

        yield rent_house_item
```

> 部分数据处理的说明:
>
> 1. 带有区间的面积和价格均取最小值;
> 2. 显示为 `x室/房` 统一处理为 `x居室` .

一个调用例子如下 (以 北京 为例, 其他城市类似):

```
class BeiJingRentSpider(scrapy.Spider):
    """
    北京租房数据
    """
    name = "beijing_rent"
    allowed_domains = ["bj.lianjia.com"]
    start_urls = ["https://bj.lianjia.com/zufang/"]
    # 自定义管道
    custom_settings = {
        'ITEM_PIPELINES': {"lianjia.pipelines.BeijingRentHouseItemPipeline": 200, }
    }

    def start_requests(self):
        crawler = Crawler()
        # 列出所有行政区，为了后续构建爬虫的url
        districts = ["dongcheng", "xicheng", "chaoyang", "haidian", "fengtai", "shijingshan", "t
                     "daxing", "yizhuangkaifaqu", "shunyi", "fangshan", "mentougou", "pinggu", '
                     "yanqing"]
        step_price = 100
        for district in districts:
            # 价格0到三万区间内，每500元递增，保证每次爬取页数在100页以内
            for price in range(0, 3_0000, step_price):
                yield Request(url=f'https://bj.lianjia.com/zufang/{district}/brp{price}erp{price
                              callback=crawler.process_url, dont_filter=True,
                              cb_kwargs={'city': "北京", 'low_price': price, 'high_price': price
                                         'prefix': "https://bj.lianjia.com"})
            # 三万以上房子数量较少可以直接爬取
            yield Request(url=f'https://bj.lianjia.com/zufang/{district}/brp30000/',
                          callback=crawler.process_url, dont_filter=True,
                          cb_kwargs={'city': "北京", 'low_price': 30000, 'high_price': 1000000,
                                     'prefix': "https://bj.lianjia.com"})
```

最后每个城市爬取完成后, 通过各自的管道将数据保存到 `csv` 文件中, 方便后续分析.

爬取结果

数据爬取结果分别在
`bj_rent_house.csv` 、 `sh_rent_house.csv` 、 `gz_rent_house.csv` 、 `sz_rent_house.csv`
和 `zmd_rent_house.csv` , 数据爬取结果在附件的 `datasets` 目录中, 其中 `source` 目录下是爬取到的数据, `unique`
目录下是去重后的数据，`all.csv` 是所有数据去重整合并处理过的结果，`process_data.py` 为处理数据的

程序, 处理后共计数据有
187977 条, 处理方式如下:

- 删除所有房屋面积大于 1000 平且房屋价格小于 10000 元的数据, 因为经过分析, 这些房屋的面积可能有误 (
  比如 `广州,整租·创基金沙江畔 1室1厅 东/南/西/北` 这条房屋信息的面积为 `3072.00` 平, 这显然不太可能)
- 删除所有房屋面积或租金小于等于 0 的数据
- 删除所有房型(`room_type`)为 `未知` 的数据

以 `url` 为标准去重前后数据条数如下:

城市	去重前数据量	去重后数据量
北京	55862	48048
上海	47144	37941
广州	87861	34511
深圳	48780	64252
驻马店	4751	3333

> 某些房屋, 比如 "独栋" 型房屋只有一套, 但是在网页中会多次展示, 即使手动请求也会显示多套 (但他们的链接相同), 所以去重后会少一些数据

比较五个城市的房租情况

1. 比较总体租金情况
   - 先对数据按照城市进行分组

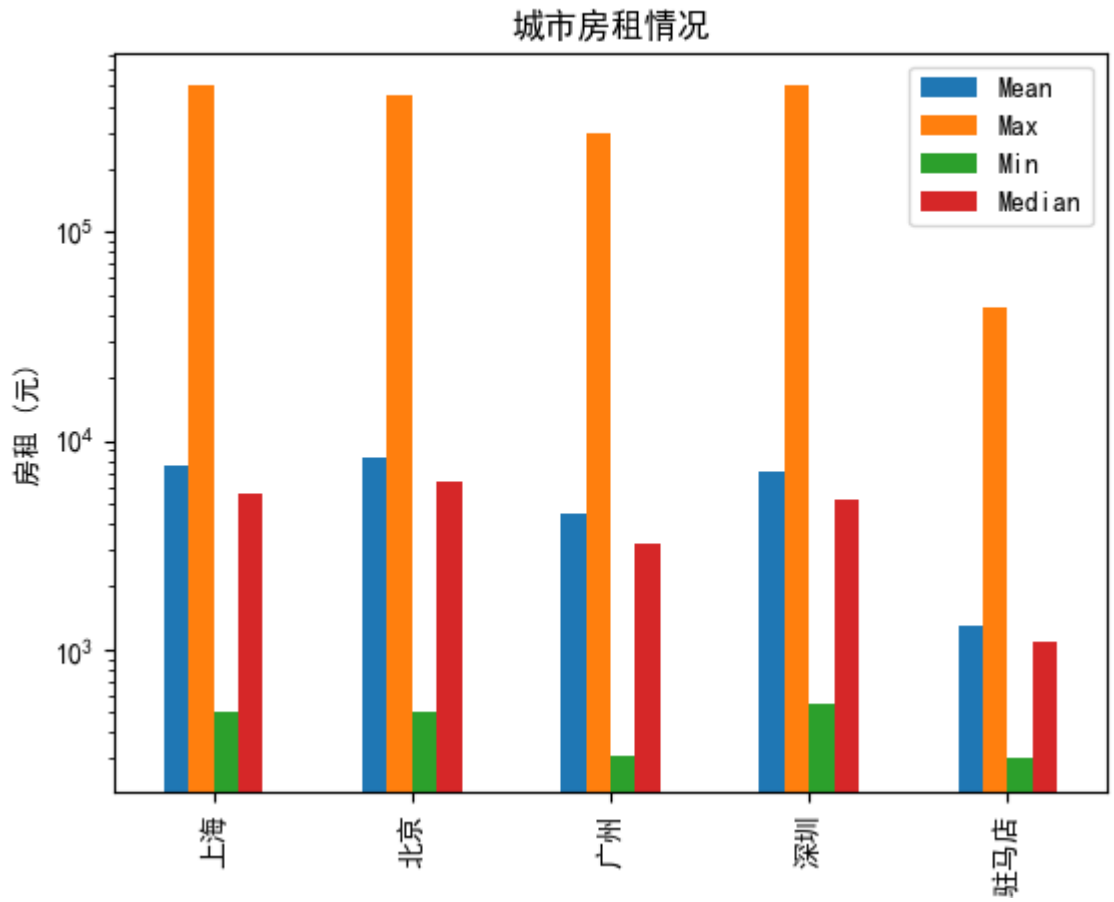
     ```
     all_data = pd.read_csv('../datasets/all.csv')
     city_data = all_data.groupby('city')
     ```

   - 然后每个城市计算平均租金, 最大租金, 最小租金和租金中位数

```
tot_price = pd.DataFrame()
tot_price['City'] = city_data['city'].unique().apply(lambda x: x[0])
tot_price['Mean'] = city_data['price'].mean()
tot_price['Max'] = city_data['price'].max()
tot_price['Min'] = city_data['price'].min()
tot_price['Median'] = city_data['price'].median()
```

- 将上述数据绘制成柱状图, 纵坐标使用对数坐标, 防止数据之间差距过大导致柱状图不明显

```
tot_price.plot(kind='bar', x='City', y=['Mean', 'Max', 'Min', 'Median'])

# 使用对数坐标，避免数字太大无法显示
plt.yscale('log')
plt.xlabel('城市')
plt.ylabel('房租（元）')
plt.title('城市房租情况')
plt.legend()
plt.savefig('imgs/城市租金情况.png')
```



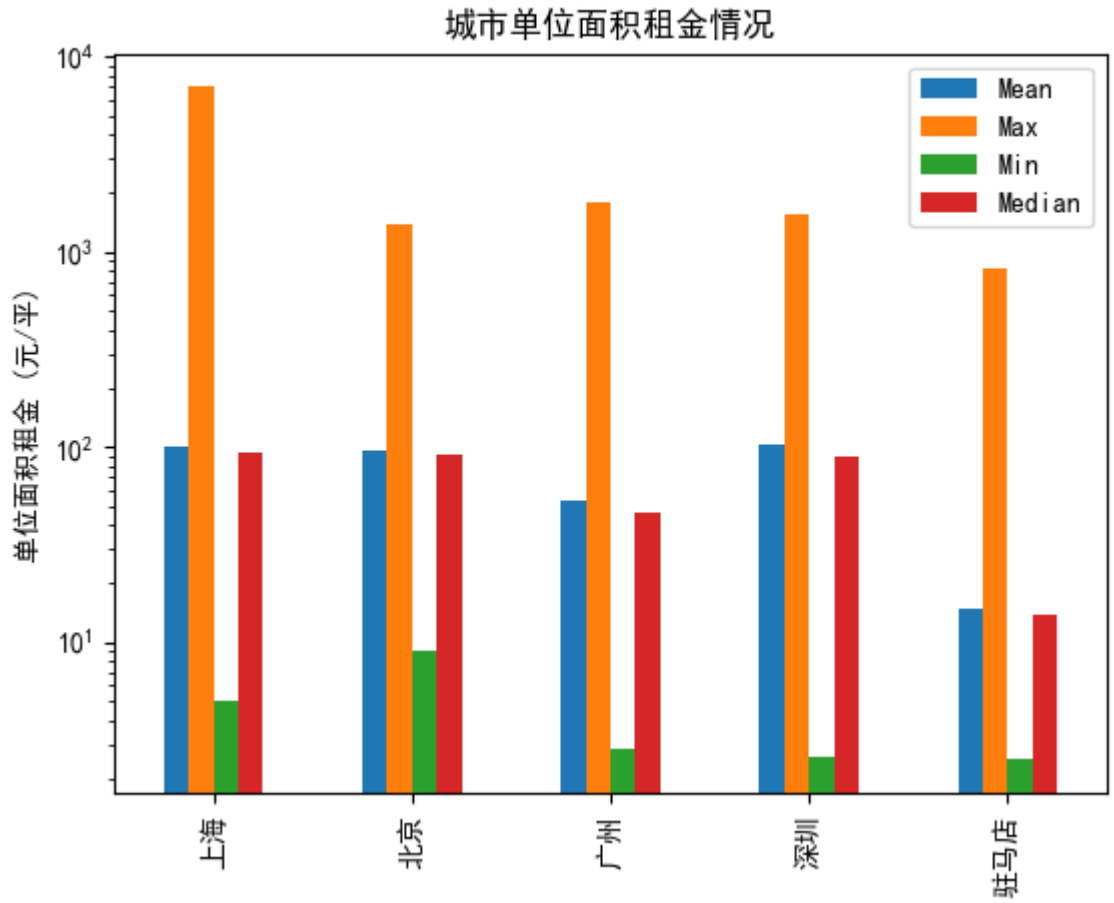
可以看到, 北京、上海和深圳三个城市的各个参数均高于其他城市, 驻马店各个参数均最低.

2. 比较单位面积租金情况

- 先对数据按照城市进行分组并计算单位面积租金的平均租金, 最大租金, 最小租金和租金中位数

