租房数据分析

2020211346

倪玮昊

数据抓取和预处理

1.抓取数据

以下的代码是一个爬虫程序，它从链家网租房信息页面爬取了房源的标题、户型、面积、价格范围等信息，然后将这些信息写入一个 CSV 文件中。

首先，它使用了 urllib 和 BeautifulSoup 库来发送 HTTP 请求并解析 HTML 页面。它还使用了 csv 库来创建并写入 CSV 文件。

其次，它首先使用了 urlopen 函数打开了链家网租房页面，然后使用 BeautifulSoup 解析了 HTML 页面，并查找了页面中的所有房源信息。

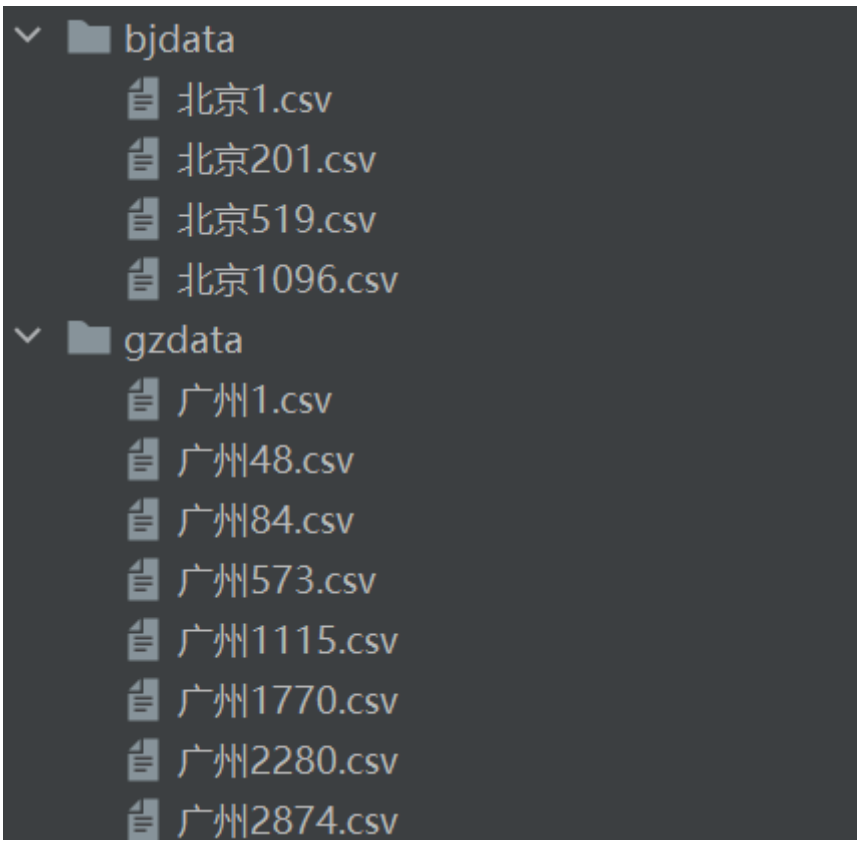
接着，它使用了一个 for 循环来遍历页面中的所有房源信息，并提取出房源的标题、户型、面积和价格范围。最后，它使用 csv 库的 writer 函数将这些信息写入了一个 CSV 文件中。

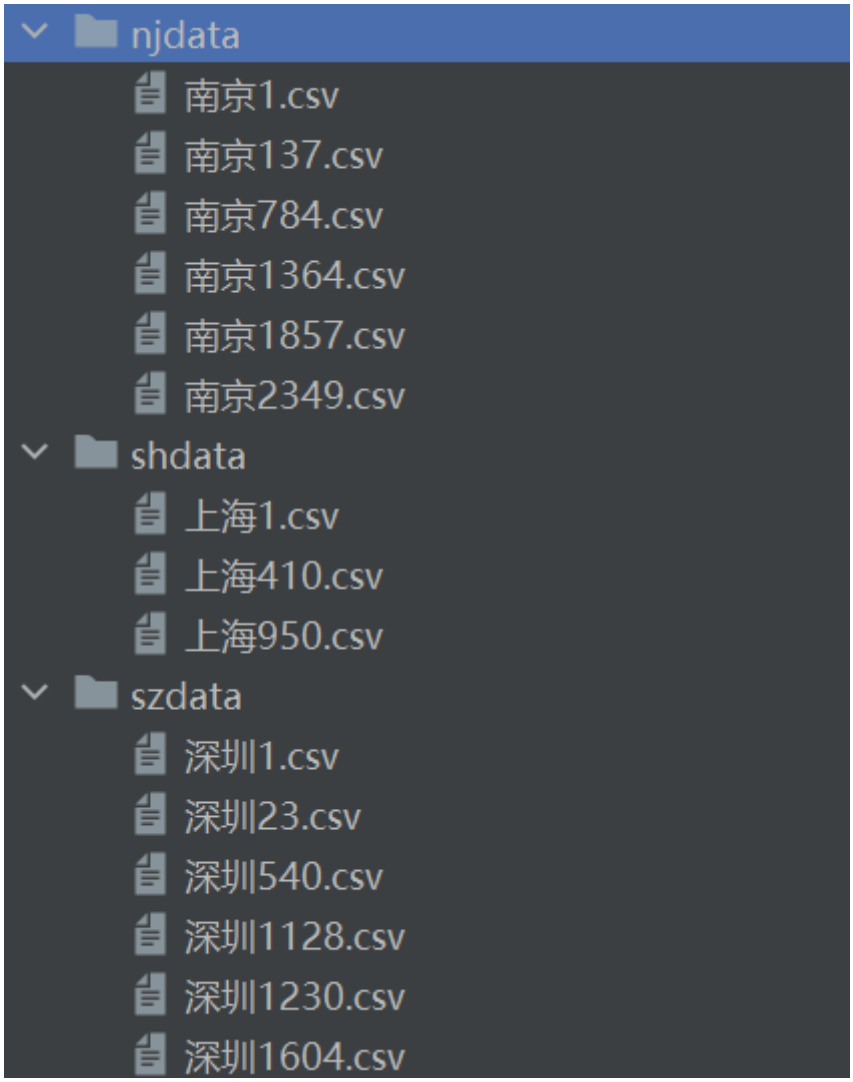
总的来说，这段代码实现了从链家网租房信息页面爬取房源信息并将其写入 CSV 文件的功能。

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import csv
csv_attributes = ['title', 'name', 'area', 'rooms', 'price_lower',
'price_upper']

info="地点"
num="页数"
name=info+str(num)
html = BeautifulSoup(urlopen("https://" + info + ".lianjia.com/zufang"),
features="lxml")
size_record = int(html.findAll(name="span", attrs={"class":"content__title--hl"})
[0].text)
size_page = int(size_record/30)
if( size_record%30 > 0 ):
    size_page = size_page+1
print(name, size_record, size_page)

# generate name.csv
with open(info+'data/'+name + ".csv", 'w', newline='', encoding='utf-8-sig') as
outputFile:
    writer = csv.writer(outputFile)
    writer.writerow(csv_attributes)
    # scrapy bjdata in pages & copy bjdata into file
    for i in range(num, size_page+1):
        print(i)
        html = BeautifulSoup(urlopen("https://" + info + ".lianjia.com/zufang/pg"
+ str(i)),
                             features="lxml")
```

```
        # get bjdata into list
        for title, des, price in zip(
                html.findAll("p", {"class": "content__list--item--title"}),
                html.findAll("p", {"class": "content__list--item--des"}),
                html.findAll("span", {"class": "content__list--item-price"})
        ):
            # solve title
            title = title.get_text().strip()
            # solve des
            des_list = des.get_text().split('/')
            if len(des_list) == 5:
                name = des_list[0].replace("\n","").strip()
                area = des_list[1].replace("\n","").strip()[:-1]
                rooms = des_list[3].replace("\n", "").strip()[0]
            elif len(des_list) == 3:
                name = ""
                area = des_list[0].replace("\n", "").strip()[:-1]
                rooms = des_list[2].replace("\n", "").strip()[0]
            # solve price
            text = price.get_text().split(' ')[0]
            if '-' in text:
                text = text.split('-')
                price_lower = text[0]
                price_upper = text[1]
            else:
                price_lower = text
                price_upper = text
            writer.writerows([[title, name, area, rooms, price_lower,
price_upper]])
```

由于在爬取过程中,遇到了反复中断的问题,所以,通过对page和info的修改,反复从中断处重新爬取,直到完全爬完所有数据(文件后面的数字表示从中止开始的页数:





2.清洗数据和融合数据

下面这段代码是一个数据处理程序，它对多个 CSV 文件进行了数据聚合、筛选和计算。

首先，它使用了 os 和 pandas 库，并创建了一个 cities 字典，其中包含了五个城市的名称和简称。然后，它使用了一个 for 循环来遍历这五个城市，并对每个城市的数据进行处理。

其次，它使用了 os 库的 listdir 函数来获取文件夹中的所有 CSV 文件，然后使用 pandas 库的 read_csv 函数将所有文件读入到一个列表中。接着，它使用了 pandas 库的 concat 函数来将这些数据帧合并成一个大的数据帧，并使用 to_csv 函数将数据帧写入到一个新的 CSV 文件中,实现了文件的合并。

然后，它使用了 read_csv 函数读取了这个新的 CSV 文件，并使用了一个新的列来计算平均价格。接着，它使用了 drop 函数删除了最低价和最高价两列。

接下来，它又使用了 to_csv 函数将处理后的数据帧写入了一个新的 CSV 文件中，并使用了 read_csv 函数读取了这个新的 CSV 文件。然后，它使用了 to_numeric 函数将面积和平均价格两列转换为数值型数据，并计算出了每平米价格。

最后，它又使用了 to_numeric 函数将房间数转换为数值型数据，并计算出了每房间的面积。然后，它使用了 to_csv 函数将处理后的数据帧写入到一个新的 CSV 文件中。

接着，它使用了 csv 库的 reader 函数读取了这个 CSV 文件，并将文件的内容读取到了一个列表中。然后，它使用了一个 for 循环遍历了这个列表中的每一行，并读取了title中的朝向的内容和第二列中断板块,并且删除了不符合规定的行。最后，它使用了 csv 库的 writer 函数将修改后的列表写入到了一个新的 CSV 文件中。

总的来说，这段代码实现了对多个 CSV 文件进行数据聚合、筛选和计算，并将处理后的数据写入到一个新的 CSV 文件中的功能。

