# Python期末大作业

班级: 2021211308

学号: 2021211391

姓名: 吴显科

## 1. 城市列表与数据情况

北京, 上海, 广州, 深圳, 成都

表头: 名称, 区域, 板块, 房型, 朝向, 面积(平米), 价格(元/月), 单位面积价格(元/平米/月)

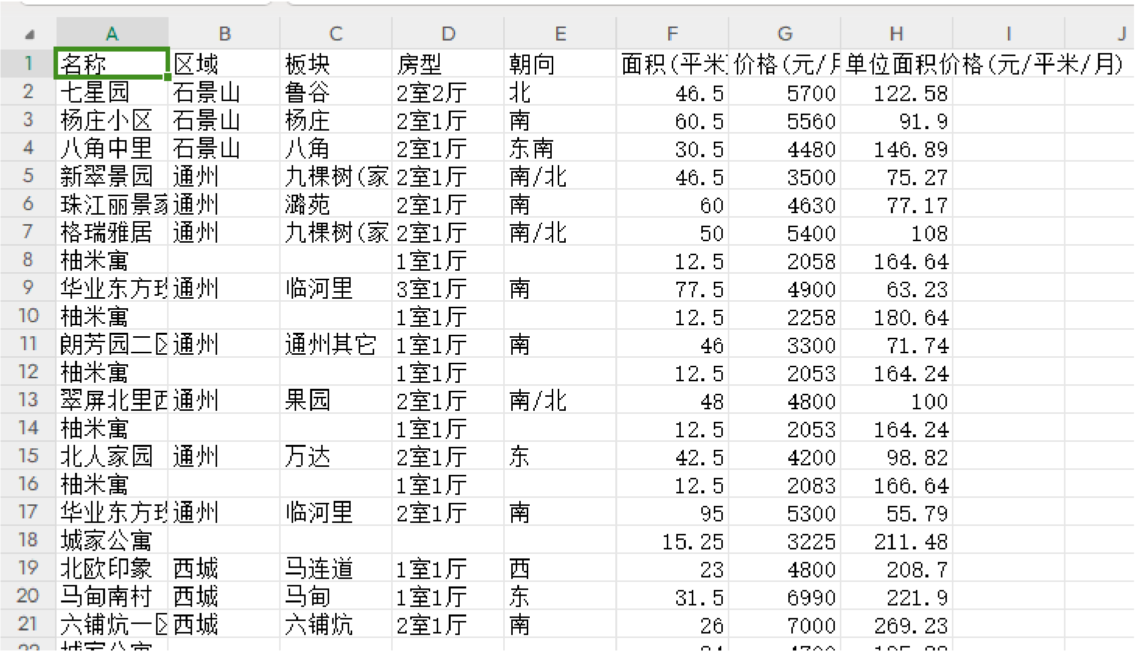
data/bj.csv 36621pieces

data/sh.csv 30003 pieces

data/gz.csv 30538 pieces

data/sz.csv 23647 pieces

data/cd.csv 33539 pieces



| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 名称 | 区域 | 板块 | 房型 | 朝向 | 面积(平米 | 价格(元/月 | 单位面积价格(元/平米/月) | | |
| 2 | 七星园 | 石景山 | 鲁谷 | 2室2厅 | 北 | 46.5 | 5700 | 122.58 | | |
| 3 | 杨庄小区 | 石景山 | 杨庄 | 2室1厅 | 南 | 60.5 | 5560 | 91.9 | | |
| 4 | 八角中里 | 石景山 | 八角 | 2室1厅 | 东南 | 30.5 | 4480 | 146.89 | | |
| 5 | 新翠景园 | 通州 | 九棵树(家 | 2室1厅 | 南/北 | 46.5 | 3500 | 75.27 | | |
| 6 | 珠江丽景家 | 通州 | 潞苑 | 2室1厅 | 南 | 60 | 4630 | 77.17 | | |
| 7 | 格瑞雅居 | 通州 | 九棵树(家 | 2室1厅 | 南/北 | 50 | 5400 | 108 | | |
| 8 | 柚米寓 | | | 1室1厅 | | 12.5 | 2058 | 164.64 | | |
| 9 | 华业东方玫 | 通州 | 临河里 | 3室1厅 | 南 | 77.5 | 4900 | 63.23 | | |
| 10 | 柚米寓 | | | 1室1厅 | | 12.5 | 2258 | 180.64 | | |
| 11 | 朗芳园二区 | 通州 | 通州其它 | 1室1厅 | 南 | 46 | 3300 | 71.74 | | |
| 12 | 柚米寓 | | | 1室1厅 | | 12.5 | 2053 | 164.24 | | |
| 13 | 翠屏北里西 | 通州 | 果园 | 2室1厅 | 南/北 | 48 | 4800 | 100 | | |
| 14 | 柚米寓 | | | 1室1厅 | | 12.5 | 2053 | 164.24 | | |
| 15 | 北人家园 | 通州 | 万达 | 2室1厅 | 东 | 42.5 | 4200 | 98.82 | | |
| 16 | 柚米寓 | | | 1室1厅 | | 12.5 | 2083 | 166.64 | | |
| 17 | 华业东方玫 | 通州 | 临河里 | 2室1厅 | 南 | 95 | 5300 | 55.79 | | |
| 18 | 城家公寓 | | | | | 15.25 | 3225 | 211.48 | | |
| 19 | 北欧印象 | 西城 | 马连道 | 1室1厅 | 西 | 23 | 4800 | 208.7 | | |
| 20 | 马甸南村 | 西城 | 马甸 | 1室1厅 | 东 | 31.5 | 6990 | 221.9 | | |
| 21 | 六铺炕一区 | 西城 | 六铺炕 | 2室1厅 | 南 | 26 | 7000 | 269.23 | | |