```
unit_price = all_data.groupby('city')[['price', 'area']].apply(lambda x: pd.Series({
    'Mean': (x['price'] / x['area']).mean(),
    'Max': (x['price'] / x['area']).max(),
    'Min': (x['price'] / x['area']).min(),
    'Median': (x['price'] / x['area']).median()
})).reset_index()
```

- 绘制柱状图, 同样使用对数坐标防止数据之间差距过大无法显示



可以看到, 北京、上海和深圳三个城市的各个参数均高于其他城市, 其中上海的单位面积租金的最大值最大, 驻马店各个参数均最低.

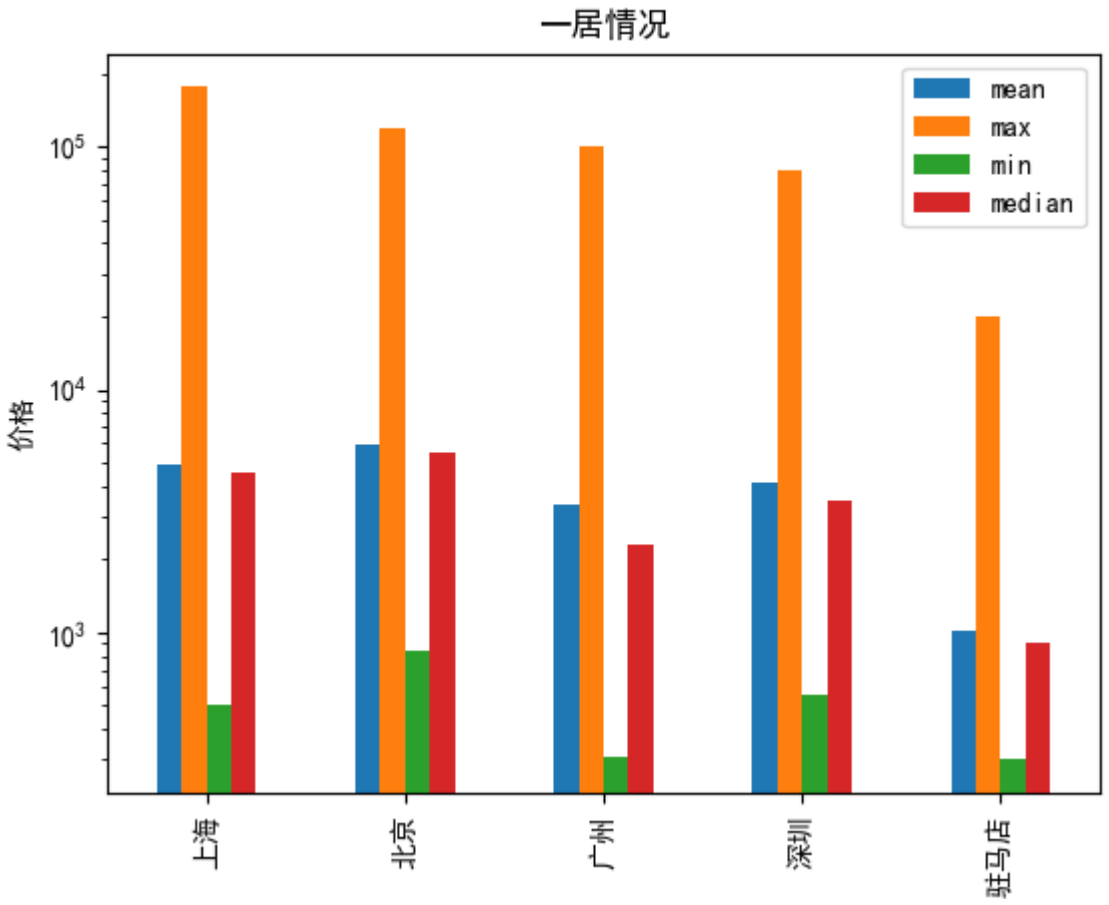
比较五个城市各居室的情况

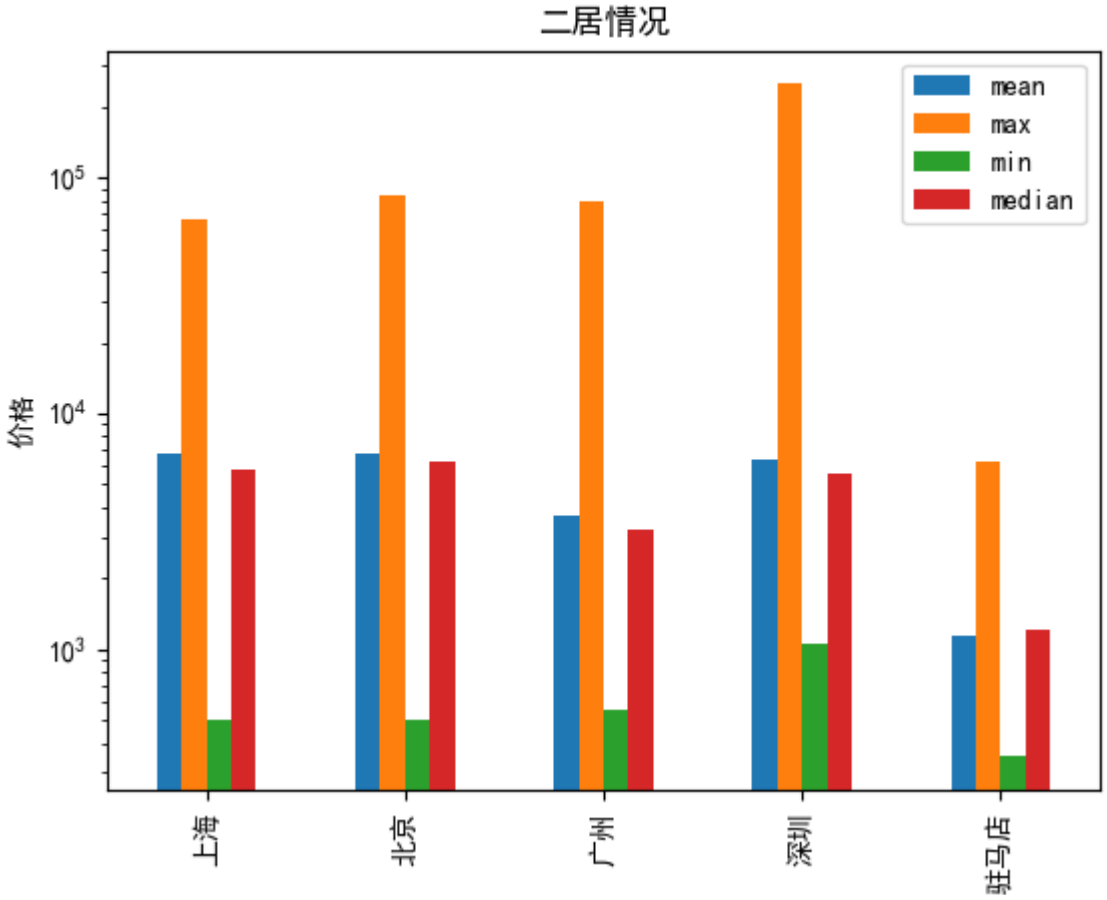
- 先对数据按城市和居室分类, 计算各居室平均租金, 最大租金, 最小租金和租金中位数, 挑选出一居室、二居室和三居室的数据进行绘图

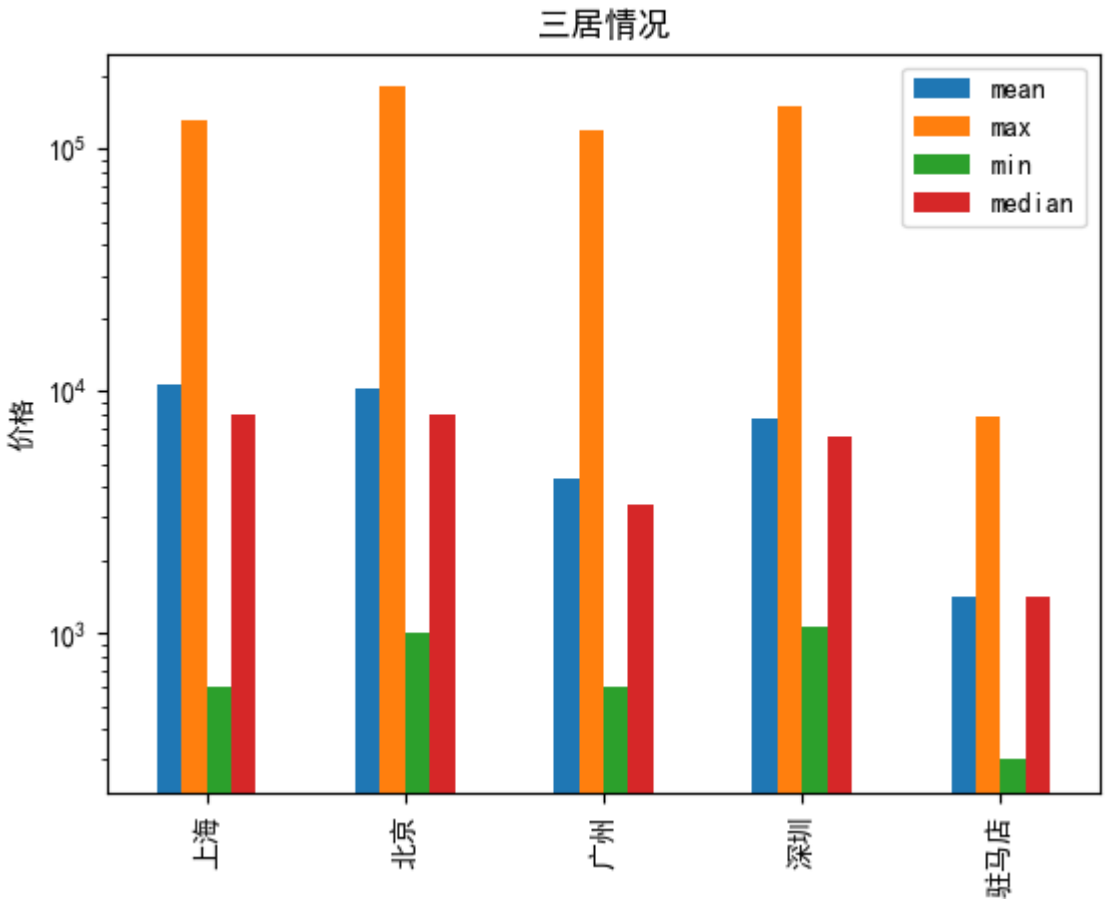
```
statistics = data.groupby(['city', 'room_type']).agg({
    'price': ['mean', 'max', 'min', 'median']
}).reset_index()

filtered_data = statistics[statistics['room_type'].isin(['1', '2', '3'])]
```

- 对每个居室绘制柱状图

```
room_type_1 = filtered_data[filtered_data['room_type'] == '1']
room_type_1.plot(kind='bar', x='city', y='price', legend=True,
                 title='一居情况', xlabel='城市', ylabel='价格')
plt.yscale('log')
plt.savefig('imgs/一居情况.png')
```




二居情况
mean
max
min
median
价格
10^5
10^4
10^3
上海
北京
广州
深圳
驻马店


三居情况
mean
max
min
median
价格
10^5
10^4
10^3
上海
北京
广州
深圳
驻马店

可以看到, 北京、上海和深圳的各居室的平均租金和中位数租金均高于其他城市, 驻马店的各个参数均低于其他城市.

计算分析每个城市各板块均价

因为各个城市板块数量过多, 难以画图, 所以使用导出成 `csv` 文件 *(之后会对各行政区进行分析)*

1. 整体租金均价
   - 读入数据, 并按照 `town` (也就是板块) 字段分组计算平均租金, 并输出到 `csv` 文件中