```
import os
import pandas as pd
import csv
cities = {'广州': 'gz', '上海': 'sh', '北京': 'bj', '深圳': 'sz', '南京': 'nj'}
for name, info in cities.items():
    folder=info+"data"
    print("对"+name+"的数据进行合并和计算")
    # 获取文件夹中的所有 CSV 文件
    filenames = [file for file in os.listdir(folder) if file.endswith(".csv")]
    # 将所有文件读入到一个列表中
    df_list = [pd.read_csv(folder+"/" + file) for file in filenames]
    # 合并数据帧
    df = pd.concat(df_list)
    # 将合并后的数据帧写入新的 CSV 文件
    df.to_csv(folder+".csv", index=False)
    # 读取 CSV 文件
    df = pd.read_csv(folder+".csv")
    # 计算平均价格添加为新列
    df["price_average"] = (df["price_lower"] + df["price_upper"])/2
    print("计算平均价格成功")
    df = df.drop("price_lower", axis=1)# 删除列
    df = df.drop("price_upper", axis=1)# 删除列
    print("删除最低价和最高价列成功")
    # 将处理后的数据帧写入新的 CSV 文件
    df.to_csv(folder+".csv", index=False)
    # 读取 CSV 文件
    df = pd.read_csv(folder + ".csv")
    df["area"]=pd.to_numeric(df["area"], errors="coerce")
    df["price_average"]=pd.to_numeric(df["price_average"], errors="coerce")
    df["price_perArea"]=df["price_average"]/df["area"]
    df["price_perArea"]=df["price_perArea"].round(2)
    print("计算每平米价格成功")
    df["rooms"] = pd.to_numeric(df["rooms"], errors="coerce")
    df["area_perRoom"] = df["area"] / df["rooms"]
    df["area_perRoom"] = df["area_perRoom"].round(2)
    print("计算每房间面积成功")
    df.to_csv(folder+".csv", index=False)
    print("保存成功\n")
    #读取增加一列朝向
    with open(folder+".csv", 'r', encoding='utf-8') as file:
        # 创建 CSV 读取器
        reader = csv.reader(file)
        # 将 CSV 文件的内容读取到列表中
        rows = list(reader)

    # 遍历每一行，并修改第二列
    for row in rows:
        words = row[0].split()
        word2 = row[1].split('-')
        if(words[0]=="title" ):
            row.append("forword")
            row.append("block")
```

```
        elif(len(words)>=3 and len(word2)==3 and words[len(words)-1]!="1室1厅"
and words[len(words)-1]!="2室0厅"and words[len(words)-1]!="2室1厅"and
words[len(words)-1]!="3室1厅"and words[len(words)-1]!="3室2厅"and
words[len(words)-1]!="4室2厅"):
            row.append(words[len(words)-1])
            row.append(word2[1])
    # 打开 CSV 文件，并创建 CSV 写入器
    with open(folder+".csv", 'w', encoding='utf-8',newline='') as file:
        writer = csv.writer(file)
        # 将修改后的数据写回到 CSV 文件中
        writer.writerows(rows)
    df = pd.read_csv(folder+".csv")
    # 删除某列为空的行
    df = df.dropna(subset=["block"])

    # 将修改后的 DataFrame 写入 CSV 文件
    df.to_csv(folder+".csv", index=False)
```

以bjdata.txt为例,最后清理出的数据效果为:

title	name	area	rooms	price_average	price_perArea	area_perRoom	forword	block
整租·瑞都国际北区1室1厅 北	通州-九棵树(家乐福)-瑞都国际北区	61.8	1	4500	72.82	61.8	北	九棵树(家乐福)
整租·建欣苑五里 2室1厅 北	丰台-大红门-建欣苑五里	85	2	6500	76.47	42.5	北	大红门
整租·地杰长安驿 1室0厅 北	朝阳-建国门外-地杰长安驿	45.47	1	6400	140.75	45.47	北	建国门外
整租·会展誉景 2房间 北	顺义-中央别墅区-会展誉景	50	2	4380	87.6	25	北	中央别墅区
整租·玲珑天地 1房间 北	海淀-定慧寺-玲珑天地	43.91	1	5500	125.26	43.91	北	定慧寺
整租·安贞西里 1室1厅 北	朝阳-安贞-安贞西里	38.32	1	6100	159.19	38.32	北	安贞
整租·北回归线 3房间 北	昌平-回龙观-北回归线	74	3	5800	78.38	24.67	北	回龙观
整租·上线6号 1房间 北	朝阳-常营-上线6号	51.62	1	3600	69.74	51.62	北	常营
整租·林肯公园 1房间 北	亦庄开发区-亦庄-林肯公园	36	1	4400	122.22	36	北	亦庄
整租·鸿坤七星长安 3房间 北	门头沟-滨河西区-鸿坤七星长安	65	3	3500	53.85	21.67	北	滨河西区

title	name	area	rooms	price_average	price_perArea	area_perRoom	forword	block
整租·林肯公园2房间复式北	亦庄开发区-亦庄-林肯公园	39.41	2	6800	172.55	19.7	北	亦庄
整租·首开幸福广场1室1厅 北	朝阳-工体-首开幸福广场	94.55	1	16000	169.22	94.55	北	工体
整租·鑫源国际3房间北	丰台-成寿寺-鑫源国际	53	3	7200	135.85	17.67	北	成寿寺
整租·后现代城B区 2室1厅北	朝阳-大望路-后现代城B区	111.83	2	11000	98.36	55.92	北	大望路
整租·长安6号 1室1厅北	朝阳-建国门外-长安6号	63	1	7000	111.11	63	北	建国门外
整租·常楹公元自由派 1房间北	朝阳-常营-常楹公元自由派	55.92	1	6000	107.3	55.92	北	常营
整租·国融国际1室0厅 北	亦庄开发区-亦庄-国融国际	33	1	3800	115.15	33	北	亦庄
整租·北欧印象1室0厅 北	西城-马连道-北欧印象	43.79	1	4700	107.33	43.79	北	马连道
整租·舒至嘉园2室1厅 北	海淀-西直门-舒至嘉园	77	2	9100	118.18	38.5	北	西直门
整租·东亚三环1房间北	丰台-马家堡-东亚三环	54	1	4900	90.74	54	北	马家堡

title	name	area	rooms	price_average	price_perArea	area_perRoom	forword	block
整租·长安6号 1室1厅 北	朝阳-建国门外-长安6号	63	1	7000	111.11	63	北	建国门外
整租·兴隆家园 1室1厅 北	朝阳-高碑店-兴隆家园	60.57	1	6000	99.06	60.57	北	高碑店
整租·荣丰2008 1室1厅 北	西城-广安门-荣丰2008	30	1	6000	200	30	北	广安门
整租·万年花城二期 1室0厅 北	丰台-玉泉营-万年花城二期	47	1	5000	106.38	47	北	玉泉营
整租·茶贸国际中心 1房间 北	西城-马连道-茶贸国际中心	40.33	1	3800	94.22	40.33	北	马连道
整租·方圆公寓 2室1厅 北	海淀-白石桥-方圆公寓	118	2	16000	135.59	59	北	白石桥
整租·国融国际 1室0厅 北	亦庄开发区-亦庄-国融国际	33	1	3800	115.15	33	北	亦庄
整租·后现代城D区 1室1厅 北	朝阳-大望路-后现代城D区	71	1	8500	119.72	71	北	大望路
整租·铁四区 1室1厅 北	西城-木樨地-铁四区	47.1	1	6500	138	47.1	北	木樨地
整租·流星花园二区 1室0厅 北	昌平-回龙观-流星花园二区	47.3	1	4900	103.59	47.3	北	回龙观

title	name	area	rooms	price_average	price_perArea	area_perRoom	forword	block
整租·北欧印象1室1厅 北	西城-马连道-北欧印象	43.26	1	4800	110.96	43.26	北	马连道
整租·霄云里8号1房间北	朝阳-燕莎-霄云里8号	65	1	8200	126.15	65	北	燕莎
整租·强佑清河新城府学上院2室1厅 北	海淀-清河-强佑清河新城府学上院	63	2	6100	96.83	31.5	北	清河
整租·绿地启航国际南区1房间北	顺义-后沙峪-绿地启航国际南区	41	1	3200	78.05	41	北	后沙峪
整租·鲁能钓鱼台美高梅公馆2房间北	丰台-刘家窑-鲁能钓鱼台美高梅公馆	46	2	7000	152.17	23	北	刘家窑
整租·华腾园3室1厅北	朝阳-双井-华腾园	107	3	10500	98.13	35.67	北	双井
整租·冬季星空1室1厅 北	丰台-马家堡-冬季星空	51	1	5500	107.84	51	北	马家堡
整租·城南大道2房间北	丰台-马家堡-城南大道	62	2	11000	177.42	31	北	马家堡
整租·上元君庭1室0厅 北	朝阳-奥林匹克公园-上元君庭	49	1	7350	150	49	北	奥林匹克公园
整租·北京汇1室0厅北	东城-崇文门-北京汇	52.41	1	7000	133.56	52.41	北	崇文门

title	name	area	rooms	price_average	price_perArea	area_perRoom	forword	block
整租·福顺里 1室1厅 北	丰台-北大地-福顺里	43	1	4300	100	43	北	北大地
整租·西山华府禧园 1室0厅 北	海淀-马连洼-西山华府禧园	37.84	1	6300	166.49	37.84	北	马连洼
整租·珺悦国际 2房间 北	大兴-天宫院-珺悦国际	65.48	2	3800	58.03	32.74	北	天宫院
整租·朗琴园 1室1厅 北	西城-广安门-朗琴园	59	1	7500	127.12	59	北	广安门
整租·安宁佳园 1室0厅 北	海淀-安宁庄-安宁佳园	37	1	4500	121.62	37	北	安宁庄
整租·中国铁建广场 2房间 北	朝阳-北苑-中国铁建广场	57	2	5200	91.23	28.5	北	北苑
整租·世茂宫园 1室1厅 北	朝阳-东大桥-世茂宫园	114	1	19500	171.05	114	北	东大桥
整租·工体北路 2室1厅 北	东城-工体-工体北路	60	2	7000	116.67	30	北	工体
整租·燕清源 1室0厅 北	海淀-清河-燕清源	40.02	1	5500	137.43	40.02	北	清河

数据分析

1.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。

下列这段代码首先导入了 os、pandas 和 numpy 库，并初始化了一个 5 行 8 列的数组 arr。

然后它使用了一个 for 循环遍历了一个字典 cities，其中包含了五个城市的名称和信息。对于每个城市，它会读取对应的 CSV 文件，使用 pandas 库的 max、min、median 和 mean 函数计算出租房价格和每平米租房价格的最大值、最小值、中位数和平均值，并将这些值保存到 arr 数组中。

最后，它使用了 matplotlib 库绘制了每个城市的租房价格和每平米租房价格的柱状图，并将图保存到了本地。

总的来说，这段代码实现了对五个城市的租房价格和每平米租房价格进行统计和可视化的功能。