## 2. 爬虫设计:

1. 使用了面向对象设计, 设计基类BasicSpider, 通过继承分别得到爬虫bj/sh/gz/sz/cd
2. 在BasicSpider类中, 覆写了parse方法, 通过css selector获取".content__list--item"中的元素(包含了单一房源的全部信息)后, 调用parse_item方法进行后续处理. 处理结束后, 使用index更新url, 达到爬取连续爬取网页的效果

```python
def parse(self, response):
    for each in response.css(".content__list--item"):
        item = self.parse_item(each)
        yield item

    location = response.url.split("/")[4]
    idx = self.index[location]
    if idx <= 100:
        time.sleep(1)
        next_page = self.prefix + location + "/pg" + str(idx) + "/" +
"#contentList"
        self.index[location] += 1
        yield scrapy.Request(next_page, callback=self.parse)
    else:
        return None
```

3. 覆写了方法parse_item, 进行下列处理

名称: 舍去前三个字符

房型: 匹配含有"室"或"房间"的数据, 否则置空

朝向: 限制字数在[1,3]之间, 并要求只包含"东南西北"四种字符

区域&板块: 通过进一步使用css selector获取("p.content__list--item--des a::text")中的数据, 按序将其分离

面积: 筛选包含㎡的数据, 否则置空

价格&单位面积价格: 正常使用css selector获取

```
def parse_item(self, selector):
    item = BasicItem()
    info = selector.css(
        "a.content__list--item--aside[title]::attr(title)"
    ).extract_first()
    info_list = info.split(" ")
    item["名称"] = info_list[0][3:]

    typeList = [s for s in info_list if "室" in s or "房间" in s]
    item["房型"] = typeList[0] if len(typeList) > 0 else ""

    orientationList = [
        s
        for s in info_list
        if len(s) in range(1, 4) and ("东" in s or "南" in s or "西" in s
or "北" in s)
    ]
    item["朝向"] = orientationList[0] if len(orientationList) > 0 else ""

    info = selector.css("p.content__list--item--des a::text").getall()
    if len(info) > 2:
        item["区域"] = info[0]
        item["板块"] = info[1]
    else:
        item["区域"] = ""
        item["板块"] = ""

    info = selector.css("p.content__list--item--des ::text").getall()
    areaList = [s for s in info if "㎡" in s]
    item["面积"] = areaList[0].strip() if len(areaList) > 0 else ""

    item["价格"] = selector.css("span.content__list--item-price
em::text").get()
    item["单位面积价格"] = ""
    return item
```

4. 在具体的城市爬虫中, 继承basicSpider, 并按需设置start_urls(按区抓取以突破一百页限制), 并设置prefix, index等自动抓取所需参数, 这里用北京举例:

```
class BjSpider(basicSpider.BasicSpider):
    name = "bj"
    allowed_domains = ["bj.lianjia.com"]
    start_urls = [
        "https://bj.lianjia.com/zufang/dongcheng/",
        "https://bj.lianjia.com/zufang/xicheng/",
        "https://bj.lianjia.com/zufang/chaoyang/",
        "https://bj.lianjia.com/zufang/haidian/",
        "https://bj.lianjia.com/zufang/fengtai/",
        "https://bj.lianjia.com/zufang/shijingshan/",
        "https://bj.lianjia.com/zufang/tongzhou/",
        "https://bj.lianjia.com/zufang/changping/",
        "https://bj.lianjia.com/zufang/daxing/",
        "https://bj.lianjia.com/zufang/yizhuangkaifaqu/",
        "https://bj.lianjia.com/zufang/shunyi/",
        "https://bj.lianjia.com/zufang/fangshan/",
        "https://bj.lianjia.com/zufang/mentougou/",
        "https://bj.lianjia.com/zufang/pinggu/",
        "https://bj.lianjia.com/zufang/huairou/",
        "https://bj.lianjia.com/zufang/miyun/",
        "https://bj.lianjia.com/zufang/yanqing/",
    ]
    index = {
        "dongcheng": 2,
        "xicheng": 2,
        "chaoyang": 2,
        "haidian": 2,
        "fengtai": 2,
        "shijingshan": 2,
        "tongzhou": 2,
        "changping": 2,
        "daxing": 2,
        "yizhuangkaifaqu": 2,
        "shunyi": 2,
        "fangshan": 2,
        "mentougou": 2,
        "pinggu": 2,
        "huairou": 2,
        "miyun": 2,
        "yanqing": 2,
    }
    prefix = "https://bj.lianjia.com/zufang/"
```