```
tot_avg = data.groupby(['city', 'district', 'town'])['price'].mean().round(2).reset_index()
tot_avg.to_csv('各板块平均价格.csv', index=False)
```

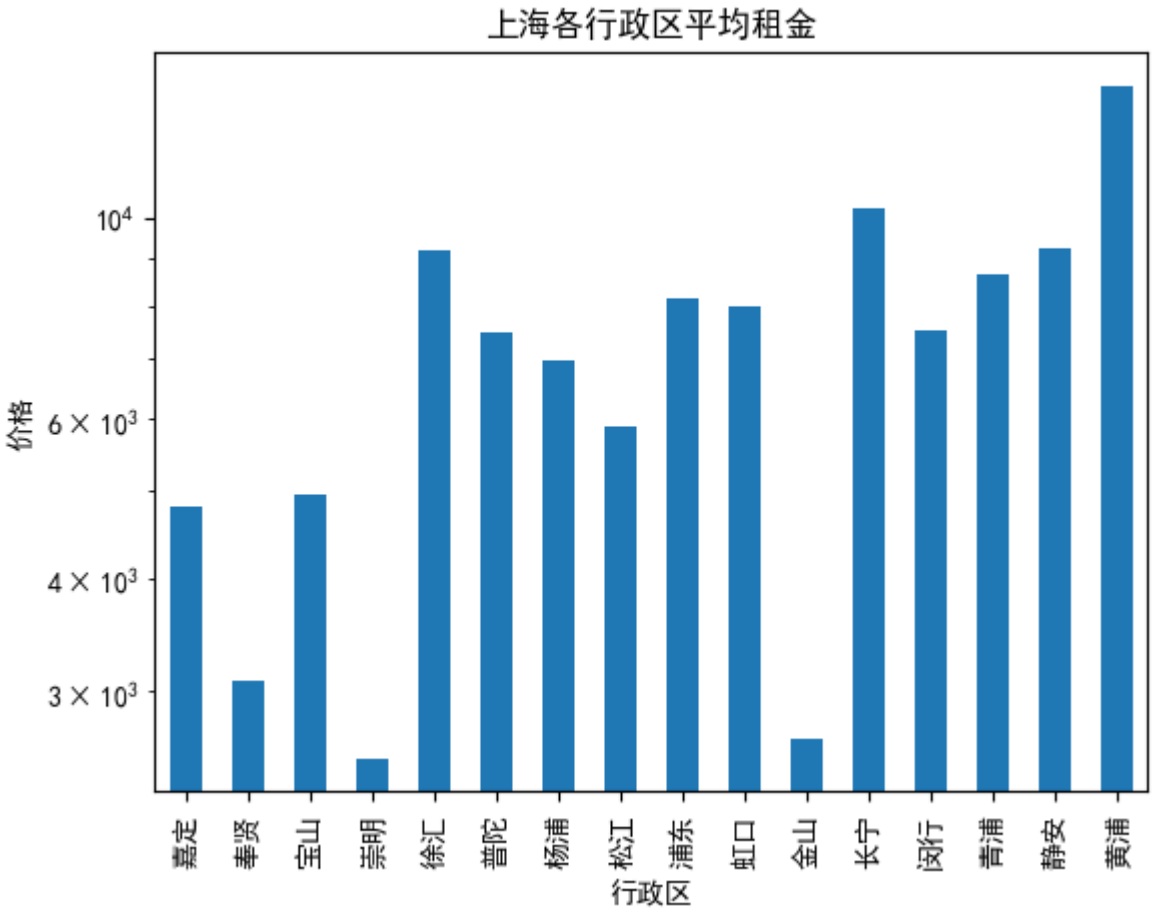
2. 单位面积租金均价
   - 按照 `town` 字段分组计算平均单位面积租金, 并输出到 `csv` 文件中

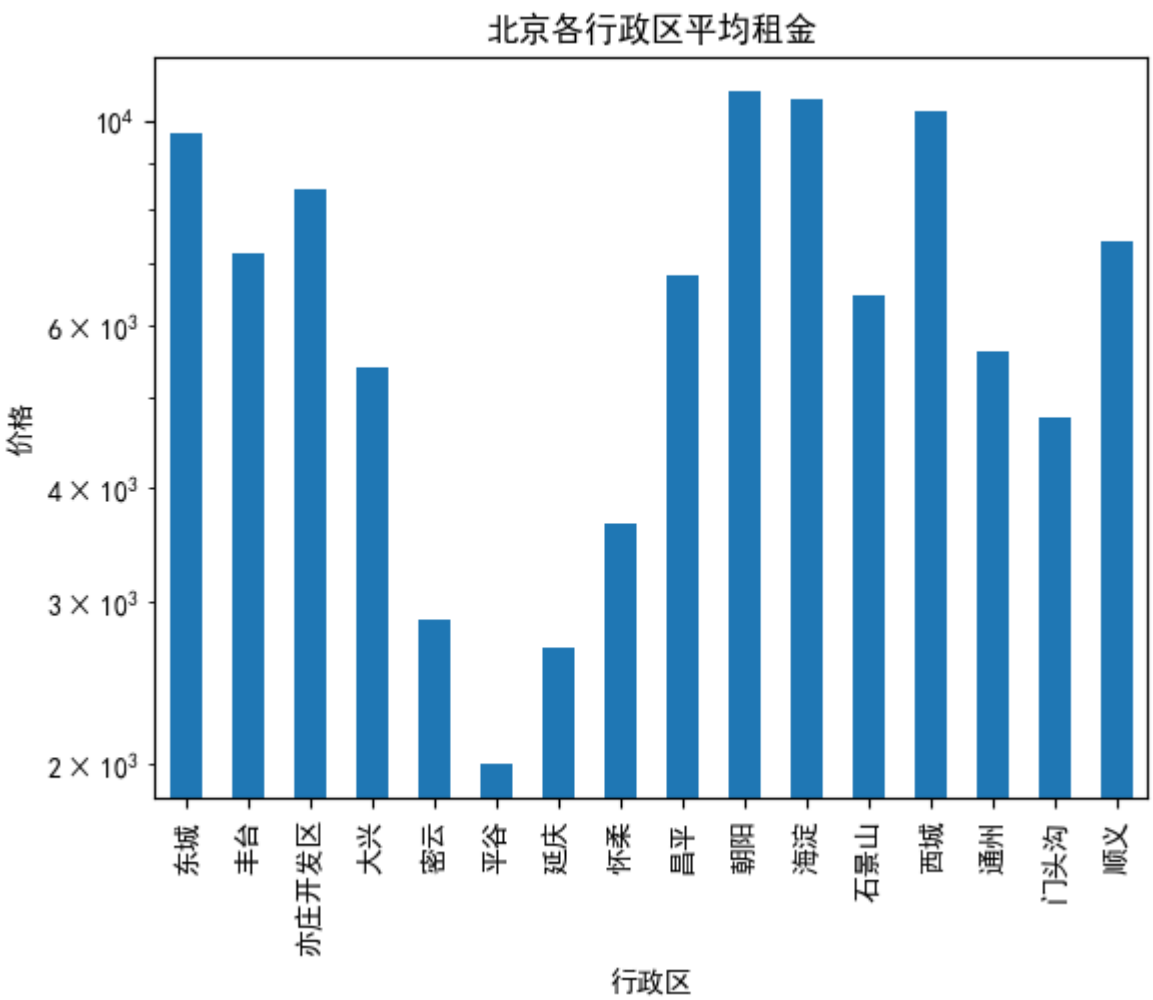
```
unit_avg = data.groupby(['city', 'district', 'town'])[['price', 'area']].apply(lambda x: pd
    'mean': (x['price'] / x['area']).mean().round(2),
})).reset_index()
unit_avg.to_csv('各板块单位面积平均租金.csv', index=False)
```

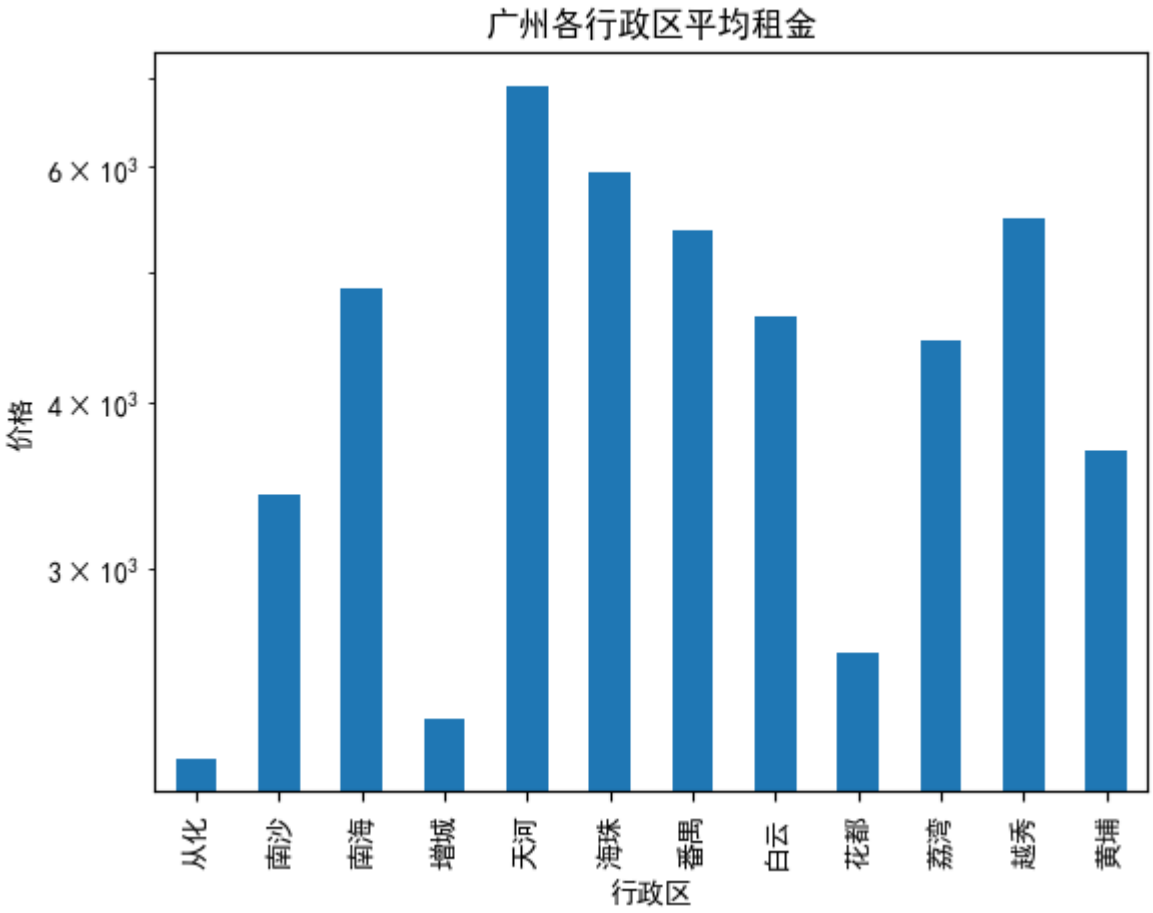
下面是对每个城市各行政区的价格分析

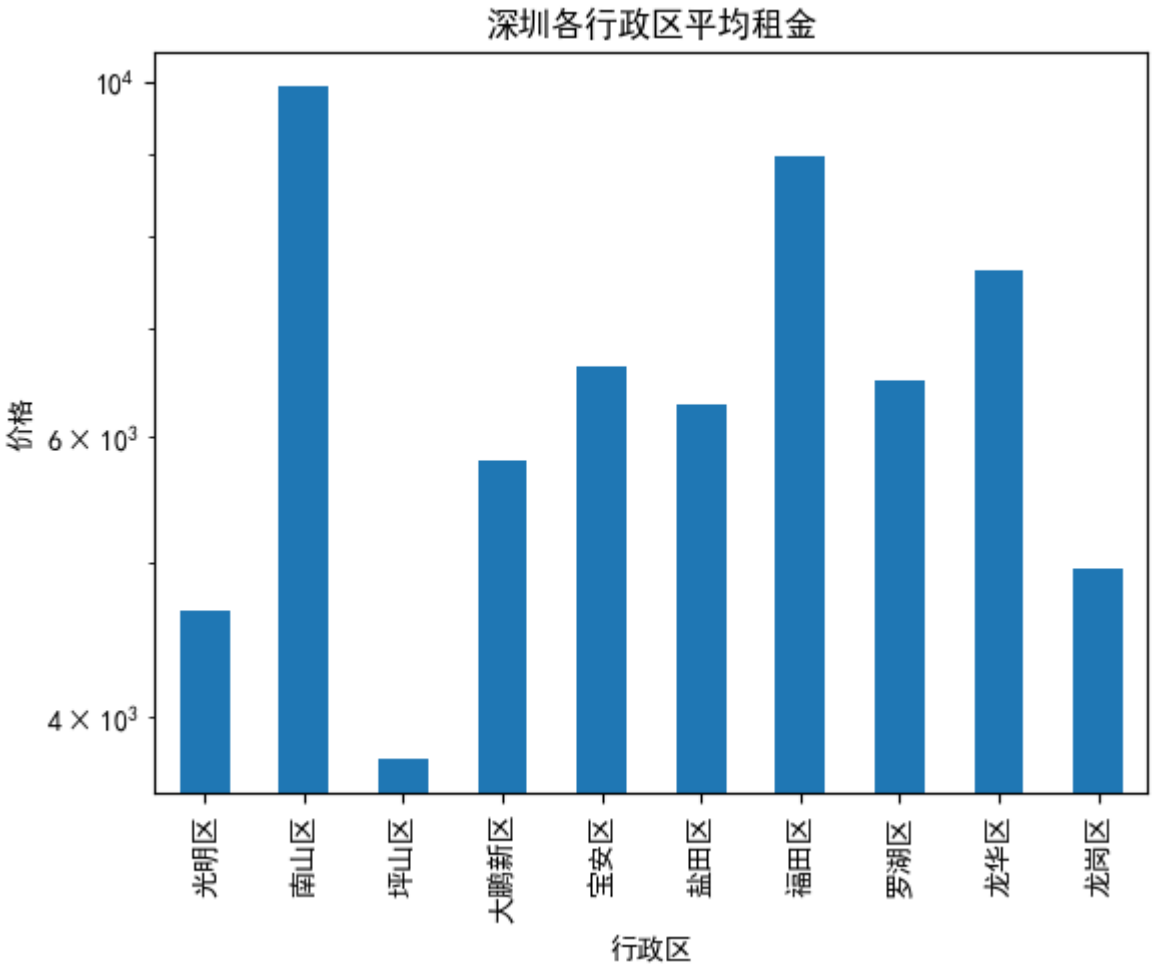
1. 整体租金均价分析
   - 读入数据, 并按照 `district` (也就是行政区) 字段分组计算平均租金, 并绘制条形图进行分析

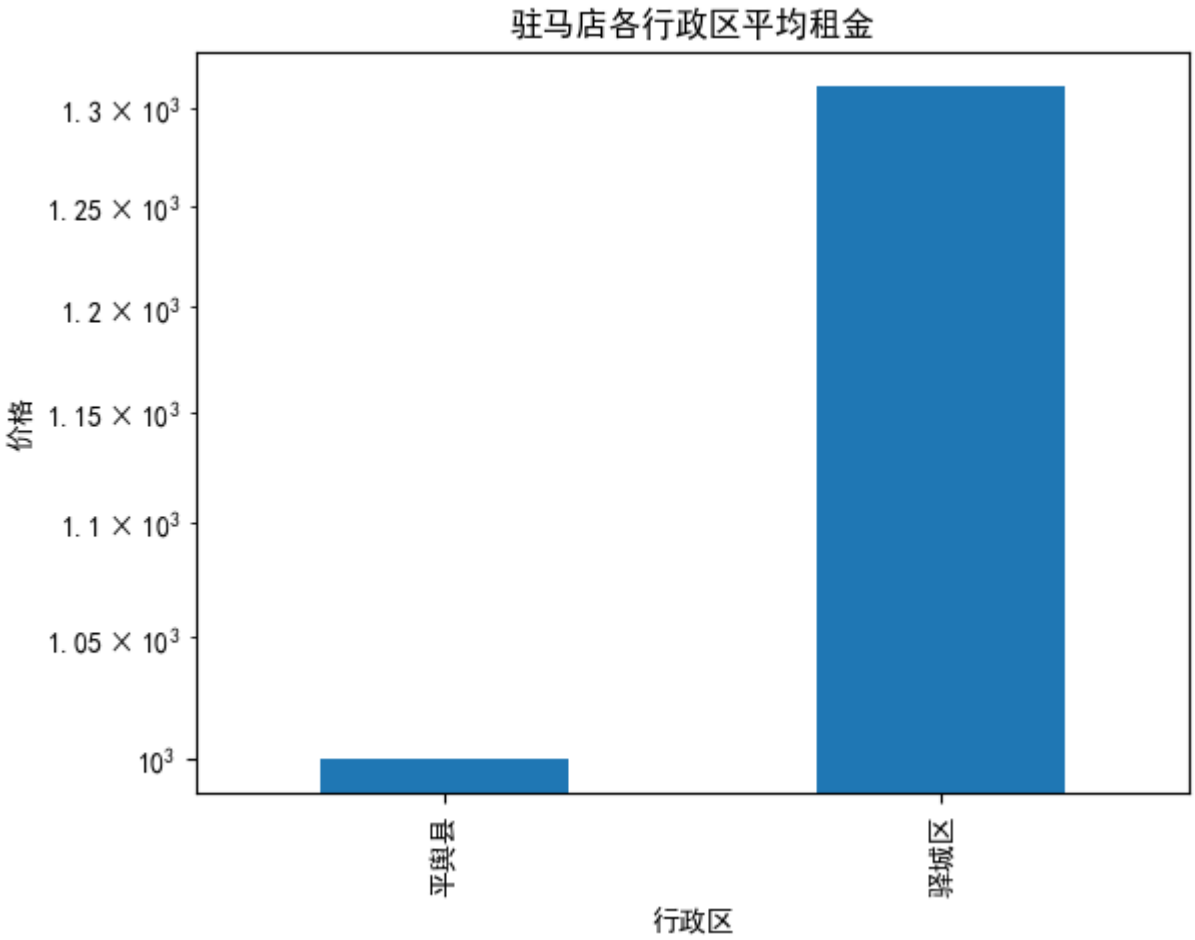
```
tot_dis_avg = data.groupby(['city', 'district'])['price'].mean().round(2).reset_index()