```
# -*- coding: utf-8 -*-
import os
import pandas as pd
import numpy as np
arr = np.zeros((5, 8))
cities = {'北京': 'bj', '上海': 'sh', '广州': 'gz', '深圳': 'sz', '南京': 'nj'}
for name, info in cities.items():
    if(name=='北京'):
        n=0;
    elif(name=='上海'):
        n=1;
    elif(name=='广州'):
        n=2;
    elif(name=='深圳'):
        n=3;
    elif(name=='南京'):
        n=4;
    word=info+"data"+".csv"
    df = pd.read_csv(word)
    max_value = df["price_average"].max()
    arr[n][0]=max_value;
    min_value = df["price_average"].min()
    arr[n][1]=min_value;
    median_value = df["price_average"].median()
    arr[n][2]=median_value;
    average_value = df["price_average"].mean().round(2)
    arr[n][3]=average_value;
    print(name+"的最高租房价格为"+str(max_value))
    print("最低租房价格为"+str(min_value))
    print("中位数租房价格为"+str(median_value))
    print("平均租房价格为"+str(average_value))
    max_value = df["price_perArea"].max()
    arr[n][4]=max_value;
    min_value = df["price_perArea"].min()
    arr[n][5]=min_value;
    median_value = df["price_perArea"].median()
    arr[n][6]=median_value;
    average_value = df["price_perArea"].mean().round(2)
    arr[n][7]=average_value;
    print(name+"的最高每平米租房价格为"+str(max_value))
    print("最低每平米租房价格为"+str(min_value))
    print("中位数每平米租房价格为"+str(median_value))
```

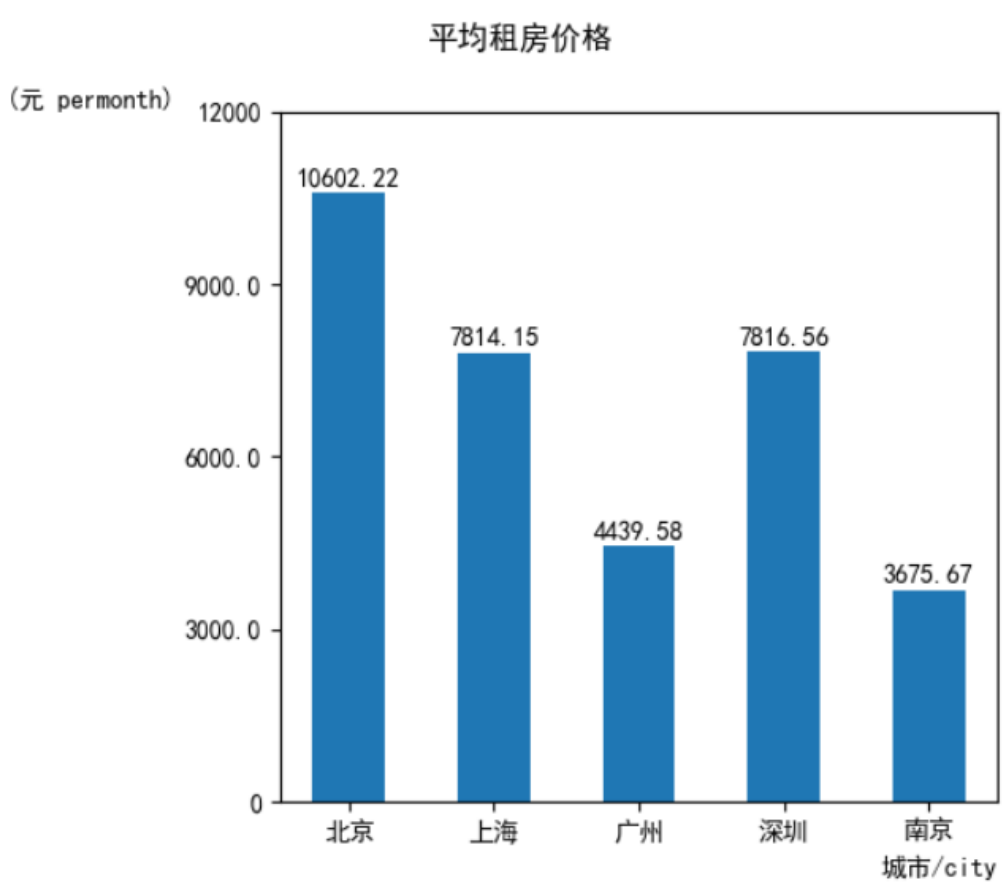
```
    print("平均每平米租房价格为"+str(average_value)+"\n")

# 导入 matplotlib 库
import matplotlib
from matplotlib import pyplot as plt
chart = {0: '最高租房价格', 1: '最低租房价格', 2: '中位数租房价格', 3: '平均租房价格', 4: '最高每平米租房价格', 5: '最低每平米租房价格', 6: '中位数每平米租房价格', 7: '平均每平米租房价格'}
for num, info in chart.items():
    # 定义横坐标的年份
    位置 = ['北京', '上海', '广州', '深圳', '南京']
    # 定义纵坐标的刻度值
    if(num==0) : max=400000
    elif(num==1) :max=1500
    elif(num==2) :max=10000
    elif(num==3) :max=12000
    elif(num==4) :max=5000
    elif(num==5) :max=10
    elif(num==6) :max=120
    elif(num==7) :max=120
    刻度 = [0, max/4, max/2, 3*max/4, max]
    # 设置中文字体
    plt.rcParams['font.sans-serif'] = ['SimHei']
    # 创建画布并设置属性
    画布 = plt.figure(num = '展示图', facecolor = 'white')
    # 设置 x 轴标签
    plt.xlabel('城市/city', loc = 'right')
    # 设置 y 轴标签
    plt.ylabel('(元 permonth)', loc = 'top', rotation = 0)
    # 设置 y 轴刻度标签
    plt.yticks(刻度, 刻度)
    # 设置 y 轴范围
    plt.ylim(0, max)
    # 调整子图的位置
    plt.subplots_adjust(left=0.3)
    # 设置主标题
    plt.suptitle(info)
    # 定义柱状图的数据
    价格 =[arr[0][num],arr[1][num],arr[2][num],arr[3][num],arr[4][num]]
    # 创建柱状图
    plt.bar(位置, 价格, width = 0.5, align ='center')
    # 为每个柱添加文本标签
    for 年, 数量 in zip(位置, 价格):
        plt.text(年, 数量+max/100, 数量, ha = 'center')
    # 显示图表
    plt.show()
```

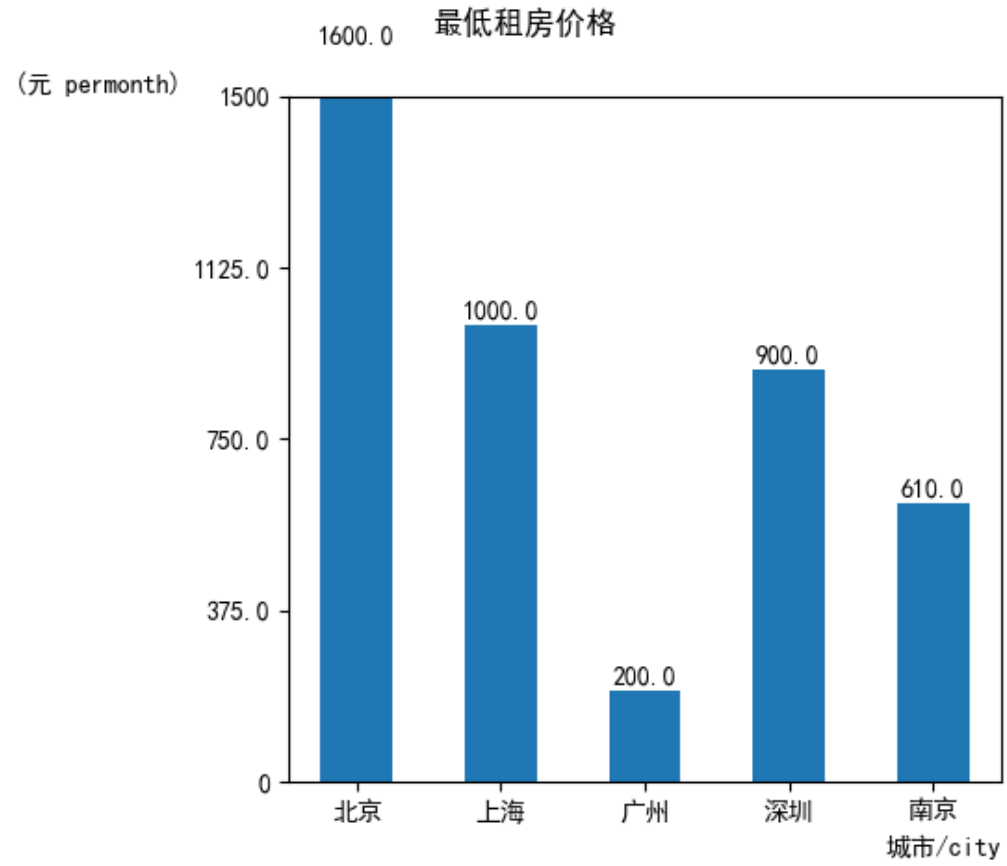
分析结果为:

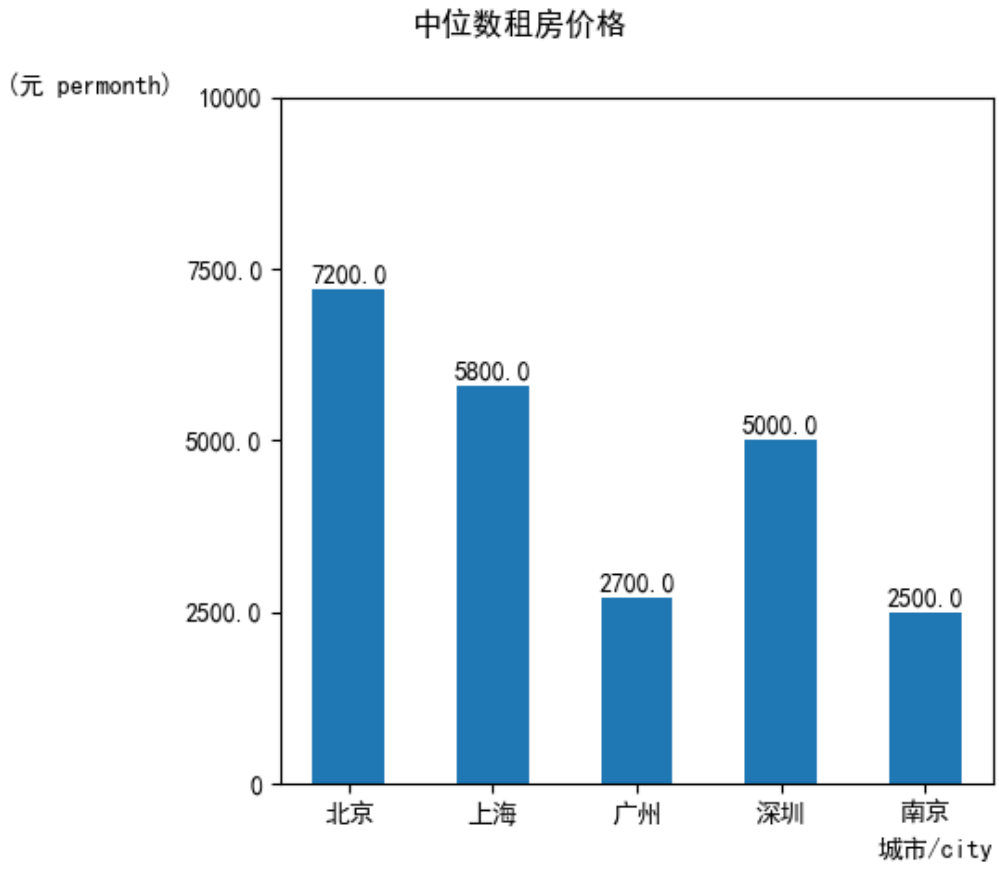
```
下面这段数据说明了什么:北京的最高租房价格为350000.0
最低租房价格为1600.0
中位数租房价格为7200.0
平均租房价格为10602.22
北京的最高每平米租房价格为979.2
```

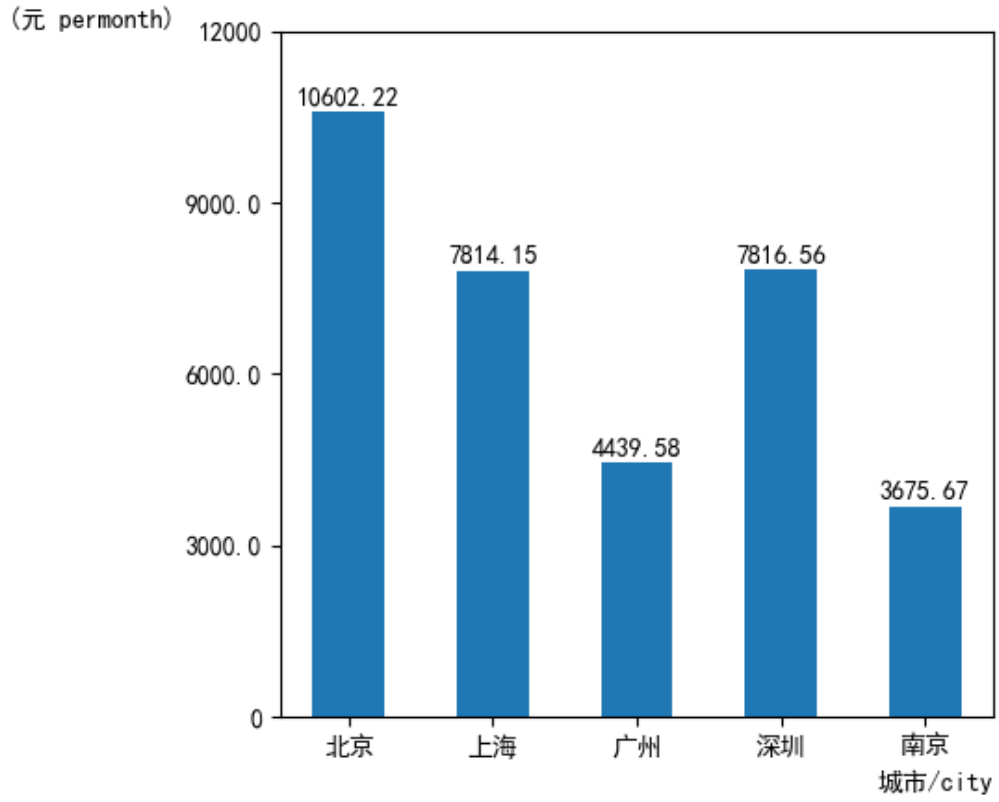
```
最低每平米租房价格为15.49
中位数每平米租房价格为103.02
平均每平米租房价格为107.4