## 3. 总体房租对比

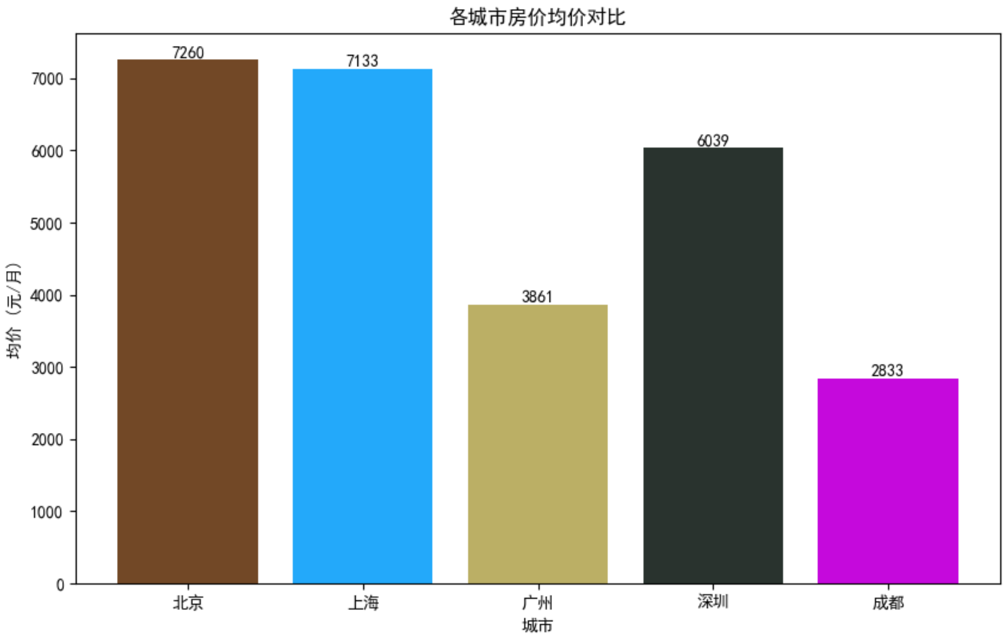
1. data_analyse/overall/overall_DataAnalyzer.py: 从数据中获取数据, 并计算出租金的均价、最高价、最低价、中位数, 以及单位面积租金（元/平米）的均价、最高价、最低价、中位数

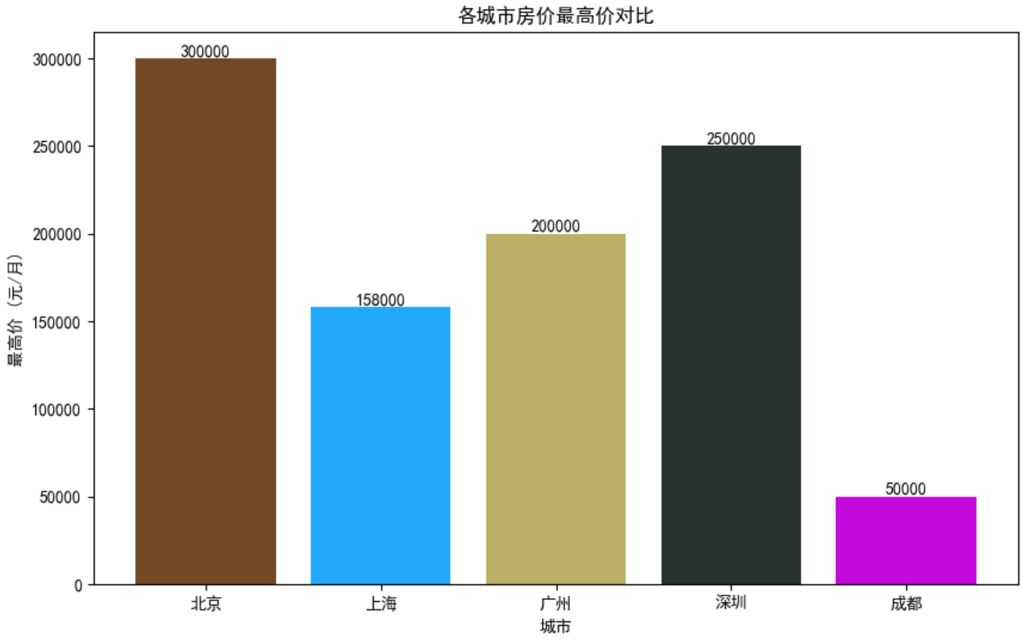
```
    def calculate_statistics(self):
        # Calculating statistics for 价格(元/月) and 单位面积价格(元/平米/月)
        price_stats = self.data["价格(元/月)"].describe(percentiles=[0.5])
        unit_price_stats = self.data["单位面积价格(元/平米/月)"].describe(percentiles=[0.5])

        # Extracting required statistics
        self.price_summary = {
            "价格均价": int(price_stats["mean"]),
            "价格最高价": int(price_stats["max"]),
            "价格最低价": int(price_stats["min"]),
            "价格中位数": int(price_stats["50%"]),
        }

        self.unit_price_summary = {
            "单位面积价格均价": round(unit_price_stats["mean"], 2),
            "单位面积价格最高价": round(unit_price_stats["max"], 2),
            "单位面积价格最低价": round(unit_price_stats["min"], 2),
            "单位面积价格中位数": round(unit_price_stats["50%"], 2),
        }
```