for city in tot_dis_avg['city'].unique():
    city_data = tot_dis_avg[tot_dis_avg['city'] == city]

    plt.figure()
    city_data.plot(kind='bar', x='district', y='price', legend=False,
                   title=f'{city}各行政区平均租金', xlabel='城市', ylabel='价格')
    plt.yscale('log')
    plt.savefig(f'imgs/{city}各行政区平均租金.png')
```


上海各行政区平均租金
价格
10^4
6×10^3
4×10^3
3×10^3
嘉定
奉贤
宝山
崇明
徐汇
普陀
杨浦
松江
浦东
虹口
金山
长宁
闵行
青浦
静安
黄浦
行政区


北京各行政区平均租金
价格
10^4
6×10^3
4×10^3
3×10^3
2×10^3
东城
丰台
亦庄开发区
大兴
密云
平谷
延庆
怀柔
昌平
朝阳
海淀
石景山
西城
通州
门头沟
顺义
行政区


广州各行政区平均租金
价格
6×10^3
4×10^3
3×10^3
从化
南沙
南海
增城
天河
海珠
番禺
白云
花都
荔湾
越秀
黄埔
行政区

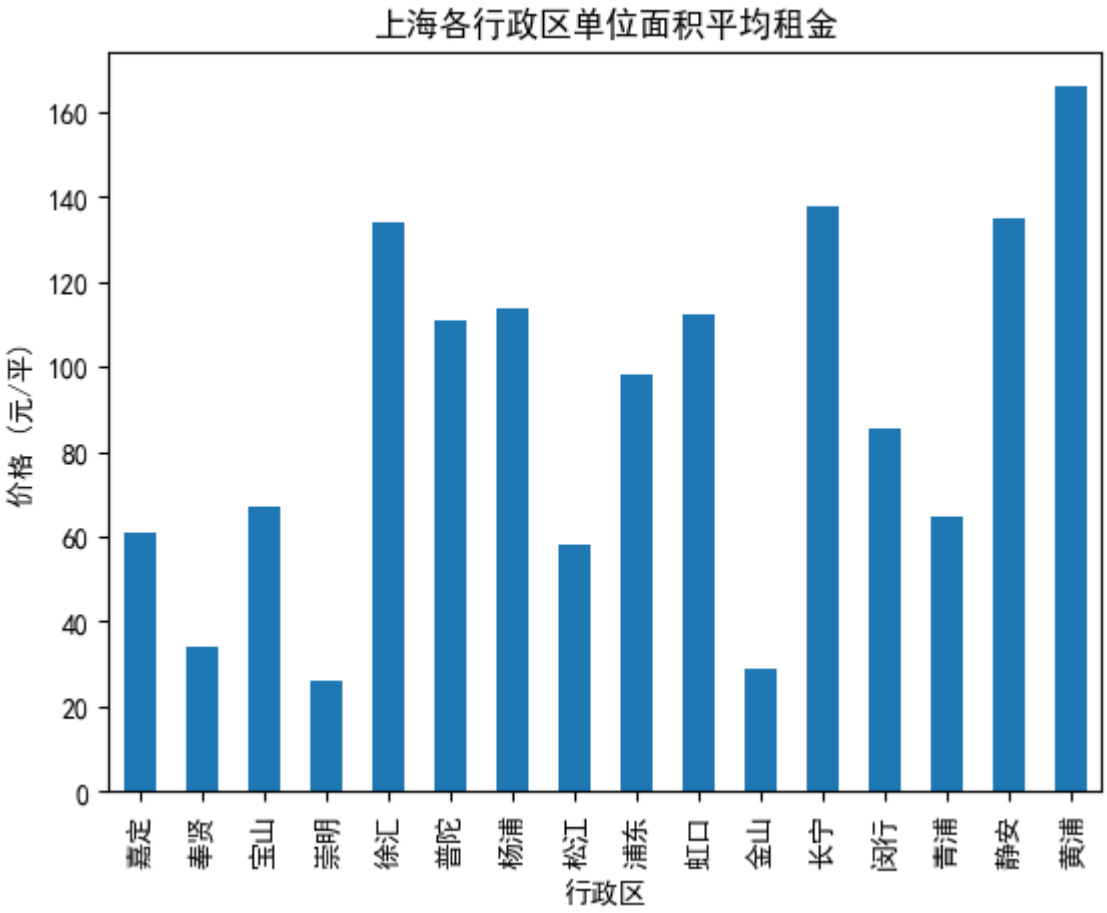

深圳各行政区平均租金
价格
10^4
6×10^3
4×10^3
光明区
南山区
坪山区
大鹏新区
宝安区
盐田区
福田区
罗湖区
龙华区
龙岗区
行政区

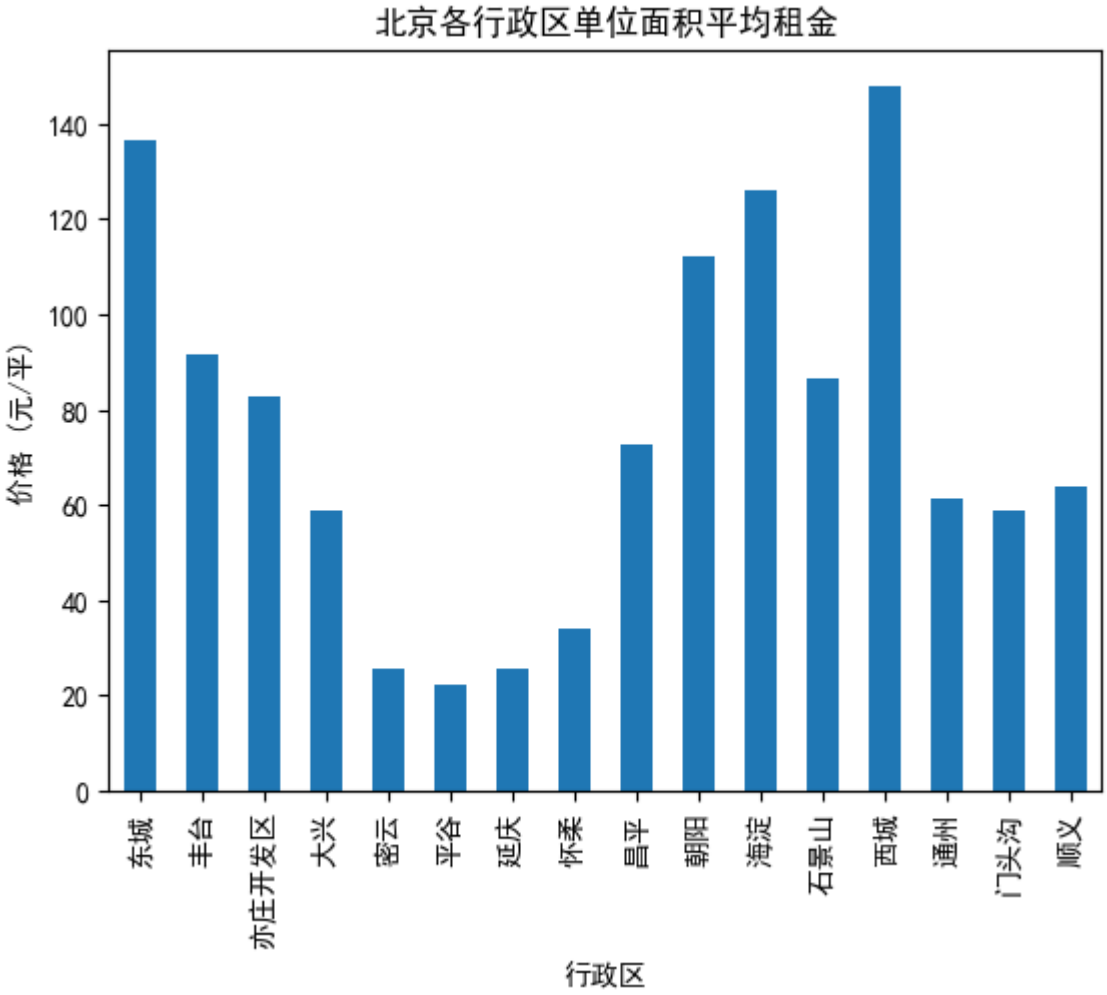


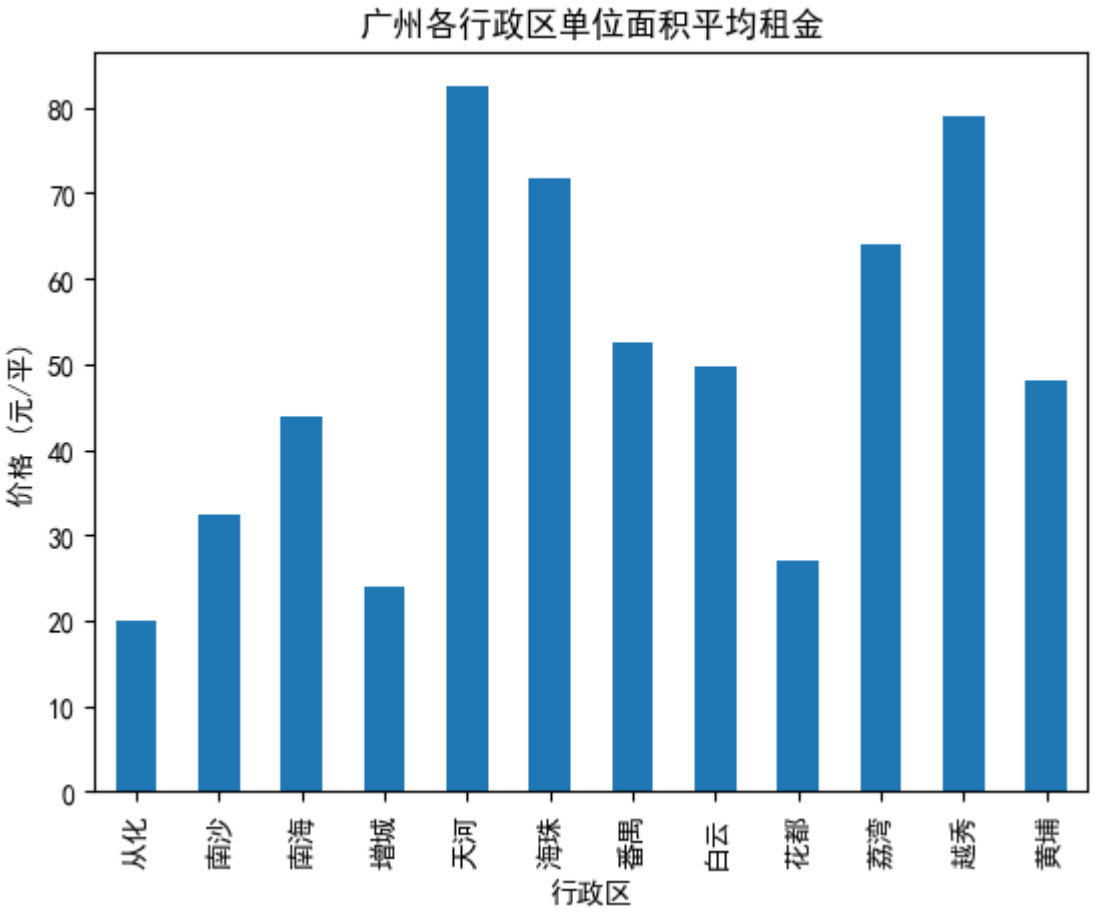
可以看到上海的黄埔、 北京的朝阳、 广州的天河和深圳的南山在各自的城市的各个行政区中平均租金最高; 上海的崇明、北京的平谷、广州的从化和深圳的坪山在各自城市的各个行政区中的平均租金最低.

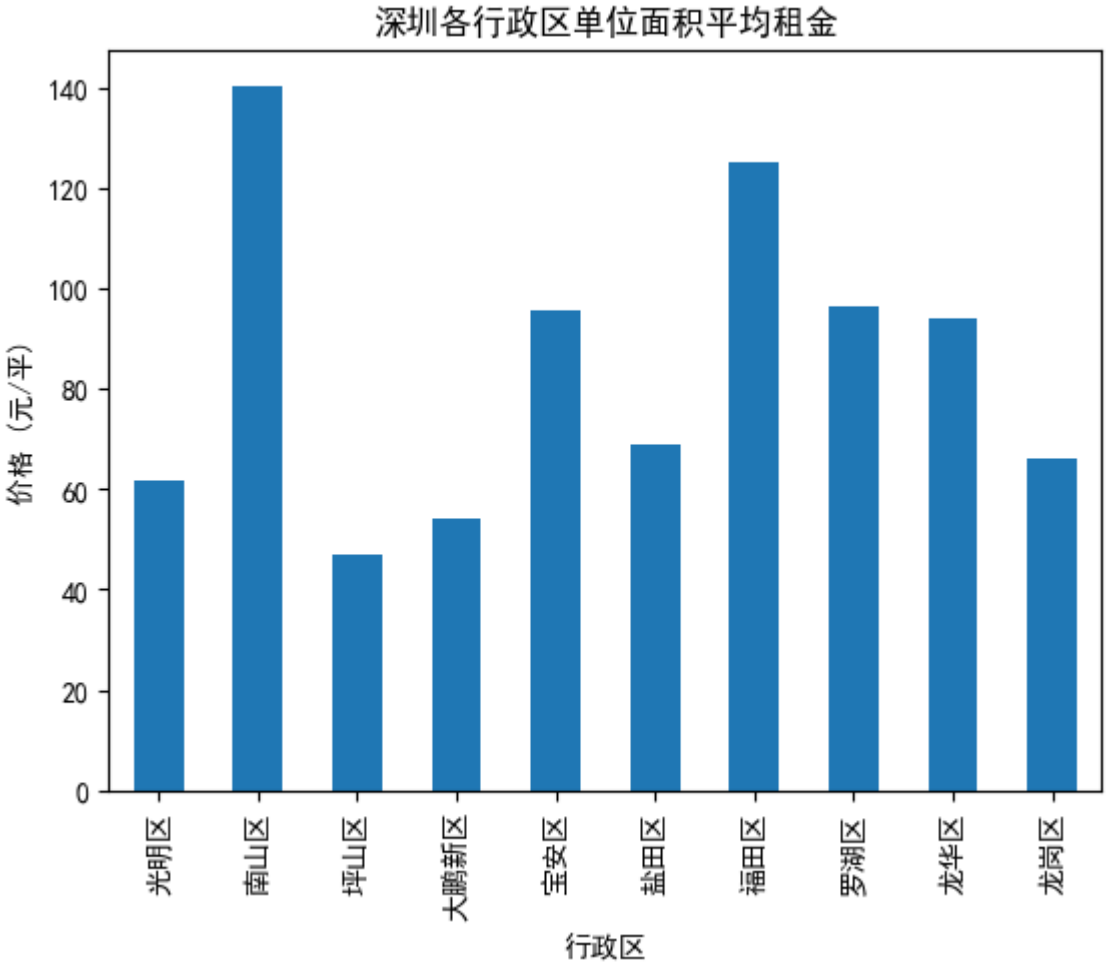
2. 单位面积租金均价分析
   - 读入数据, 并按照 `district` (也就是行政区) 字段分组计算单位面积平均租金, 并绘制条形图进行分析

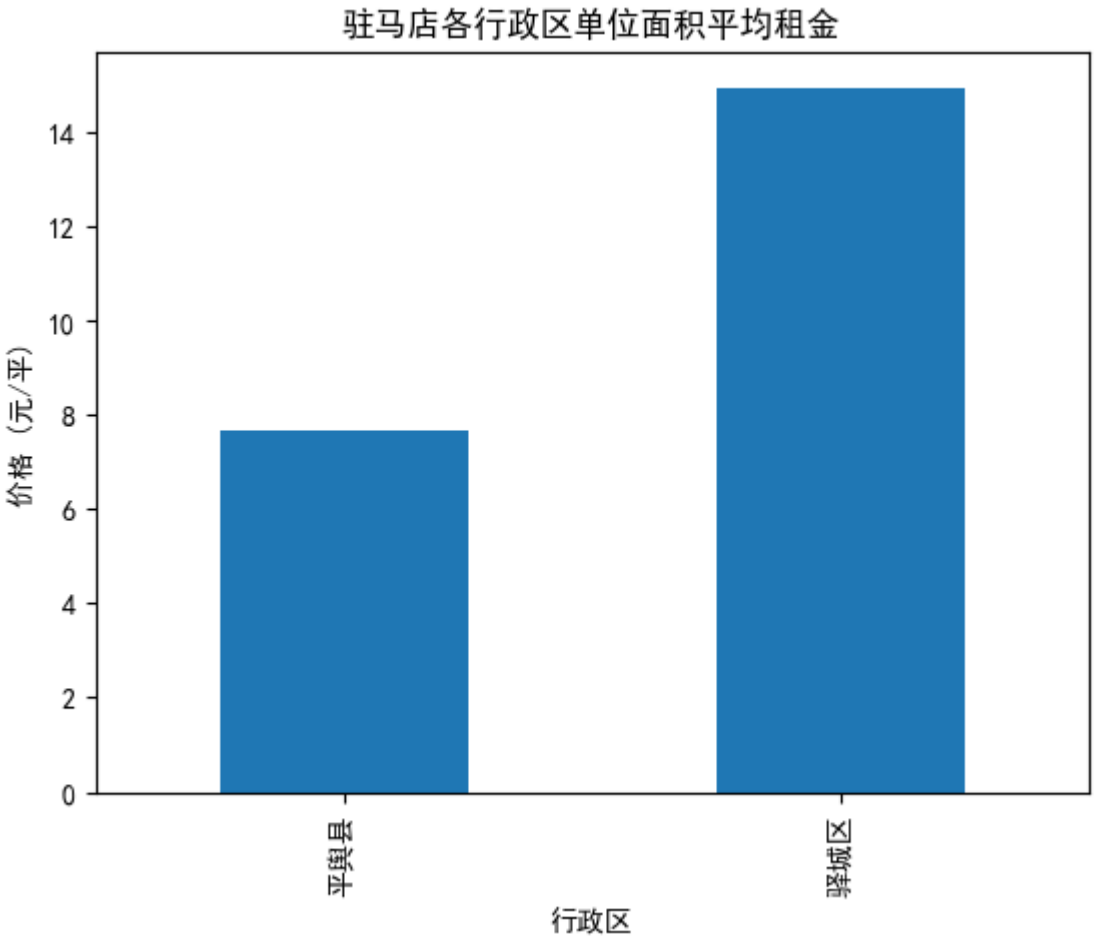
```
unit_dis_avg = data.groupby(['city', 'district'])[['price', 'area']].apply(lambda x: pd.Ser
    'mean': (x['price'] / x['area']).mean().round(2),
})).reset_index()