上海的最高租房价格为90000.0
最低租房价格为1000.0
中位数租房价格为5800.0
平均租房价格为7814.15
上海的最高每平米租房价格为384.62
最低每平米租房价格为8.57
中位数每平米租房价格为95.33
平均每平米租房价格为103.0

广州的最高租房价格为90000.0
最低租房价格为200.0
中位数租房价格为2700.0
平均租房价格为4439.58
广州的最高每平米租房价格为2333.33
最低每平米租房价格为2.06
中位数每平米租房价格为40.82
平均每平米租房价格为51.47

深圳的最高租房价格为250000.0
最低租房价格为900.0
中位数租房价格为5000.0
平均租房价格为7816.56
深圳的最高每平米租房价格为4222.22
最低每平米租房价格为3.27
中位数每平米租房价格为80.0
平均每平米租房价格为91.6

南京的最高租房价格为170000.0
最低租房价格为610.0
中位数租房价格为2500.0
平均租房价格为3675.67
南京的最高每平米租房价格为292.13
最低每平米租房价格为0.39
中位数每平米租房价格为33.46
平均每平米租房价格为39.68
```

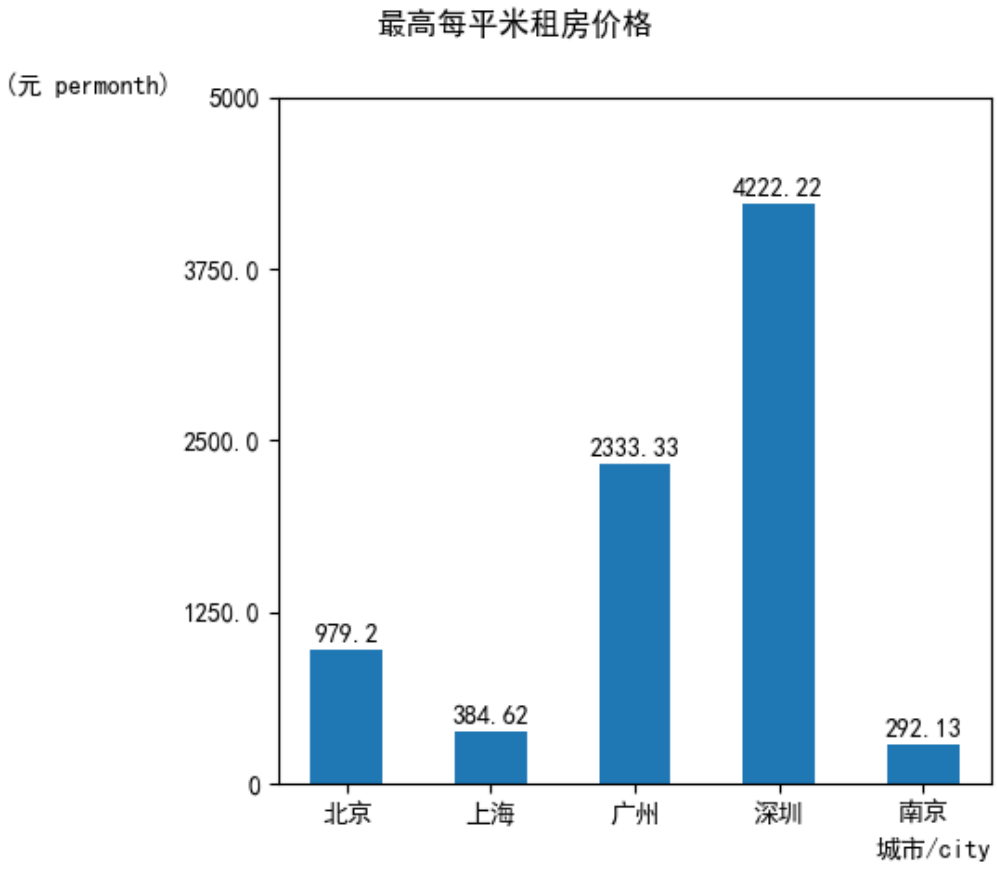
转化为图像为:

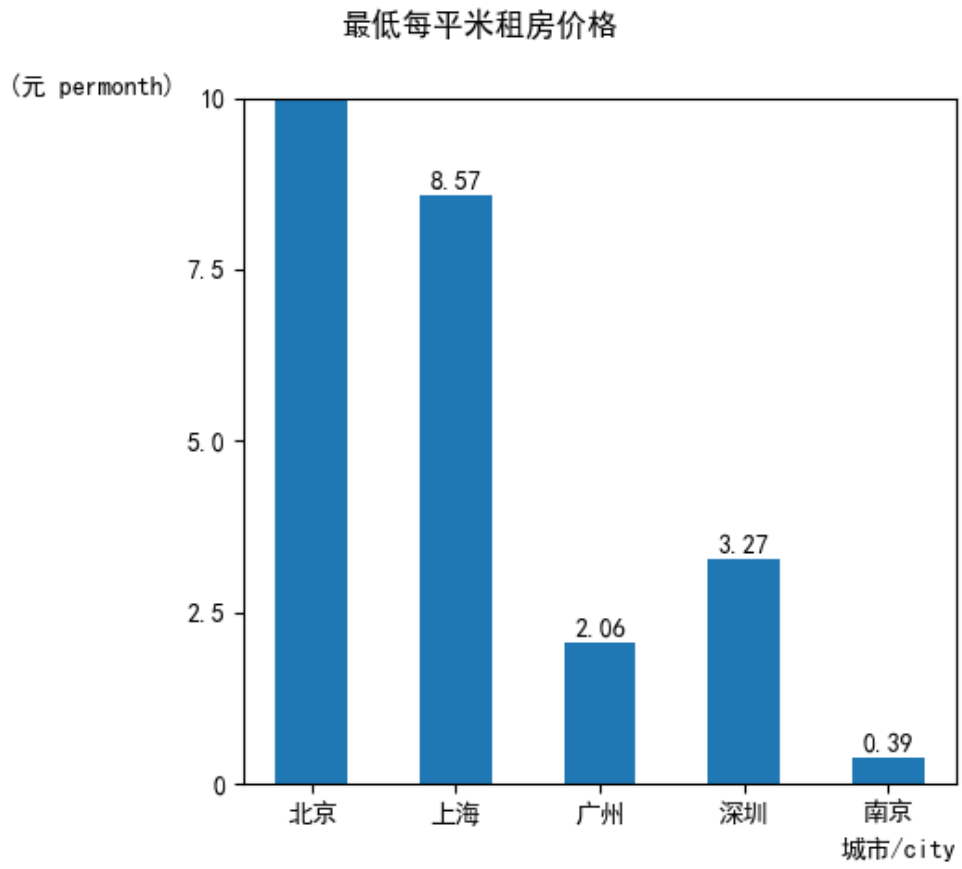

平均租房价格
（元 permonth）
12000
9000.0
6000.0
3000.0
0
10602.22
7814.15
4439.58
7816.56
3675.67
北京
上海
广州
深圳
南京
城市/city

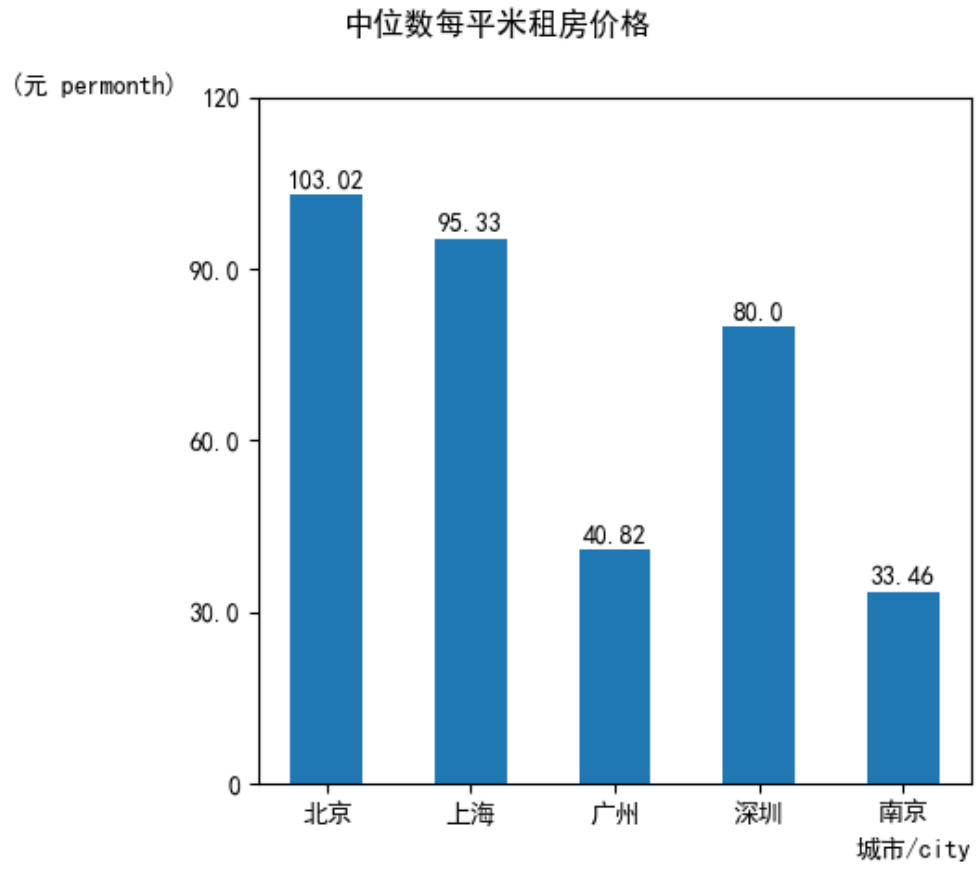

最低租房价格
（元 permonth）
1600.0
1500
1125.0
750.0
375.0
0
1000.0
900.0
200.0
610.0
北京
上海
广州
深圳
南京
城市/city

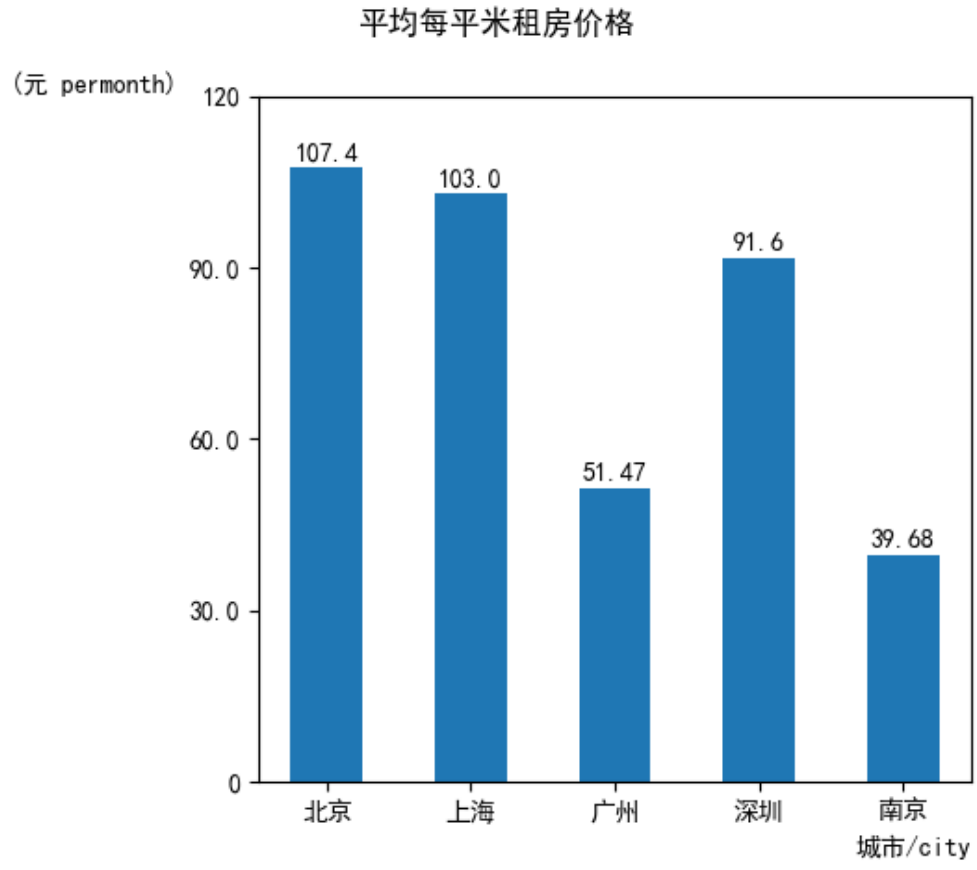

中位数租房价格
(元 permonth)
10000
7500.0
5000.0
2500.0
0
7200.0
5800.0
2700.0
5000.0
2500.0
北京
上海
广州
深圳
南京
城市/city


平均租房价格
(元 permonth)
12000
9000.0
6000.0
3000.0
0
10602.22
7814.15
4439.58
7816.56
3675.67
北京
上海
广州
深圳
南京
城市/city


最高每平米租房价格
（元 permonth）
5000
3750.0
2500.0
1250.0
0
979.2
384.62
2333.33
4222.22
292.13
北京
上海
广州
深圳
南京
城市/city


最低每平米租房价格
（元 permonth）
10
7.5
5.0
2.5
0
8.57
2.06
3.27
0.39
北京
上海
广州
深圳
南京
城市/city





2.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

下面这段代码的作用是读取一些包含房屋价格信息的 CSV 文件，然后对每个城市的每绍房屋数据进行统计分析。

首先，定义了一个名为 "cities" 的字典，用于保存每个城市的名称和相应的 CSV 文件名。然后，使用 for 循环遍历了这个字典，对于每个城市，它都会打开与城市对应的 CSV 文件，并使用 csv 模块中的 reader 函数创建一个 CSV 读取器，然后遍历这个读取器中的所有行，对于每一行，它都会提取第三列（居室信息）和第四列（价格信息），并将价格信息转换为浮点数。

接下来，使用另一个 for 循环遍历统计结果字典（保存了所有居室数据的列表），对于每一组数据，它会计算均价、最高值和最低值，并使用 pandas 模块计算中位数。最后，它会打印出这些统计信息。