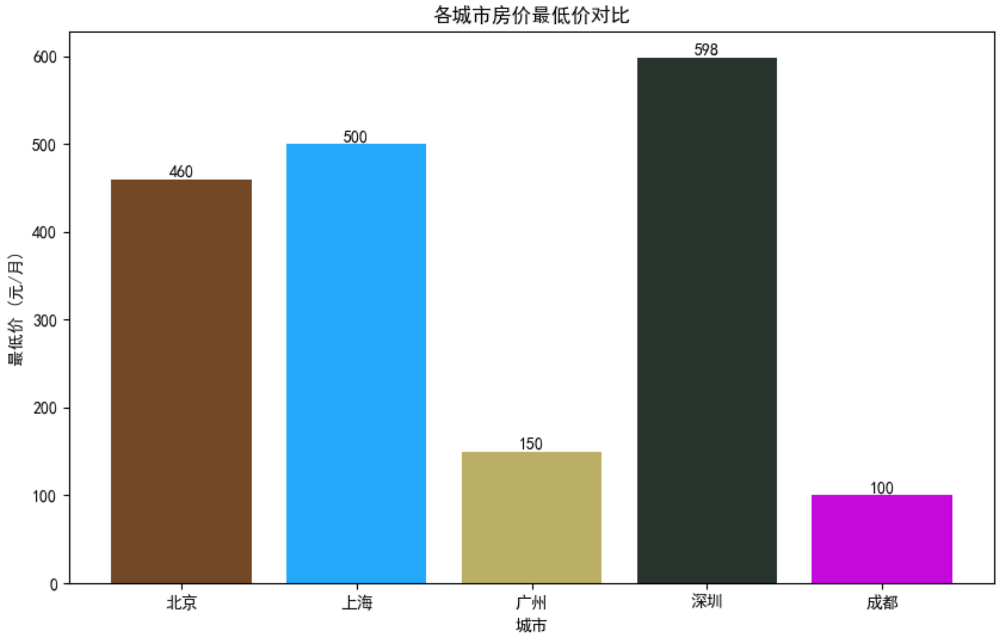
2. data_analyse/overall/overall_visulization.py: 分别调用上述代码进行计算五个城市的数据, 以租金/单位面积租金为区分, 分别作四个柱状图, 分别展示五个成熟的均价/最高价/最低价/中位数信息

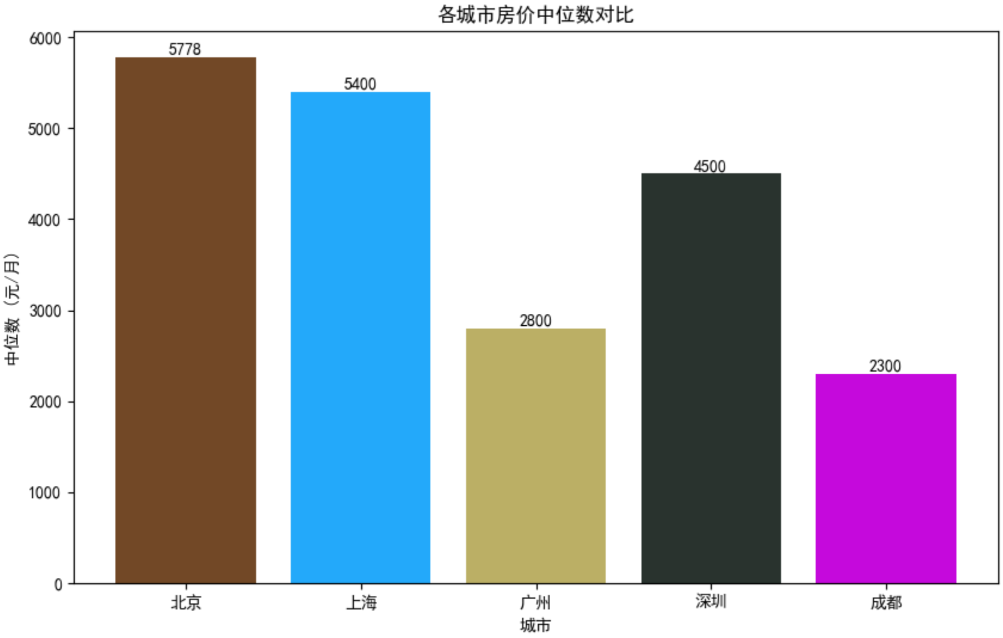
```
unit_price_summaries = [
    BJ.get_unit_price_summary(),
    SH.get_unit_price_summary(),
    GZ.get_unit_price_summary(),
    SZ.get_unit_price_summary(),
    CD.get_unit_price_summary(),
]

# 提取绘图所需的值
means = [summary["单位面积价格均价"] for summary in unit_price_summaries]
maxes = [summary["单位面积价格最高价"] for summary in unit_price_summaries]
mins = [summary["单位面积价格最低价"] for summary in unit_price_summaries]
medians = [summary["单位面积价格中位数"] for summary in unit_price_summaries]
categories = ["北京", "上海", "广州", "深圳", "成都"]
colors = [
    "#" + "".join([random.choice("0123456789ABCDEF") for j in range(6)])
    for i in range(len(categories))
]
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False

# 绘制均价条形图
plt.figure(figsize=(10, 6))
bars = plt.bar(categories, means, color=colors)
for bar in bars:
    yval = bar.get_height()
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        yval,
        round(yval, 2),
          va="bottom",
          ha="center",
    )
plt.xlabel("城市")
plt.ylabel("均价 (元/平米/月)")
plt.title("各城市单位面积房价均价对比")
plt.show()
```