for city in unit_dis_avg['city'].unique():
    city_data = unit_dis_avg[unit_dis_avg['city'] == city]

    plt.figure()
    city_data.plot(kind='bar', x='district', y='mean', legend=False,
                   title=f'{city}各行政区单位面积平均租金', xlabel='城市', ylabel='价格（元/平

    plt.savefig(f'imgs/{city}各行政区单位面积平均租金.png')
```


上海各行政区单位面积平均租金
价格（元/平）
0
20
40
60
80
100
120
140
160
嘉定
奉贤
宝山
崇明
徐汇
普陀
杨浦
松江
浦东
虹口
金山
长宁
闵行
青浦
静安
黄浦
行政区


北京各行政区单位面积平均租金
价格（元/平）
0
20
40
60
80
100
120
140
东城
丰台
亦庄开发区
大兴
密云
平谷
延庆
怀柔
昌平
朝阳
海淀
石景山
西城
通州
门头沟
顺义
行政区


广州各行政区单位面积平均租金
价格（元/平）
0
10
20
30
40
50
60
70
80
从化
南沙
南海
增城
天河
海珠
番禺
白云
花都
荔湾
越秀
黄埔
行政区


深圳各行政区单位面积平均租金
价格（元/平）
0
20
40
60
80
100
120
140