```
# -*- coding: utf-8 -*-
import csv
import pandas as pd


# 创建结果字典，用于保存每组数据的均价、最高值、最低值
cities = {'北京': 'bj', '上海': 'sh', '广州': 'gz', '深圳': 'sz', '南京': 'nj'}


for name, info in cities.items():
    results = {}
    # 打开 CSV 文件
    with open(info+'data.csv', 'r',encoding='utf-8') as file:
        # 创建 CSV 读取器
        reader = csv.reader(file)
        # 遍历每一行
        for row in reader:
            if(row[0]=='title'):continue
            # 获取居室信息
            key = row[3]
            # 将第三列的值转换为数字
            value = float(row[4])

            # 如果字典中没有对应的键，则创建一个新的列表
            if key not in results:
                results[key] = []
            # 将值添加到列表中
            results[key].append(value)

    # 遍历每一组数据
    for key, values in results.items():
        if key == '1.0' or key=='1':
            han = '一'
        elif key == '2.0'or key=='2':
            han = '二'
        elif key == '3.0'or key=='3':
            han = '三'
        else: continue
        # 计算均价
        avg = sum(values) / len(values)
        # 计算最高值
        max_value = max(values)
        # 计算最低值
        min_value = min(values)
        #求中位数
        mid_value = pd.Series(values).median()
```

```
        # 打印统计结果
        print("在"+name+ han+"居室的", '平均价: ', avg, ' 最高价: ', max_value, ' 最低价: ', min_value, ' 中位数: ', mid_value)
    print("-------------------------------------------------")
```

分析结果为:

```
在北京二居室的 平均价:  8495.45786755573  最高价:  55000.0  最低价:  1600.0  中位数:  6900.0
在北京一居室的 平均价:  7939.031522025851  最高价:  73000.0  最低价:  1600.0  中位数:  6090.0
在北京三居室的 平均价:  12541.959866938512  最高价:  222000.0  最低价:  2400.0  中位数:  9000.0
-------------------------------------------------
在上海二居室的 平均价:  6469.609311740891  最高价:  35000.0  最低价:  1200.0  中位数:  5800.0
在上海三居室的 平均价:  12076.082125603865  最高价:  65000.0  最低价:  2000.0  中位数:  8800.0
在上海一居室的 平均价:  5183.99229199728  最高价:  25000.0  最低价:  1000.0  中位数:  4700.0
-------------------------------------------------
在广州一居室的 平均价:  3898.789470635932  最高价:  48000.0  最低价:  200.0  中位数:  2200.0
在广州二居室的 平均价:  2631.74772989929  最高价:  28000.0  最低价:  494.0  中位数:  2400.0
在广州三居室的 平均价:  3481.84892764081  最高价:  55000.0  最低价:  575.0  中位数:  3000.0
-------------------------------------------------
在深圳一居室的 平均价:  4721.163422131148  最高价:  180000.0  最低价:  900.0  中位数:  4000.0
在深圳二居室的 平均价:  5608.721237369261  最高价:  38000.0  最低价:  1090.0  中位数:  5000.0
在深圳三居室的 平均价:  6767.2096129441625  最高价:  70000.0  最低价:  1090.0  中位数:  5800.0
-------------------------------------------------
在南京一居室的 平均价:  4607.335868997155  最高价:  170000.0  最低价:  610.0  中位数:  2200.0
在南京三居室的 平均价:  3140.7952025240106  最高价:  26000.0  最低价:  610.0  中位数:  2400.0
在南京二居室的 平均价:  2802.10249886929  最高价:  82000.0  最低价:  610.0  中位数:  2700.0
```

3.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

下面这段代码的作用是读取一些包含房屋价格信息的 CSV 文件，然后对每个城市的每个板块的房屋数据进行统计分析。

首先，定义了一个名为 "cities" 的字典，用于保存每个城市的名称和相应的 CSV 文件名。然后，使用 for 循环遍历了这个字典，对于每个城市，它都会打开与城市对应的 CSV 文件，并使用 csv 模块中的 reader 函数创建一个 CSV 读取器，然后遍历这个读取器中的所有行，对于每一行，它都会提取第八列（板块信息）和第四列（价格信息），并将价格信息转换为浮点数。

接下来，使用另一个 for 循环遍历统计结果字典（保存了所有板块数据的列表），对于每一组数据，它会计算均价，并将结果保留两位小数。最后，它会将统计结果保存到 CSV 文件中。

```
# -*- coding: utf-8 -*-
import csv
import pandas as pd
from matplotlib import pyplot as plt

# 创建结果字典，用于保存每组数据的均价、最高值、最低值
cities = {'北京': 'bj', '上海': 'sh', '广州': 'gz', '深圳': 'sz', '南京': 'nj'}

for name, info in cities.items():
    results = {}
    # 打开 CSV 文件
    with open(info + 'data.csv', 'r', encoding='utf-8') as file:
        # 创建 CSV 读取器
        reader = csv.reader(file)
        # 遍历每一行
```

```
        for row in reader:
            if (row[0] == 'title'): continue
            # 获取板块信息
            key = row[8]
            # 将第三列的值转换为数字
            value = float(row[4])
            # 如果字典中没有对应的键，则创建一个新的列表
            if key not in results:
                results[key] = []
            # 将值添加到列表中
            results[key].append(value)
    final={}