## 总体房租可视化图:

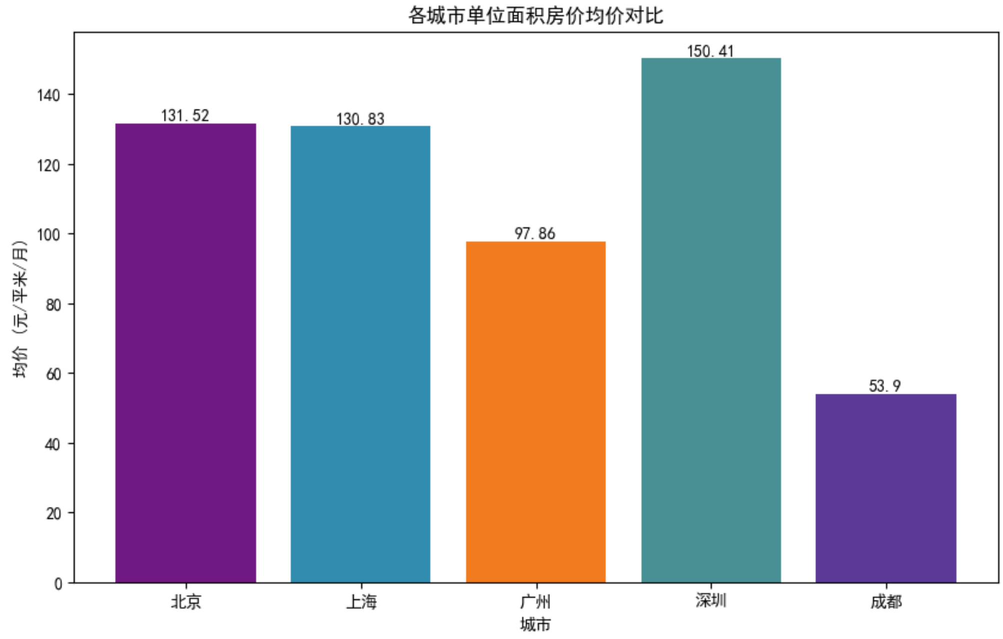


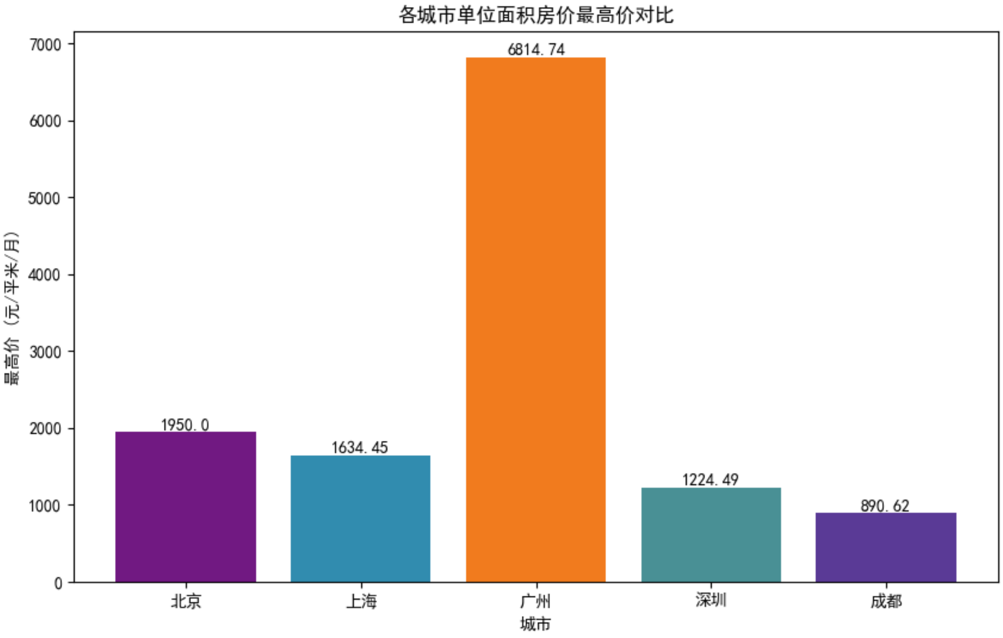


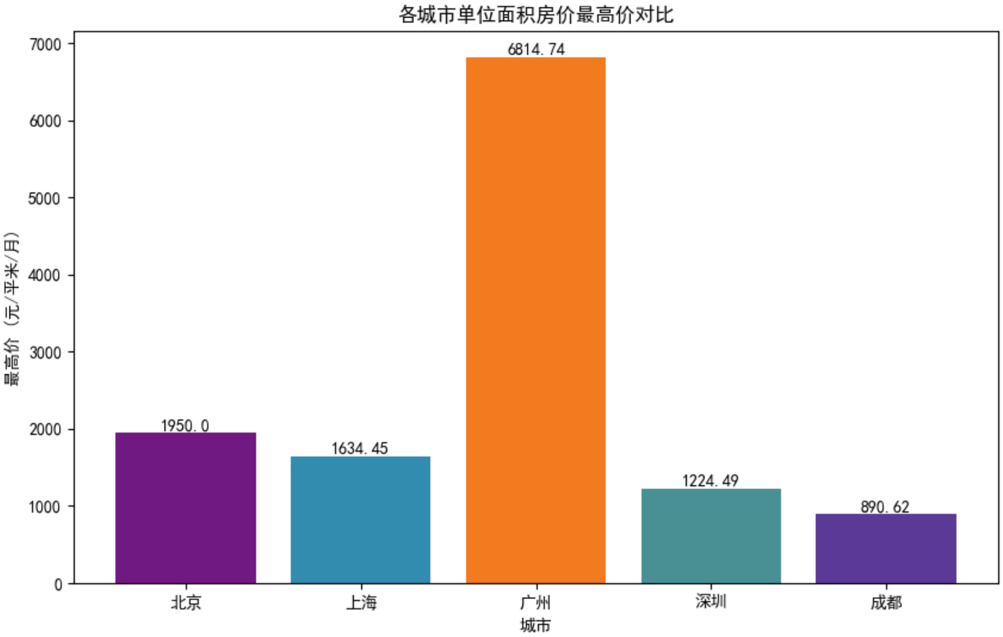

各城市房价最低价对比
最低价（元/月）
0
100
200
300
400
500
600
460
500
150
598
100
北京
上海
广州
深圳
成都
城市