光明区
南山区
坪山区
大鹏新区
宝安区
盐田区
福田区
罗湖区
龙华区
龙岗区
行政区



可以看到上海的黄埔、 北京的西城、 广州的天河和深圳的南山在各自的城市的各个行政区中平均租金最高; 上海的崇明、北京的平谷、广州的从化和深圳的坪山在各自城市的各个行政区中的平均租金最低.

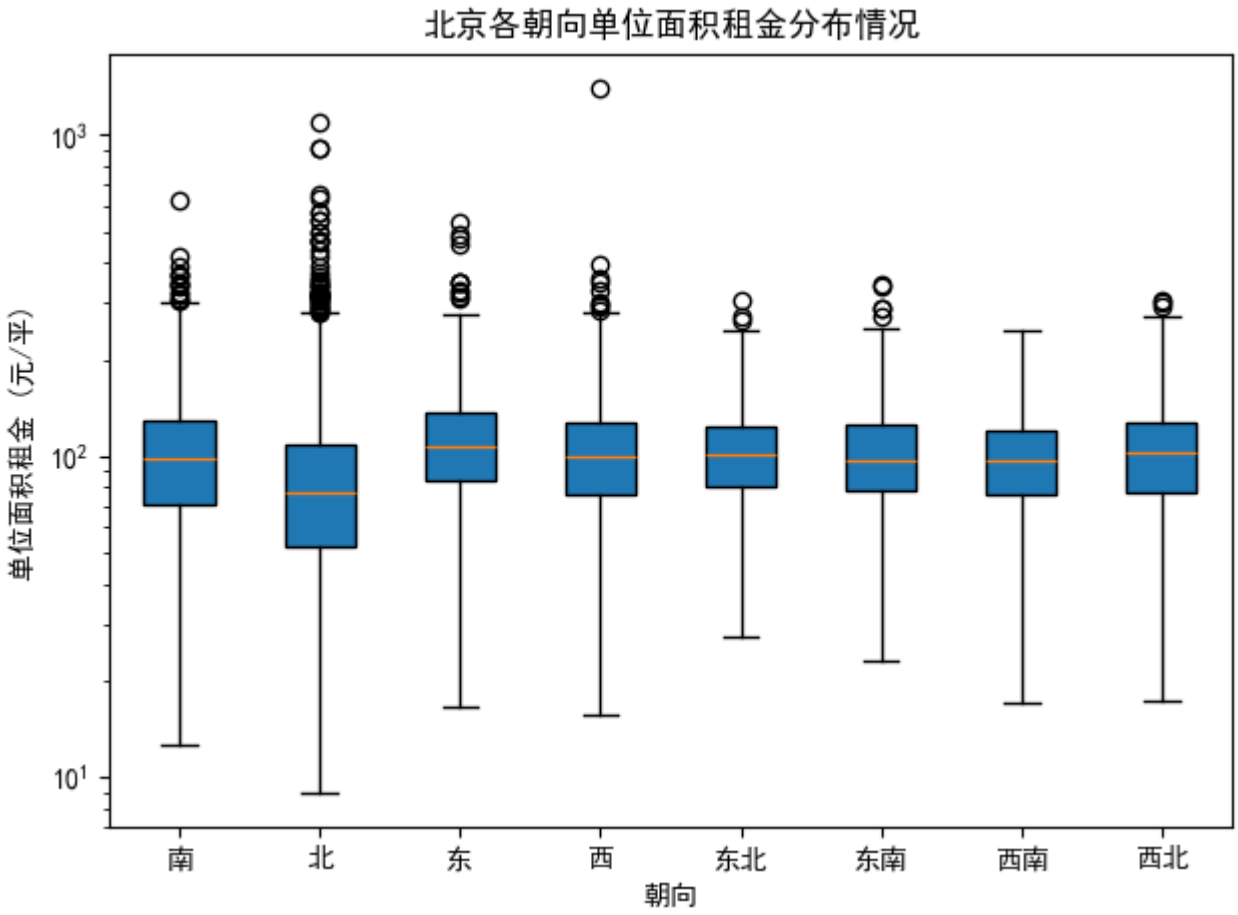
可以看到各行政区单位面积租金的排名和平均租金的排名基本一致

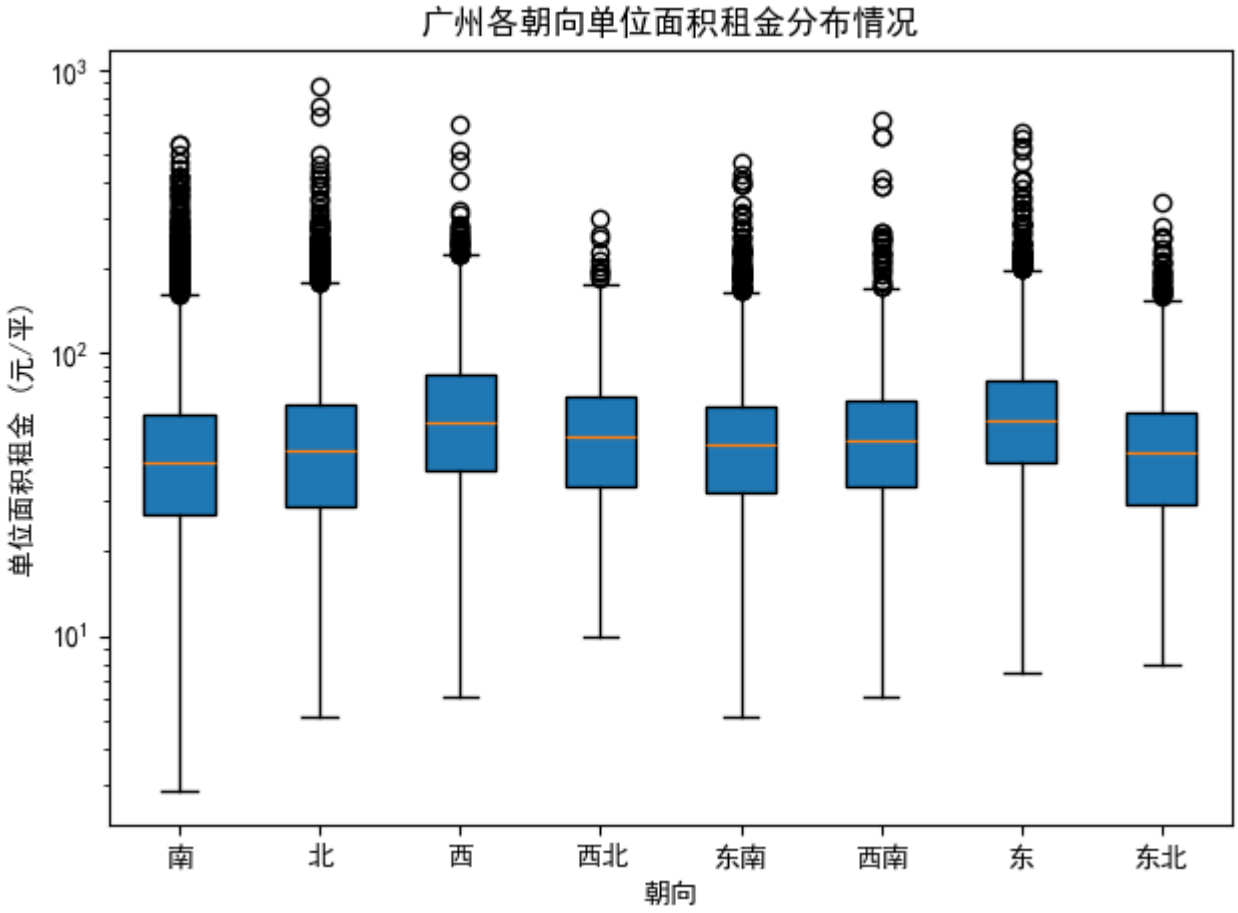
> 驻马店理论上只有驿城区一个区, 所以不予分析

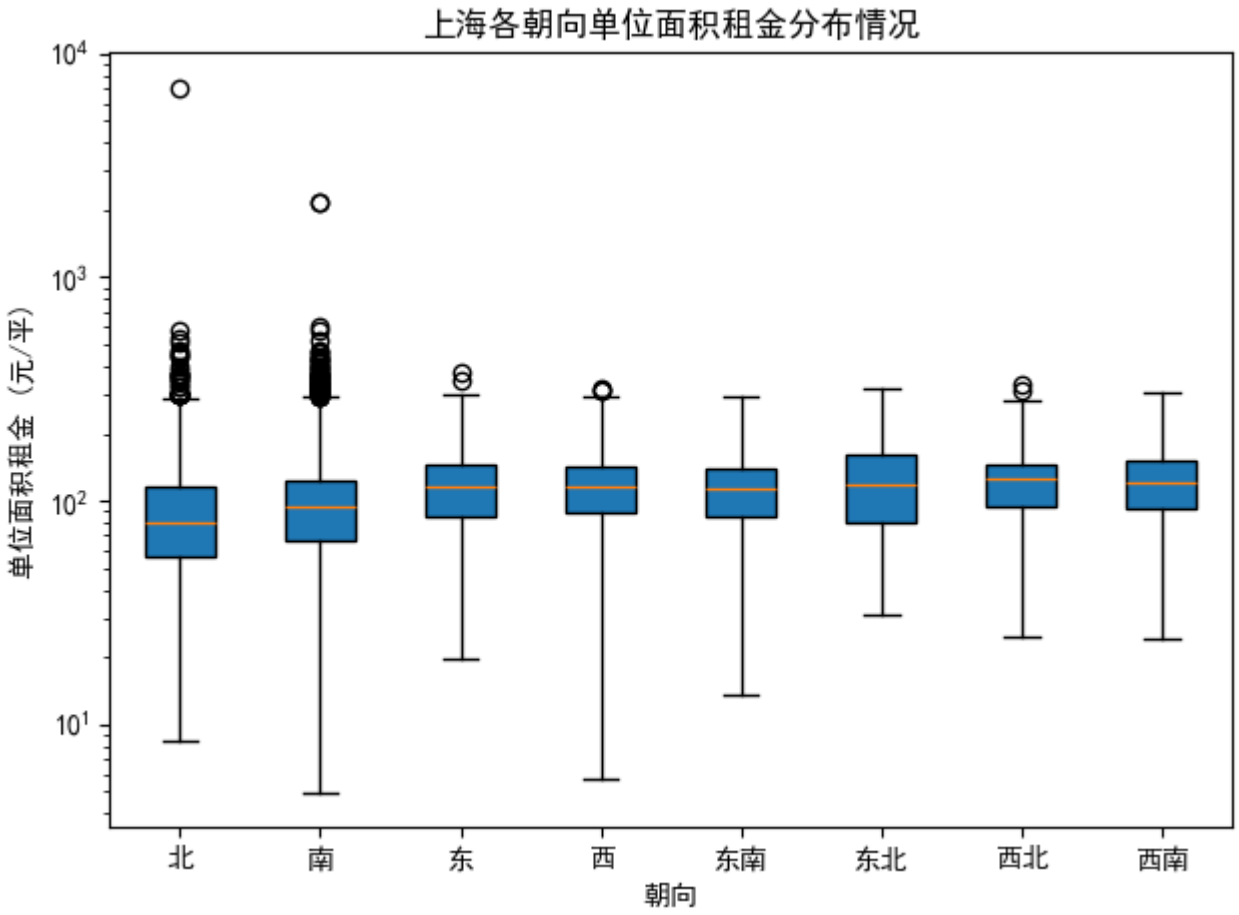
比较各个城市不同朝向的单位面积租金

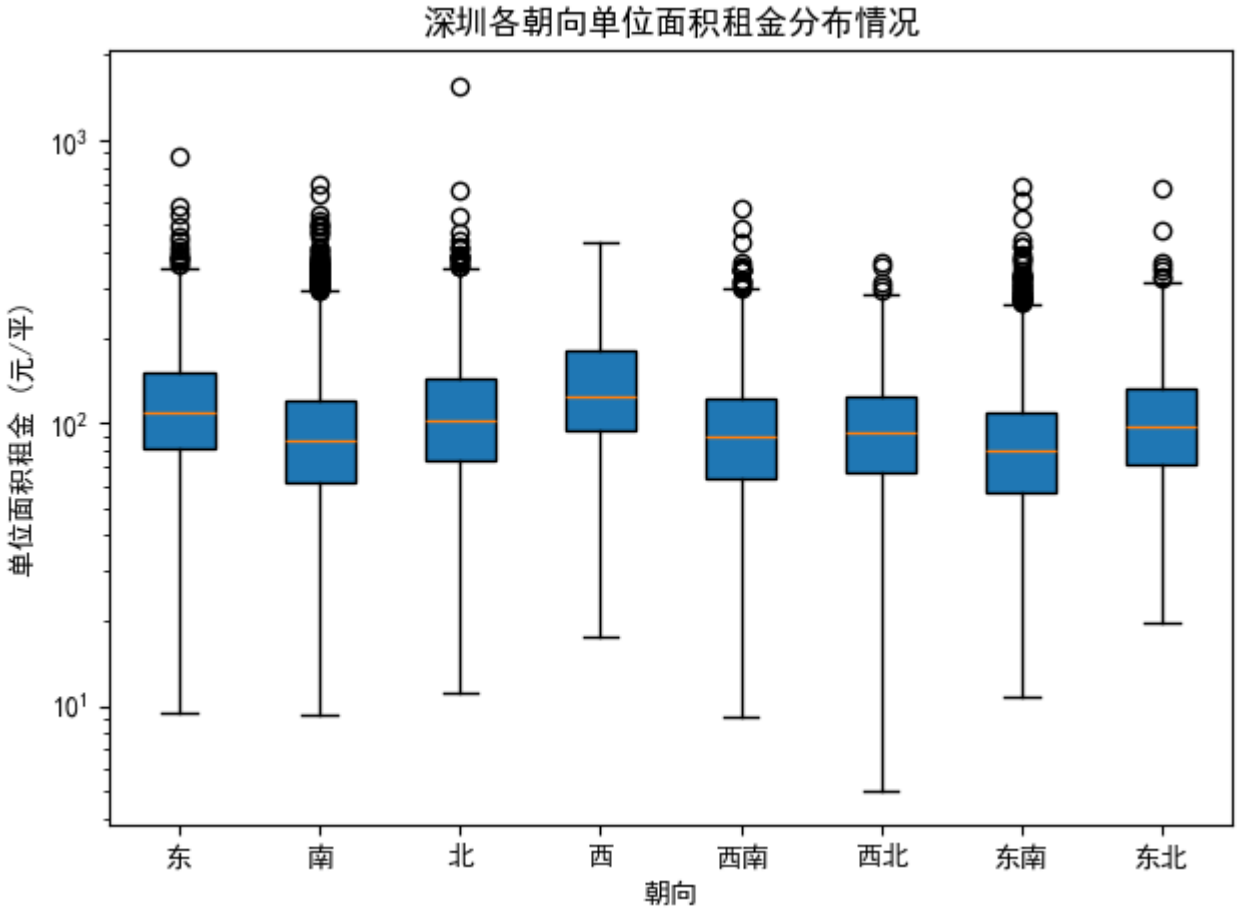
对每个城市的每个朝向进行分类, 求单位面积租金, 为了显示租金的分布情况, 所以绘制箱线图进行绘制

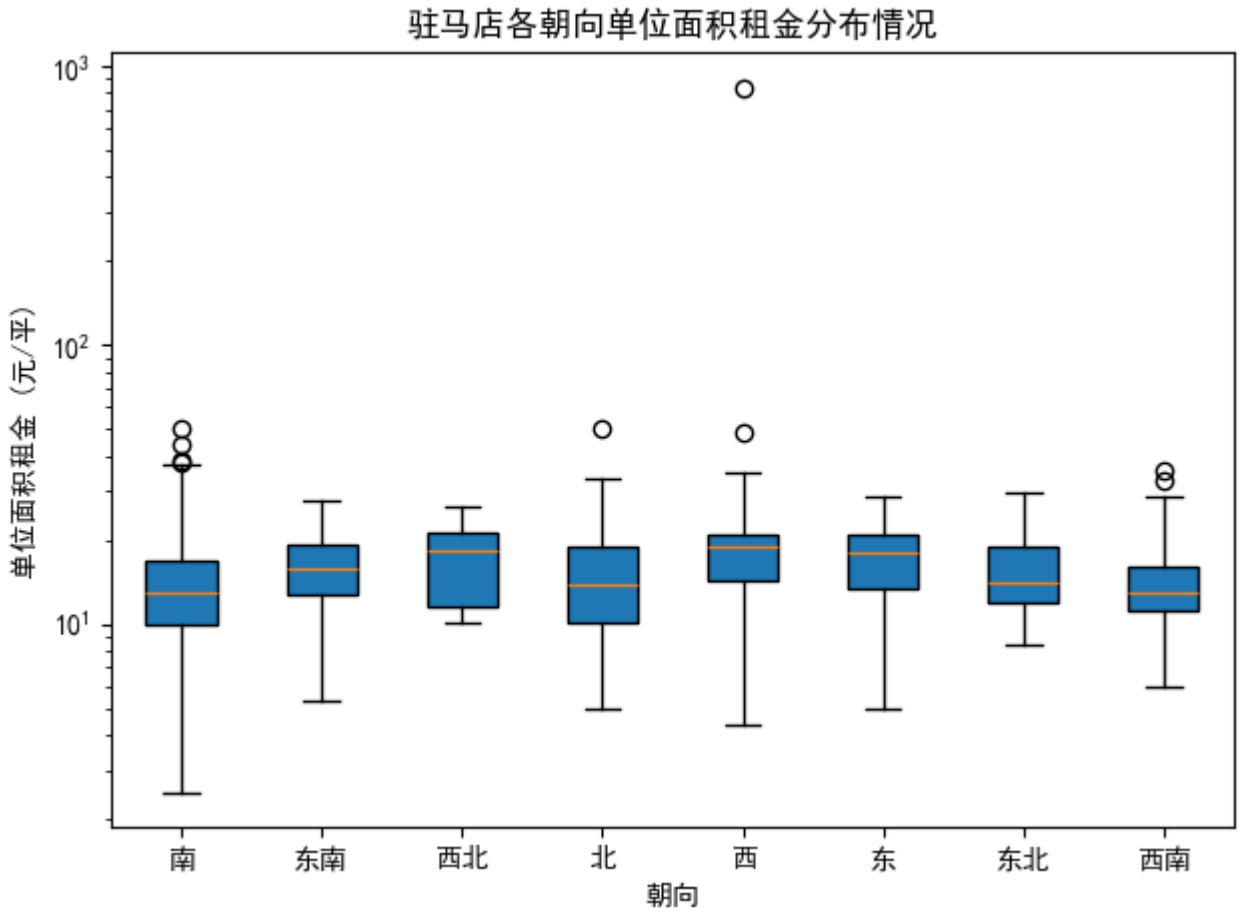
```
for city in data['city'].unique():
    city_data = data[data['city'] == city]
    orientations = city_data['orientation'].unique()
    orientations = [x for x in orientations if not pd.isnull(x)]