    # 遍历每一组数据
    for key, values in results.items():
        if key == '': continue
        # 计算均价
        avg = sum(values) / len(values)
        # avg取二位小数
        avg = round(avg, 2)
        final[key] =[]
        final[key].append(avg)
        # 打印统计结果
        print(name + "在" + key + "板块的", '平均价: ', avg)
    print("-----------------------------------------------------")
    with open(name+'block-prize.csv', 'w', newline='') as output_file:
        writer = csv.writer(output_file)
        writer.writerow(['block', 'prize'])
        writer.writerows(final.items())
```

由于结果数量太多不便于用图表示,故用表格进行保存,以北京的板块价格信息为例子

block	prize
顺义城	[3752.55]
玉泉营	[6840.76]
石门营	[5492.37]
小西天	[7498.86]
广安门	[6152.39]
果园	[6186.67]
丽泽	[6550.0]
鲁谷	[26232.67]
西北旺	[10915.22]
梨园	[3937.5]
三元桥	[17023.28]
花乡	[9000.0]
十里河	[12938.46]
九棵树(家乐福)	[6528.37]
东关	[4044.44]
百子湾	[8632.32]
东直门	[6553.66]
长阳	[3509.45]
上地	[7531.25]
青塔	[6645.26]
亦庄开发区其它	[6803.53]
滨河西区	[2831.48]
北七家	[4961.41]
临河里	[5720.42]
清河	[10151.97]
四惠	[9749.09]
垡头	[5088.89]
通州其它	[6132.23]
科技园区	[13636.86]

block	prize
工体	[10941.03]
大兴其它	[5158.19]
西三旗	[5470.28]
宋家庄	[7723.53]
通州北苑	[7311.11]
中央别墅区	[29120.09]
北京南站	[6966.67]
车公庄	[26042.72]
定福庄	[6492.19]
北苑	[6990.69]
大红门	[8428.46]
望京	[15222.31]
海淀北部新区	[6892.57]
建国门外	[10278.26]
西关环岛	[3340.0]
后沙峪	[9799.01]
石佛营	[14784.62]
旧宫	[4431.65]
新宫	[12325.0]
天通苑	[7859.28]

4.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？

下面这段代码的作用是读取一些包含房屋价格信息的 CSV 文件，然后对每个城市的每个房屋的朝向进行统计分析。

首先，定义了一个名为 "cities" 的字典，用于保存每个城市的名称和相应的 CSV 文件名。然后，使用 for 循环遍历了这个字典，对于每个城市，它都会打开与城市对应的 CSV 文件，并使用 csv 模块中的 reader 函数创建一个 CSV 读取器，然后遍历这个读取器中的所有行，对于每一行，它都会提取第七列（朝向信息）和第五列（价格信息），并将价格信息转换为浮点数。

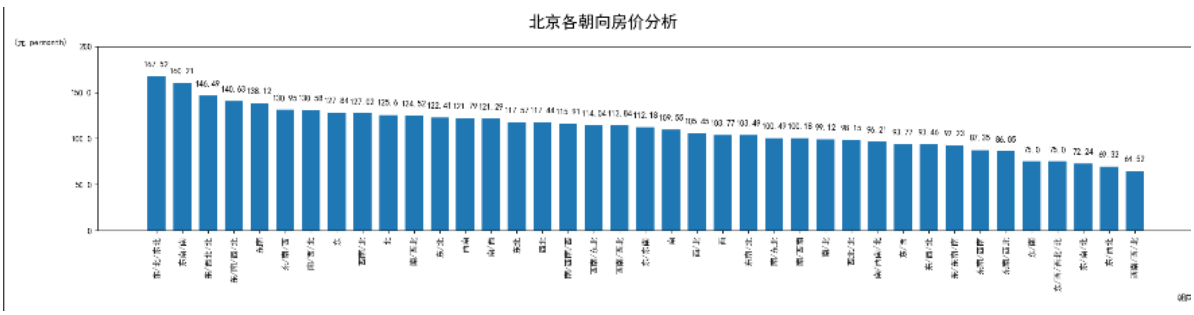
接下来，使用另一个 for 循环遍历统计结果字典（保存了所有朝向数据的列表），对于每一组数据，它会计算均价，并将结果保留两位小数。

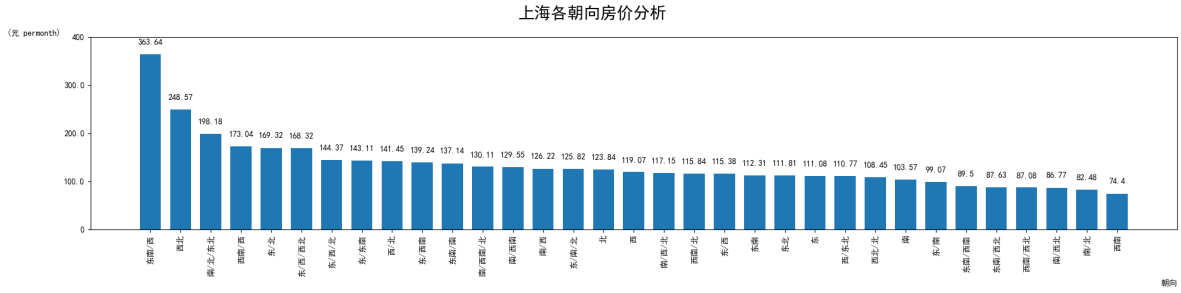
接下来，根据城市的不同，设置每个城市的 y 轴的最大值。然后，使用 matplotlib 模块的 pyplot 功能绘制柱状图。最后，将统计结果保存到 CSV 文件中。

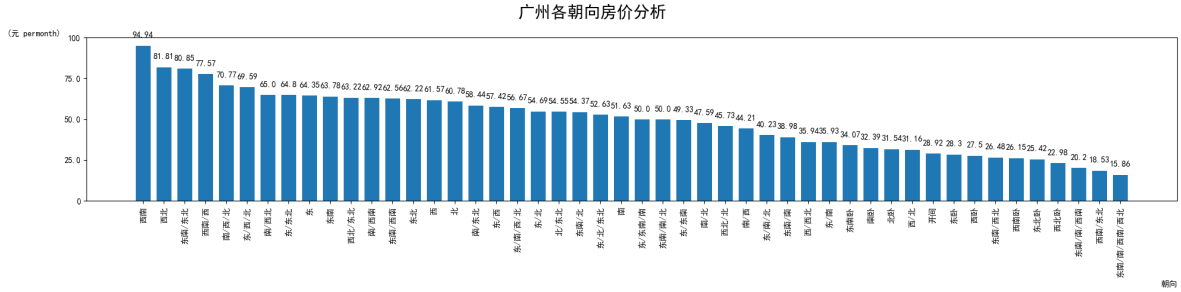
```
# -*- coding: utf-8 -*-
import csv
import pandas as pd
from matplotlib import pyplot as plt

# 创建结果字典，用于保存每组数据的均价、最高值、最低值
cities = {'北京': 'bj', '上海': 'sh', '广州': 'gz', '深圳': 'sz', '南京': 'nj'}

for name, info in cities.items():
    results = {}
    # 打开 CSV 文件
    with open(info + 'data.csv', 'r', encoding='utf-8') as file:
        # 创建 CSV 读取器
        reader = csv.reader(file)
        # 遍历每一行
        for row in reader:
            if (row[0] == 'title'): continue
            # 获取板块信息
            key = row[7]
            # 将第三列的值转换为数字
            value = float(row[5])
            # 如果字典中没有对应的键，则创建一个新的列表
            if key not in results:
                results[key] = []
            # 将值添加到列表中
            results[key].append(value)
    final={}

    # 遍历每一组数据
    for key, values in results.items():
        if key == '': continue
        # 计算均价
        avg = sum(values) / len(values)
        # avg取二位小数
        avg = round(avg, 2)
        final[key] =[]
        final[key].append(avg)
        # 打印统计结果
        print(name + "在" + key + "朝向的", '平均价: ', avg)
    print("-------------------------------------------------")
    if (info == 'bj'):
        max = 200
    elif (info == 'sh'):
        max = 400
    elif (info == 'gz'):
        max = 100
    elif (info == 'sz'):
        max = 300
    elif (info == 'nj'):
        max = 150
    刻度 = [0, max / 4, max / 2, 3 * max / 4, max]
    # 设置中文字体
    plt.rcParams['font.sans-serif'] = ['SimHei']
    # 创建画布并设置属性
    画布 = plt.figure(num='展示图', facecolor='white')
```

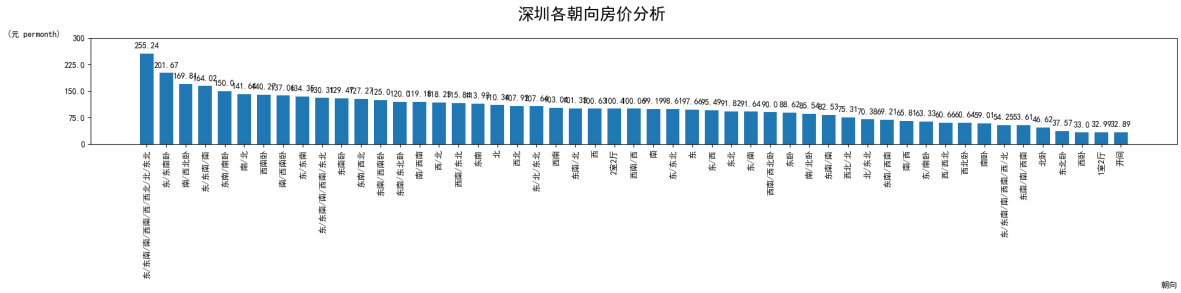
```
    # 设置 x 轴标签
    plt.xlabel('朝向', loc='right')
    # 设置 y 轴标签
    plt.ylabel('(元 permonth)', loc='top', rotation=0)
    # 设置 y 轴刻度标签
    plt.yticks(刻度, 刻度)
    # 设置 y 轴范围
    plt.ylim(0, max)
    # 调整子图的位置
    plt.subplots_adjust(left=0.3)
    # 设置主标题
    plt.suptitle(name + '各朝向房价分析', fontsize=20)
    #增大图表宽度
    画布.set_size_inches(20, 5)

    #柱状图下标竖向排列但字体为横向
    plt.xticks(rotation=90)

    # 定义柱状图的数据
    价格 = []
    朝向 = []
    #将final按value大小排序
    final = sorted(final.items(), key=lambda x: x[1], reverse=True)
    for key, values in final:
        价格.append(values[0])
        朝向.append(key)
    #柱上面显示数字竖向排列

#增大柱的间距

    # 创建柱状图
    plt.bar(朝向, 价格, width=0.7, align='center')

    # 为每个柱添加文本标签
    for 年, 数量 in zip(朝向, 价格):
        plt.text(年, 数量 + max / 20, 数量, ha='center')
    #防止文本标签遮挡
    plt.tight_layout()