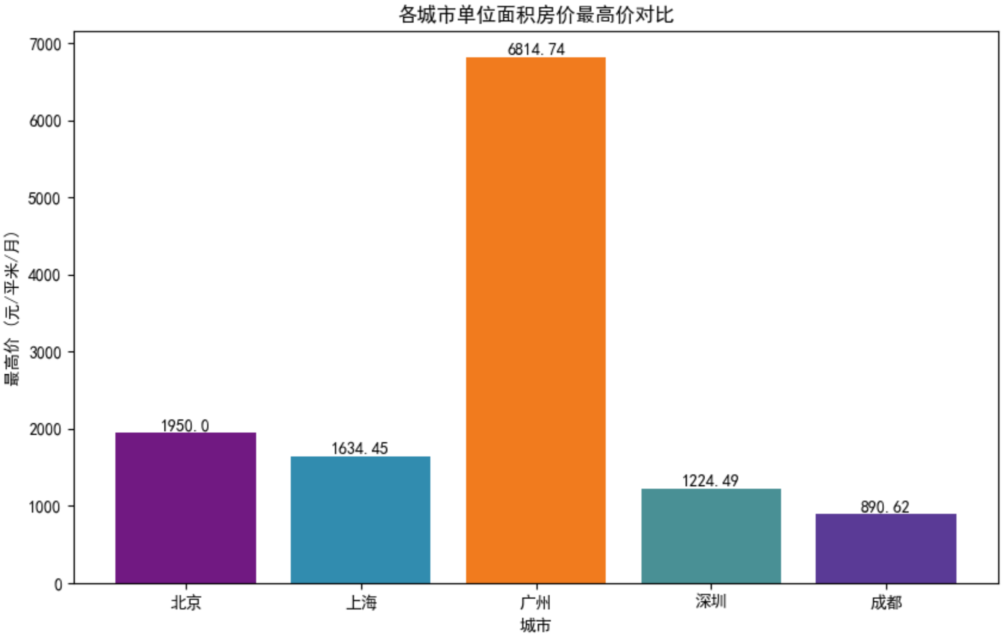

各城市房价中位数对比
中位数（元/月）
0
1000
2000
3000
4000
5000
6000
5778
5400
2800
4500
2300
北京
上海
广州
深圳
成都
城市

## 单位面积房租可视化图:


各城市单位面积房价均价对比
均价（元/平米/月）
0
20
40
60
80
100
120
140
131.52
130.83
97.86
150.41
53.9
北京
上海
广州
深圳
成都
城市


各城市单位面积房价最高价对比
最高价（元/平米/月）
0
1000
2000
3000
4000
5000
6000
7000
1950.0
1634.45
6814.74
1224.49
890.62
北京
上海
广州
深圳
成都
城市





## 4. 比较居室情况

1. 清洗数据, 获取有数字, 且居室数在1,2,3的数据

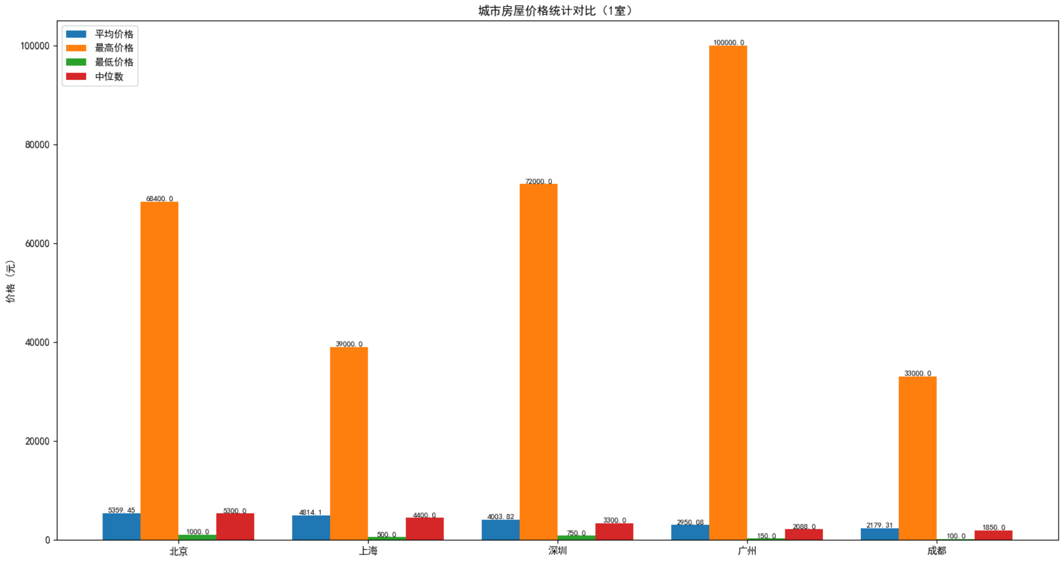
```
def clean_data(self):
    # Cleaning the data
    self.data.dropna(subset=["房型"], inplace=True)
    self.data["室的数量"] = self.data["房型"].str.extract(r"(\d+)室")
    # Remove rows where room number extraction failed
    self.data.dropna(subset=["室的数量"], inplace=True)
    self.data["室的数量"] = self.data["室的数量"].astype(int)
    # Filtering for 1, 2, and 3 rooms only
    self.data = self.data[self.data["室的数量"].isin([1, 2, 3])]
```