    # 创建一个空列表，用于存储每个朝向的单位面积租金数据
    orientation_prices = []

    # 遍历每个朝向
    for orientation in orientations:
        orientation_data = city_data[city_data['orientation'] == orientation]
        area_price = orientation_data['price'] / orientation_data['area']
        orientation_prices.append(area_price)

    # 绘制箱线图
    plt.figure()
    plt.boxplot(orientation_prices, patch_artist=True, whis=3)

    plt.title(f'{city}各朝向单位面积租金分布情况')
    plt.xlabel('朝向')
    plt.ylabel('单位面积租金（元/平）')

    plt.xticks(range(1, len(orientations) + 1), orientations)
    plt.yscale('log')

    plt.tight_layout()
    plt.savefig(f'imgs/{city}各朝向单位面积租金分布情况.png')
```


北京各朝向单位面积租金分布情况
单位面积租金（元/平）
10^3
10^2
10^1
南
北
东
西
东北
东南
西南
西北
朝向


广州各朝向单位面积租金分布情况
单位面积租金（元/平）
10^3
10^2
10^1
南
北
西
西北
东南
西南
东
东北
朝向


上海各朝向单位面积租金分布情况
单位面积租金（元/平）
朝向
北
南
东
西
东南
东北
西北
西南


深圳各朝向单位面积租金分布情况
单位面积租金（元/平）
朝向
东
南
北
西
西南
西北
东南
东北



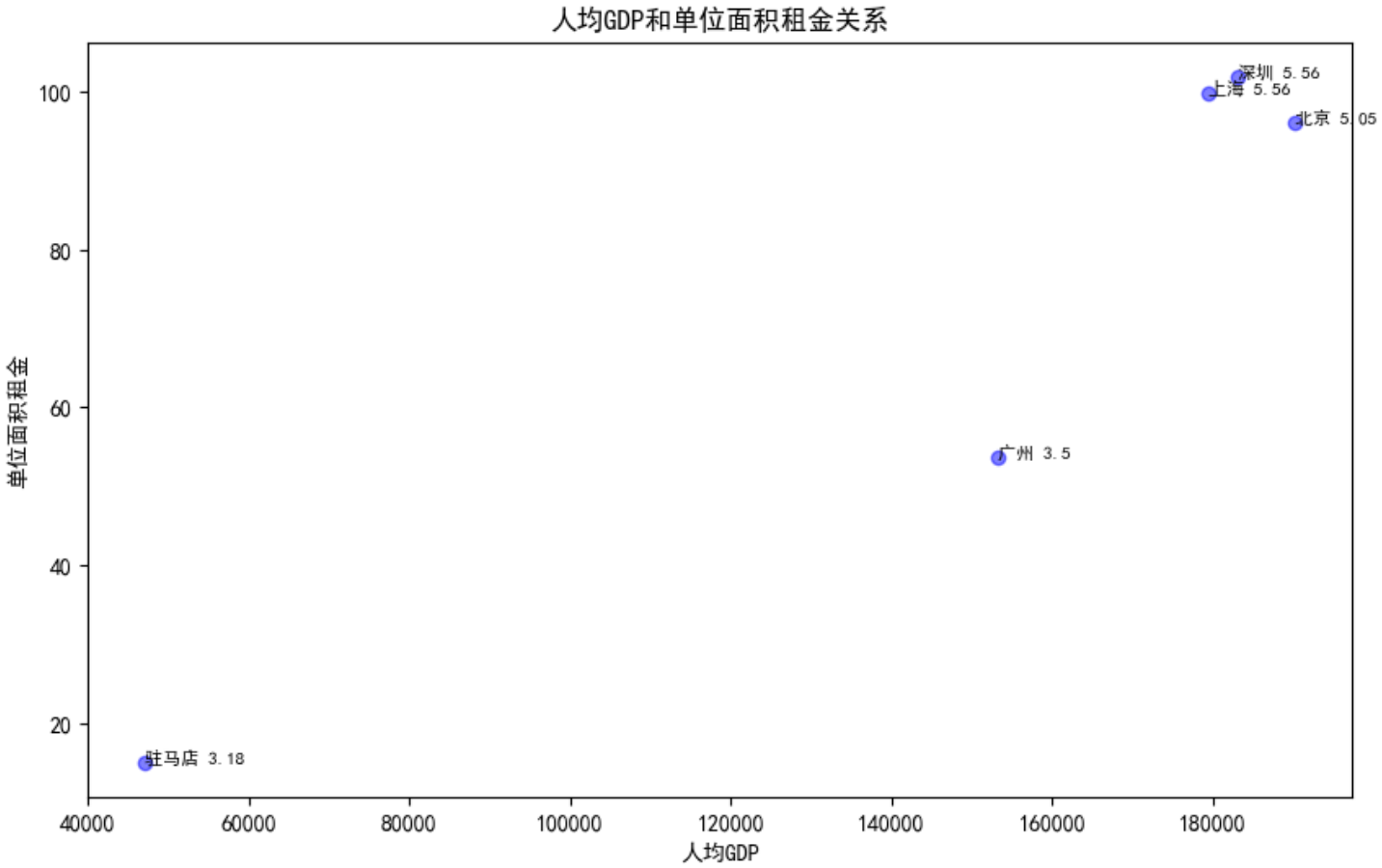
经过分析可知, 北京的东朝向、广州的东和西朝向、上海的东北朝向、深圳的西朝向、驻马店的东和西和西北朝向的单位面积租金较高,
这可能是因为东和西朝向的房子温度比较适宜, 并且也不会太晒.

北京的北朝向、广州、上海、深圳和驻马店的南朝向的房子单位面积租金较低. 这可能是因为北京位于北方, 北朝向的房子室内见不到太阳导致温度较低,
而其他城市比较偏向于南方, 南朝向的房子会导致太晒, 所以租金较低.

再进一步分析, 在北上广深这四个城市的箱线图中, 异常点过多, 这说明不是正态分布, 可能还有一些其他因素, 比如地理位置也对房子租金影响较大.

各个城市人均 GDP 和单位面积租金的关系

查阅各个城市 2022 年 `人均 GDP` 并存入 `datasets/city.csv` 文件, 绘制散点图进行分析

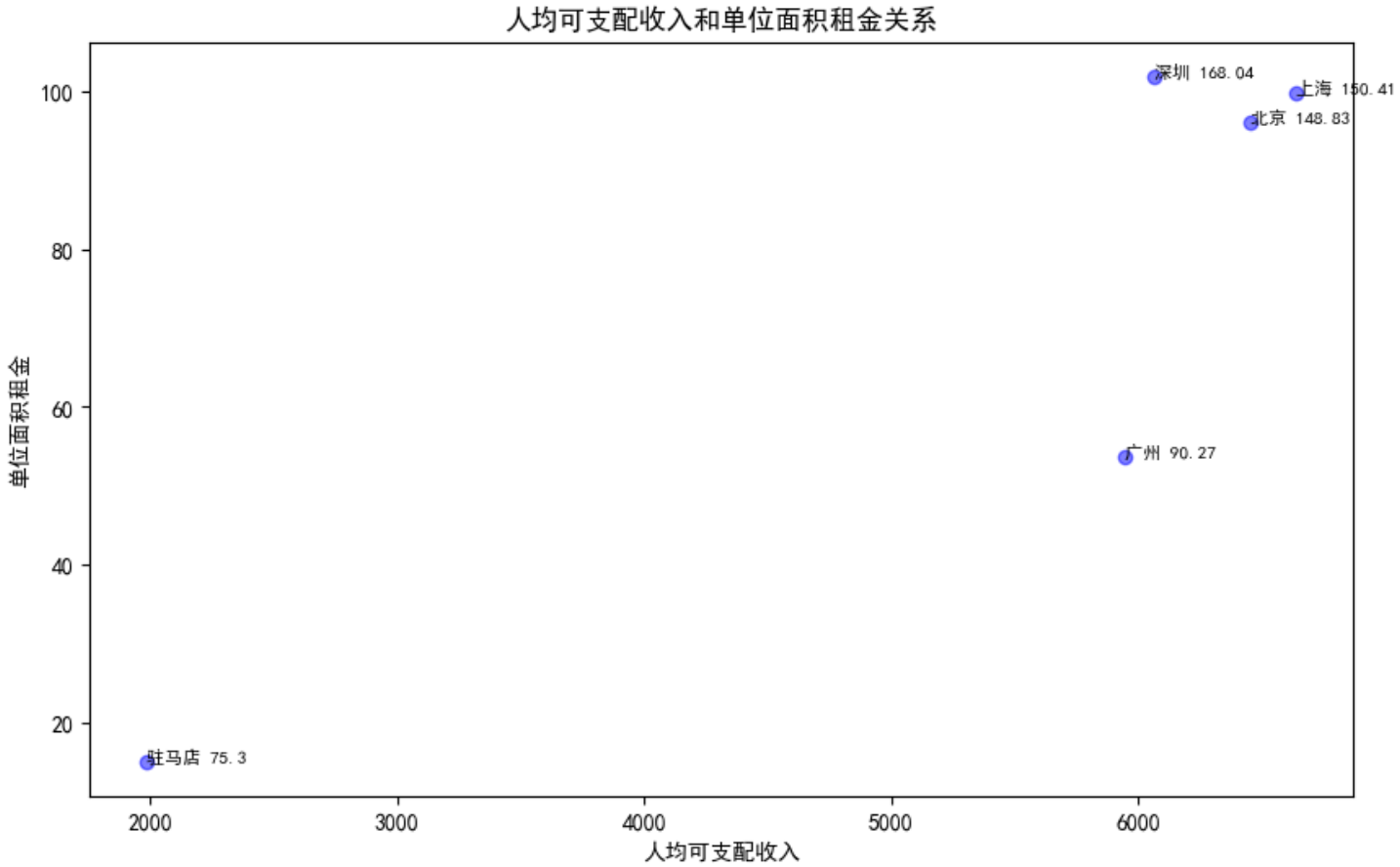


图中性价比计算公式为 $\frac{\text{单位面积租金}\times 10000}{\text{人均}GDP}$, 这个公式可以反应单位面积租金对于人均 GDP 的占比情况. 可以看到驻马店性价比最高, 广州其次, 上海和深圳相等

各城市平均工资和单位面积租金分布关系

因为无法得到各个城市的平均工资情况, 所以使用人均可支配收入代替平均工资进行分析. 查阅各个城市 2022 年 人均可支配收入

并存入 `datasets/city.csv` 文件, 绘制散点图进行分析



图中负担计算公式为 $\frac{\text{单位面积租金}\times 10000}{\text{人均可支配收入}}$, 这个公式可以反应单位面积租金对于人均可支配收入的占比情况.

可以看到在深圳租房负担最重, 驻马店负担最轻.