    # 显示图表
    # 柱上面显示数字竖向排列

    plt.show()
```

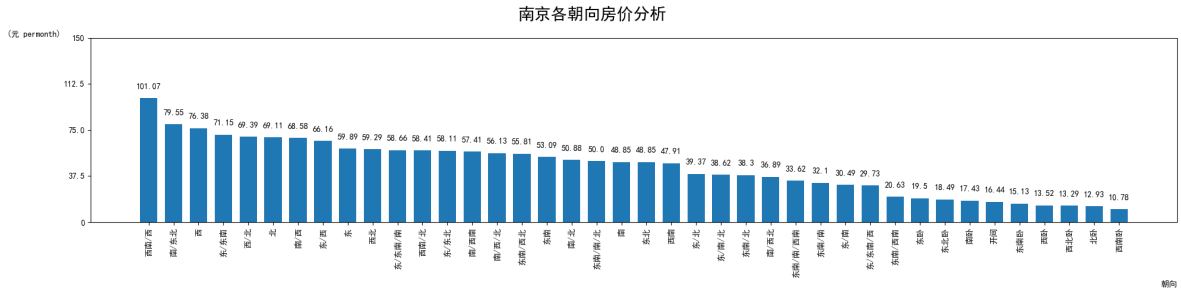
分析结果:











对于所有城市,都有向南的价格高于向北,因为房子朝南，采光是比朝北的房子好的。由于在我国，所在的部位坐落于北半球地图，太阳光全是对着南方地区的，因此朝南的房子，一般都是比朝北的房子更能触碰多一点的太阳光照射。且朝南的房子房型，一般生活阳台也会设计方案在南边，衣服裤子也会能迅速晾干好。房子朝南的话，自然通风一般也是比朝北的房子好的。朝南的房子，在夏季时处在迎风面，因而室内通风会比较好;可是在冬天的情况下，房子便会处在背风面的，防寒保暖实际效果便会更优。而朝北的房子恰好反过来，夏季时炎热极其，冬季则冰冷湿冷。

而东西的价格则不确定,可能是收到地形和东西发展程度的影响,比如浦东发展程度好于浦西,可能就导致了浦东的房价更高一些

5.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。相对而言，在哪个城市租房的性价比最高?

下面这段代码用于绘制租金与人均GDP对比图，首先通过 plt.subplots() 函数创建了一个新的画布，然后调用 ax1.plot() 和 ax2.plot() 函数分别在同一个画布中绘制了两条折线图。

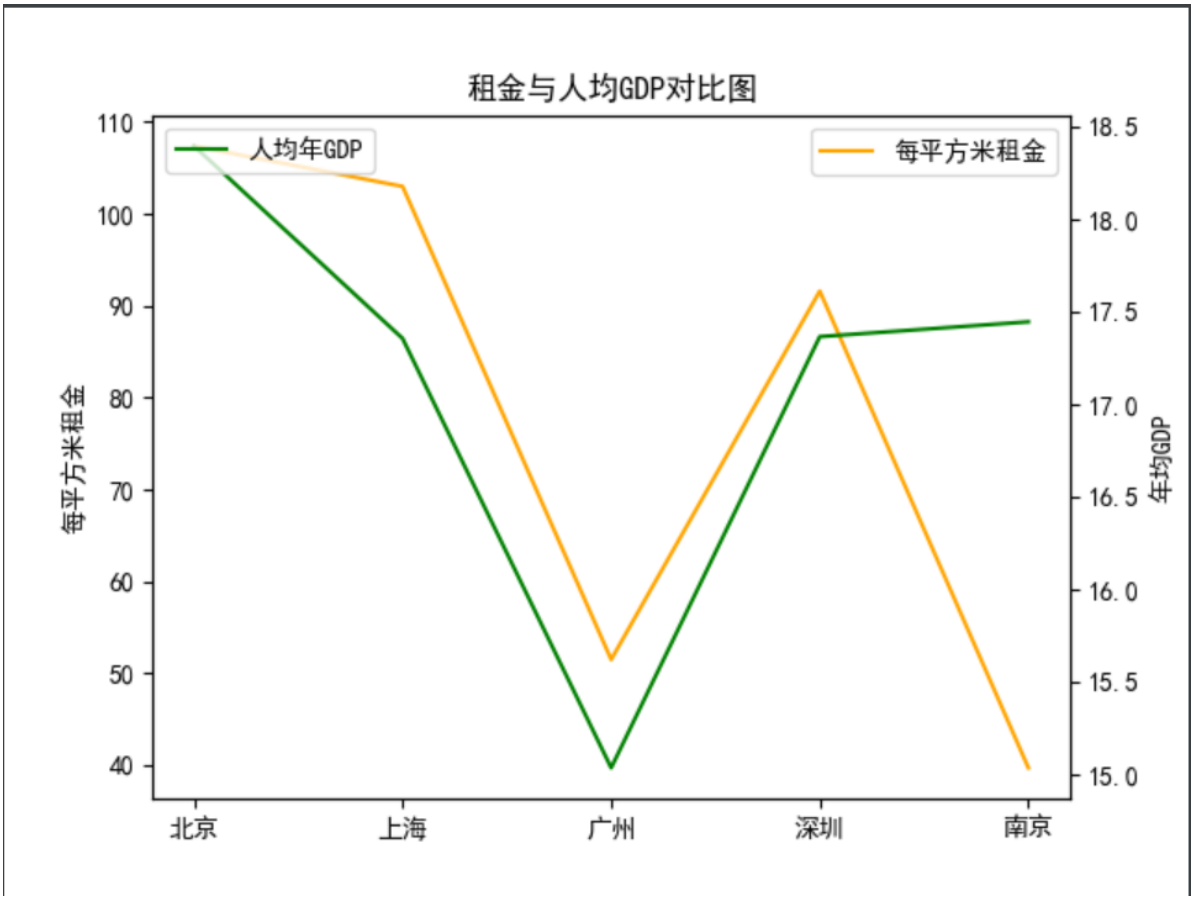
其中，x 和 y1 列表分别表示横坐标和纵坐标的数据，y2 列表表示第二条折线图的纵坐标数据。

通过调用 plt.xticks() 函数设置横坐标的刻度标签，通过调用 plt.title() 函数设置图表标题，通过调用 plt.xlabel() 和 ax1.set_ylabel() 函数分别设置横坐标和纵坐标的标签。

最后，通过调用 ax1.legend() 和 ax2.legend() 函数向图表中添加图例，并通过调用 plt.show() 函数显示图表。

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei'] # 解决中文显示问题
# 准备数据
x = [1, 2, 3, 4, 5]
y1 = [107.4,103.0,51.47,91.6,39.68]#每平方米租金
y2 = [18.4, 17.36, 15.04, 17.37, 17.45]#人均GDP
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
# 绘制折线图
ax1.plot(x, y1, color='orange',  label='每平方米租金')
ax2.plot(x, y2, color='green',  label='人均年GDP')

# 设置横坐标的刻度标签
plt.xticks(x, ['北京', '上海', '广州', '深圳', '南京'])

# 设置图表标题并给坐标轴加上标签
plt.title('租金与人均GDP对比图')
plt.xlabel('城市')
ax1.set_ylabel('每平方米租金')
ax2.set_ylabel('年均GDP')

# 显示图例
ax1.legend(loc=1)
ax2.legend(loc=2)



# 显示图表
plt.show()
```

分析结果:



大多数城市gdp折线和租金成正比,但是南京明显在人均gdp较高的情况下租金更少,所以在南京生活的性价比更高

6.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。相对而言，在哪个城市租房的负担最重?

下面这段代码使用 matplotlib 库在同一个图中绘制了两条折线图。分别对应五个城市的每平方米租金和人均收入数据。

首先，通过 plt.subplots() 函数创建了一个画布和一个子图，并设置了字体。然后，通过 ax1.twinx() 函数在同一个子图中创建了一个新的 y 轴。

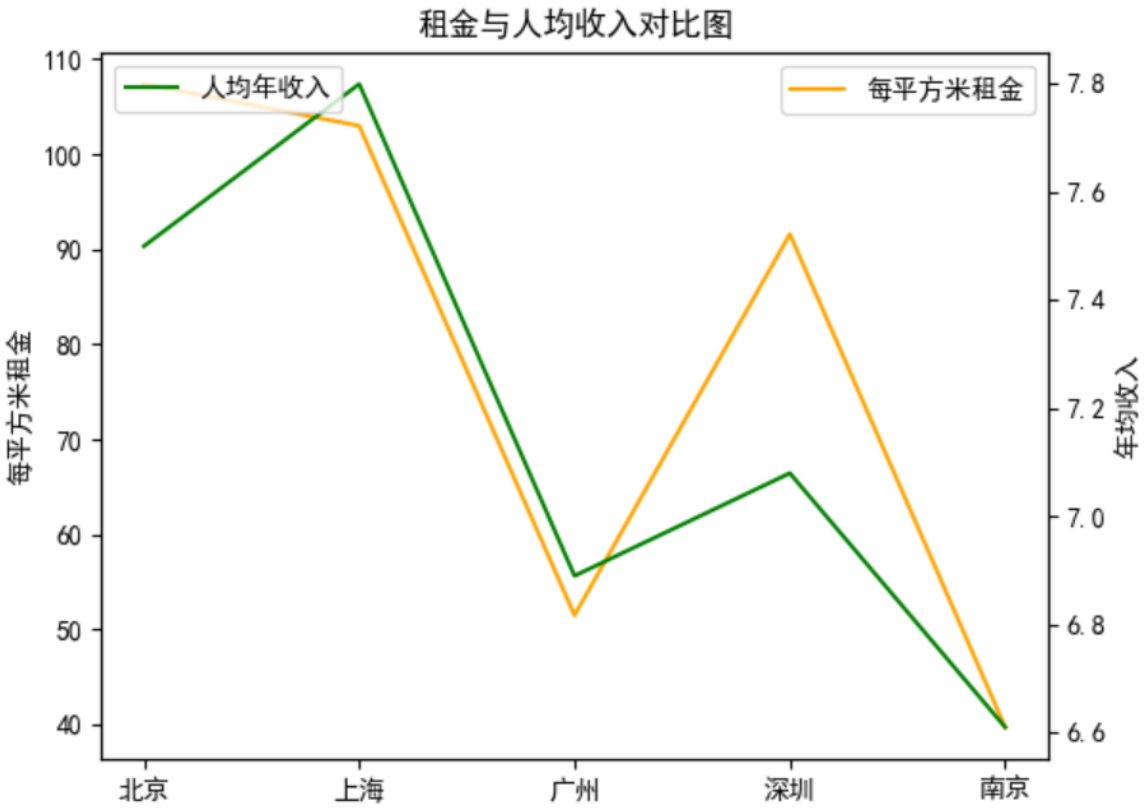
接着，使用 ax1.plot() 和 ax2.plot() 函数分别绘制了两条折线图。然后使用 plt.xticks() 函数设置横坐标的刻度标签，使用 plt.title()、plt.xlabel() 和 ax1.set_ylabel()、ax2.set_ylabel() 函数设置图表标题和坐标轴标签。

最后，使用 ax1.legend() 和 ax2.legend() 函数显示图例，并使用 plt.show() 函数显示图表。

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei'] # 解决中文显示问题
# 准备数据
x = [1, 2, 3, 4, 5]
y1 = [107.4,103.0,51.47,91.6,39.68]#每平方米租金
y2 = [7.5, 7.8, 6.89, 7.08, 6.61]#人均收入
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
# 绘制折线图
ax1.plot(x, y1, color='orange',  label='每平方米租金')
ax2.plot(x, y2, color='green',  label='人均年收入')
```

```
# 设置横坐标的刻度标签
plt.xticks(x, ['北京', '上海', '广州', '深圳', '南京'])