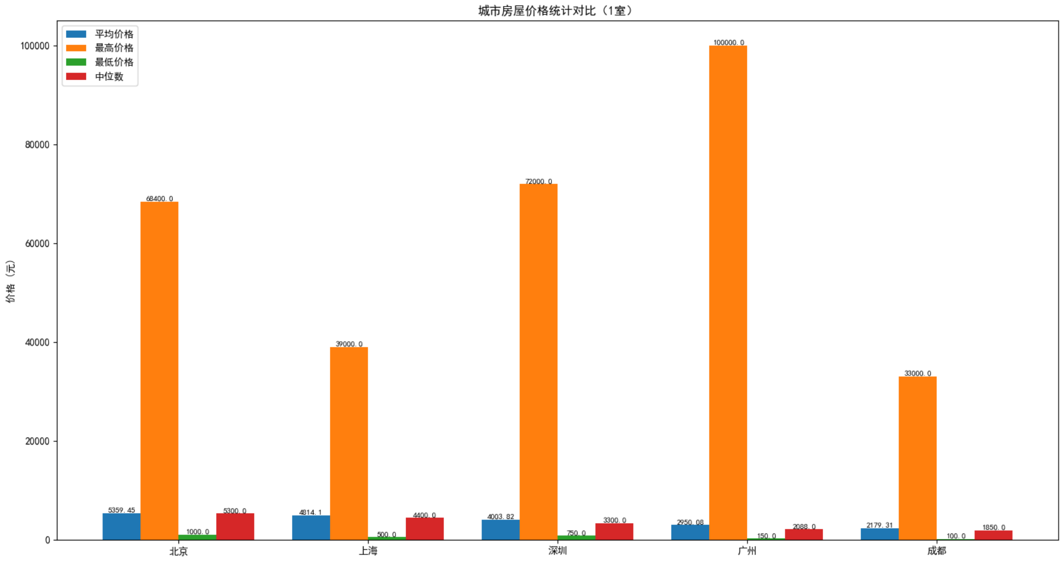
2. 计算均价/最高价/最低价/中位数, 并存储

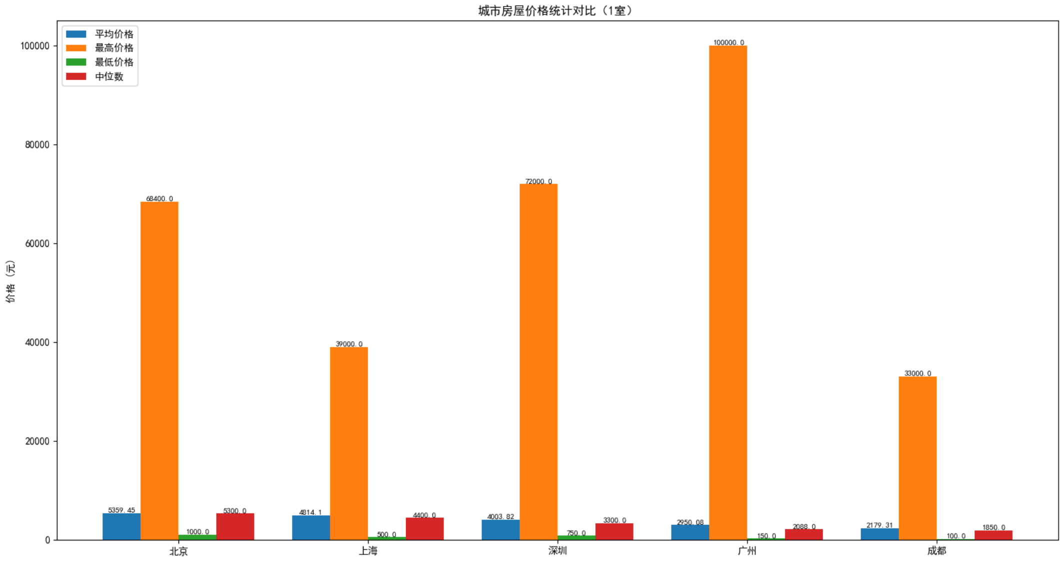
```
def group_data(self):
    # Grouping the data and calculating statistics
    grouped_data_cleaned = self.data.groupby("室的数量")["价格(元/月)"].agg(
        ["mean", "max", "min", "median"]
    )
    grouped_data_cleaned.columns = ["平均价格", "最高价格", "最低价格", "中位数"]
    grouped_data_cleaned.reset_index(inplace=True)
    self.grouped_data_cleaned = grouped_data_cleaned
```

3. 按室的多少分类, 在每张图中分别展示五个城市的均价/最高价/最低价/中位数

```
def plot_grouped_comparison_for_room(rooms, room_number, title):
    # Setting the positions and width for the bars
    pos = list(range(len(rooms)))
    width = 0.2

    plt.rcParams["font.sans-serif"] = ["SimHei"]
    plt.rcParams["axes.unicode_minus"] = False

    fig, ax = plt.subplots(figsize=(15, 8))
    labels = ["北京", "上海", "深圳", "广州", "成都"]

    # Creating bars for each metric
    for i, metric in enumerate(["平均价格", "最高价格", "最低价格", "中位数"]):
        values = [
            room.grouped_data_cleaned[room.grouped_data_cleaned["室的数量"] == room_number][
                metric
            ].mean()
            for room in rooms
        ]
        bars = ax.bar([p + width * i for p in pos], values, width, label=metric)

        # Adding the data labels on top of the bars
        for bar in bars:
            yval = bar.get_height()
            ax.text(
                bar.get_x() + bar.get_width() / 2,
                yval,
                round(yval, 2),
                va="bottom",
                ha="center",
                fontsize=8,
            )

    # Setting the x-axis labels, title, legend, and adjusting layout
    ax.set_xticks([p + 1.5 * width for p in pos])
    ax.set_xticklabels(labels)
    ax.set_title(f"{title}（{room_number}室）")
    ax.legend(loc="upper left")
    plt.ylabel("价格（元）")
    plt.tight_layout()
    plt.show()
```