# 设置图表标题并给坐标轴加上标签
plt.title('租金与人均收入对比图')
plt.xlabel('城市')
ax1.set_ylabel('每平方米租金')
ax2.set_ylabel('年均收入')

# 显示图例
ax1.legend(loc=1)
ax2.legend(loc=2)



# 显示图表
plt.show()
```

分析结果:



大多数城市收入折线和租金成正比,但是深圳在人均年收入明显较低的情况下,每平方米租金更高,所以生活在深圳的压力更大

7.查询各个城市的人口密度，分析并展示其和居室单位面积的关系。

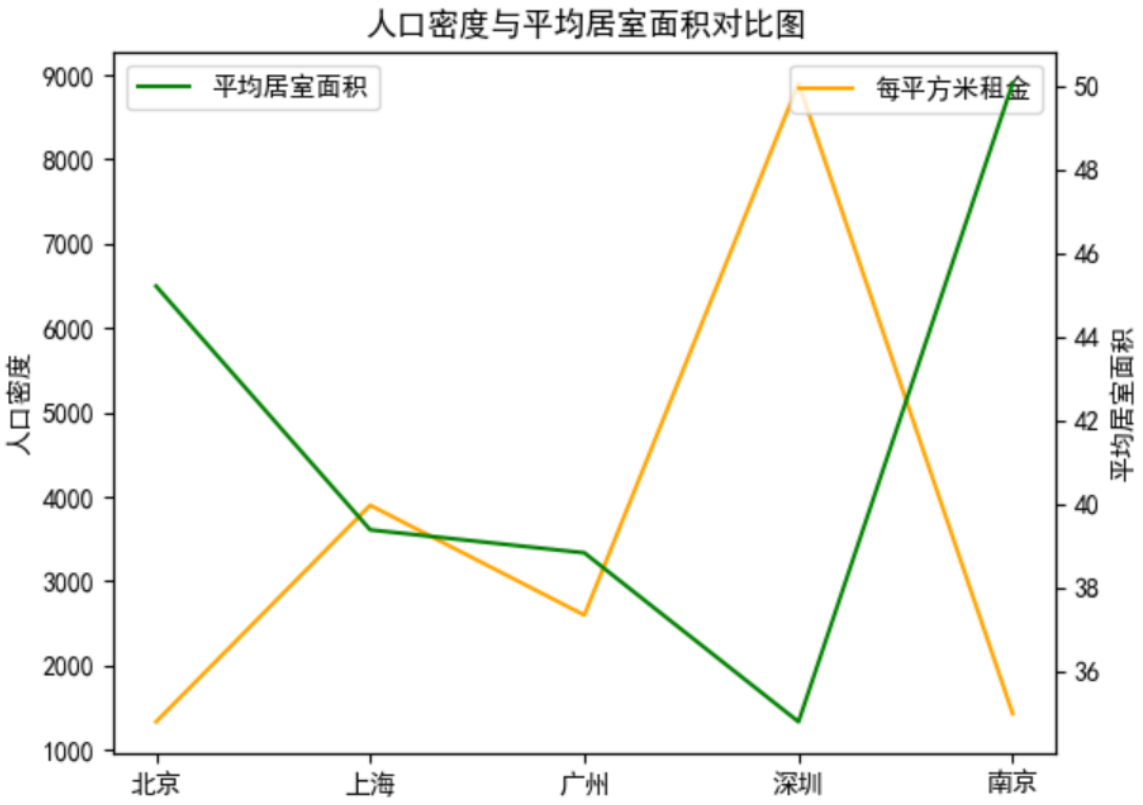
下面这段代码的作用是通过将五个城市的人口密度与平均居室面积作对比，生成一张双轴折线图，并显示出来。

首先，它会创建一个空列表 y2，用于保存平均居室面积的数据。然后，它会循环遍历五个城市，依次读取每个城市的 csv 文件，计算出每个城市的平均居室面积，并将其添加到 y2 列表中。

接下来，它会使用 matplotlib 库的函数创建一张双轴折线图，将五个城市的人口密度数据 y1 与平均居室面积数据 y2 绘制到图表中。然后，它会设置横坐标的刻度标签，图表标题和坐标轴标签，以及图例。最后，它会使用 plt.show() 函数显示图表。

```
# -*- coding: utf-8 -*-
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
y2 = []#平均居室面积
cities = {'北京': 'bj', '上海': 'sh', '广州': 'gz', '深圳': 'sz', '南京': 'nj'}
for name, info in cities.items():

    word=info+"data"+".csv"
    df = pd.read_csv(word)
    average_value = df["area_perRoom"].mean().round(2)
    y2.append(average_value)
plt.rcParams['font.sans-serif'] = ['SimHei'] # 解决中文显示问题
# 准备数据
x = [1, 2, 3, 4, 5]
y1 = [1334,3900,2598,8901,1430]#人口密度
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
# 绘制折线图
ax1.plot(x, y1, color='orange',  label='每平方米租金')
ax2.plot(x, y2, color='green',  label='平均居室面积')

# 设置横坐标的刻度标签
plt.xticks(x, ['北京', '上海', '广州', '深圳', '南京'])

# 设置图表标题并给坐标轴加上标签
plt.title('人口密度与平均居室面积对比图')
plt.xlabel('城市')
ax1.set_ylabel('人口密度')
ax2.set_ylabel('平均居室面积')

# 显示图例
ax1.legend(loc=1)
ax2.legend(loc=2)
# 显示图表
plt.show()
```

分析结果:



可以看出,人口密度和平均居室面积基本为反比,人越多,一件屋子可以住的地方越少,因为要留出更多空间容纳更多的人.

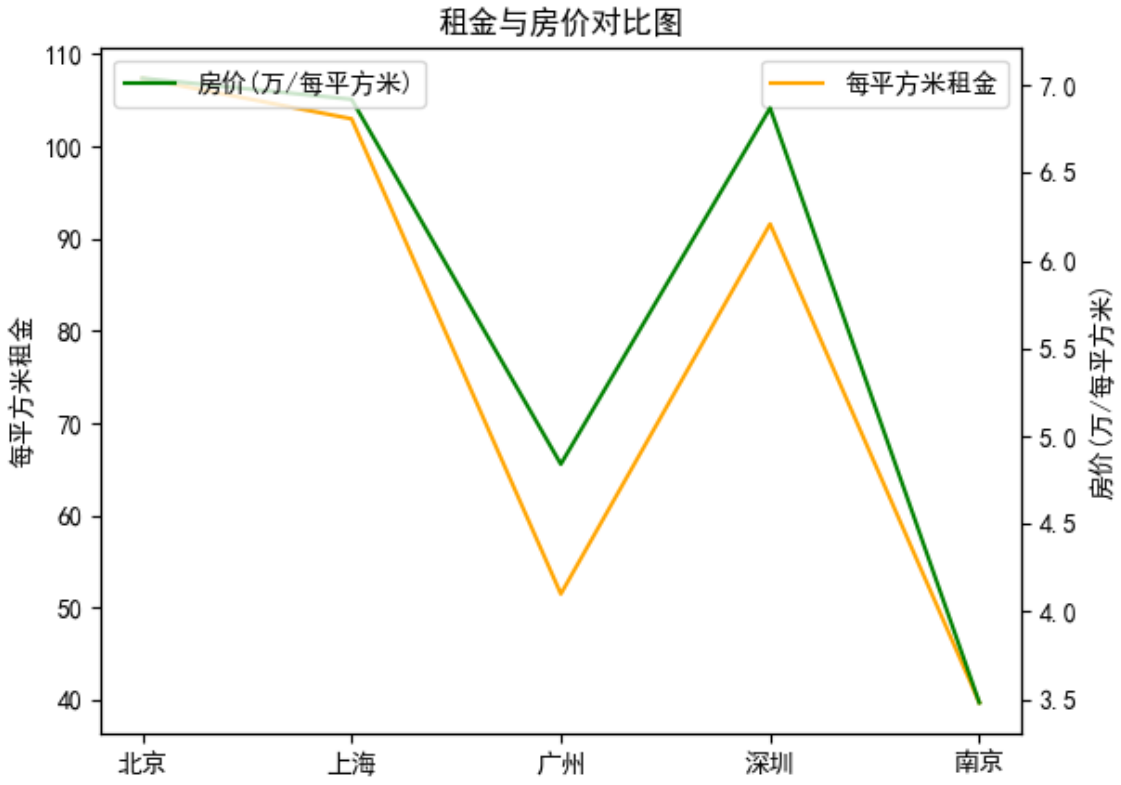
8.查询各个城市的房价，分析并展示其和房租的关系。

下面这段代码使用了 Matplotlib 库来绘制一张租金和房价对比图。首先，通过 `plt.rcParams['font.sans-serif'] = ['SimHei']` 解决了中文显示的问题。然后，定义了两组数据 `y1` 和 `y2` 分别表示每平方米的租金和房价。接着，使用 `plt.subplots()` 函数创建了一张图，并通过 `ax1.twinx()` 创建了第二个坐标轴。接下来，使用 `ax1.plot()` 和 `ax2.plot()` 分别在两个坐标轴上绘制折线图，并使用 `plt.xticks()` 函数设置了横坐标的刻度标签。最后，使用 `plt.title()`、`plt.xlabel()` 和 `ax1.set_ylabel()`、`ax2.set_ylabel()` 分别设置了图表标题、横坐标轴标签和两个坐标轴的纵坐标轴标签，使用 `ax1.legend()` 和 `ax2.legend()` 分别显示了两个图例，最后使用 `plt.show()` 函数显示图表。

```
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei'] # 解决中文显示问题
# 准备数据
x = [1, 2, 3, 4, 5]
y1 = [107.4,103.0,51.47,91.6,39.68]#每平方米租金
y2 = [7.04, 6.92, 4.84, 6.87, 3.48]#房价
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
# 绘制折线图
ax1.plot(x, y1, color='orange',  label='每平方米租金')
ax2.plot(x, y2, color='green',  label='房价(万/每平方米)')

# 设置横坐标的刻度标签
plt.xticks(x, ['北京', '上海', '广州', '深圳', '南京'])
```

```
# 设置图表标题并给坐标轴加上标签
plt.title('租金与房价对比图')
plt.xlabel('城市')
ax1.set_ylabel('每平方米租金')
ax2.set_ylabel('房价(万/每平方米)')

# 显示图例
ax1.legend(loc=1)
ax2.legend(loc=2)

# 显示图表
plt.show()
```

分析结果:



可以看出房价走势和房租走势基本成正比,可以看出影响房租的决定因素始终是房价,而中国的房租售比相比全世界比较低

实验总结

进行数据分析的作业是一个有趣且有意义的体验。我们可以使用Python的强大功能来获取、清洗、分析和可视化数据。这不仅能够帮助我们更好地理解数据，还能为我们提供有价值的见解和结论。

在做数据分析作业的过程中，我们需要我们需要注意以下几点：

1. 数据清洗：首先要对数据进行清洗，去除无效数据和空值，确保数据的准确性和完整性。
2. 数据可视化：通过图表来展示数据，更直观地了解数据的分布和趋势。
3. 发现规律：对数据进行统计分析，发现数据之间的关系和规律。
4. 得出结论：根据分析的结果，得出结论并进行预测。
5. 可视化展示结果：将分析的结果展示出来，方便理解和沟通。

这需要我们学习和使用一些Python的第三方库，例如Pandas、Numpy、Matplotlib等。学习这些库需要一些时间和练习，但是在数据分析的过程中使用这些库能够大大简化我们的工作。总之这次作业让我收获很多,我有兴趣用所学的知识跟进一步的去进行更复杂更有用的数据分析.