## 5. 比较板块情况

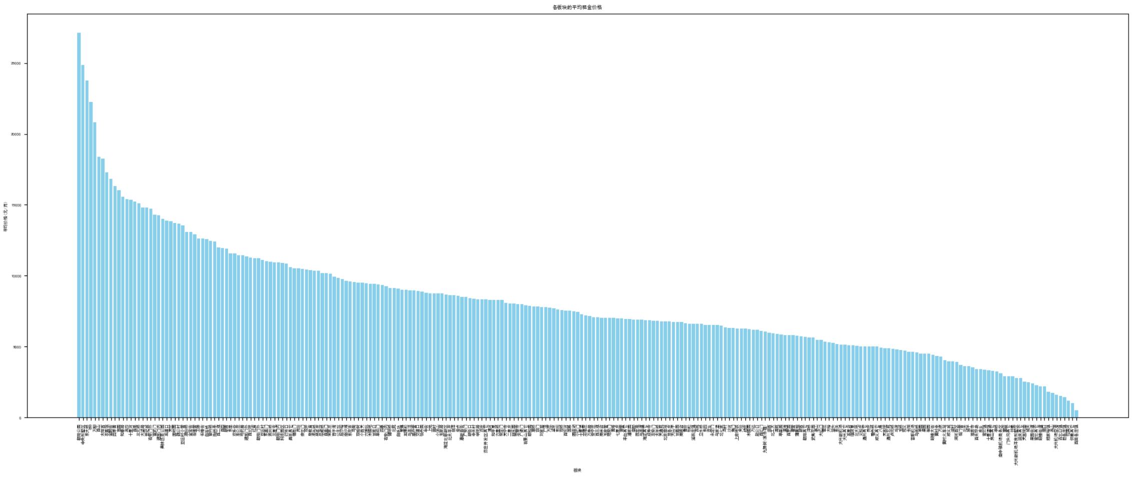
1. 因为在爬虫pipeline中进行了预处理, 可以直接使用数据
2. 计算均价并降序排序:

```
# 读取 CSV 文件
data = pd.read_csv(file_path)

# 根据板块分组并计算每个板块的平均价格
average_prices_by_area = data.groupby("板块")["价格(元/月)"].mean().reset_index()

# 按平均价格降序排序
sorted_average_prices = average_prices_by_area.sort_values(
    by="价格(元/月)", ascending=False
)
```

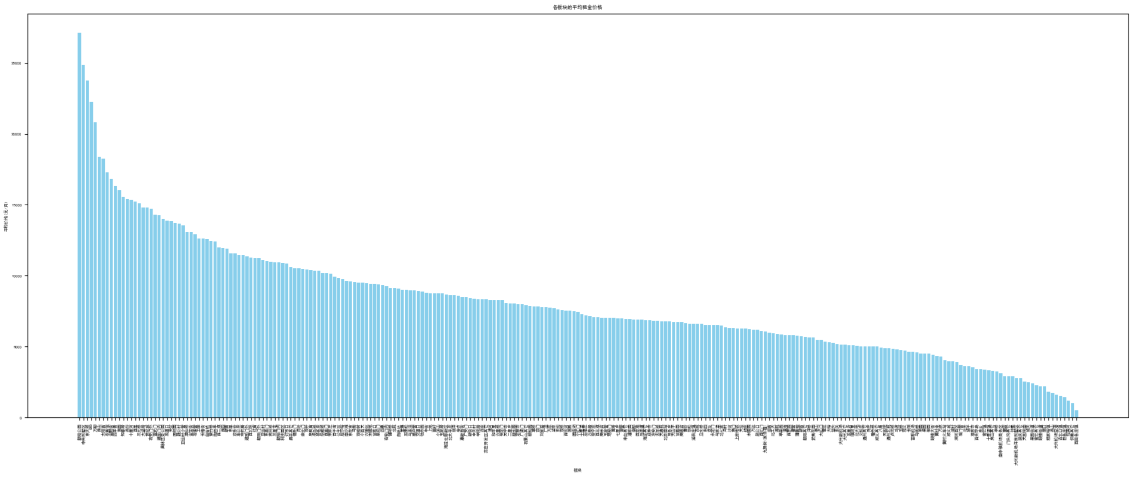
3. 生成五张图, 对应五个城市, 每张图展示所有板块及其均价

```
    plt.rcParams.update({"font.size": 5})
    plt.rcParams["font.sans-serif"] = ["SimHei"]
    plt.rcParams["axes.unicode_minus"] = False
    plt.figure(figsize=(18, 8))
    bars = plt.bar(
        sorted_average_prices["板块"], sorted_average_prices["价格(元/月)"], color="skyblue"
    )

    plt.xlabel("板块")
    plt.ylabel("平均价格(元/月)")
    plt.title("各板块的平均租金价格")
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()
```

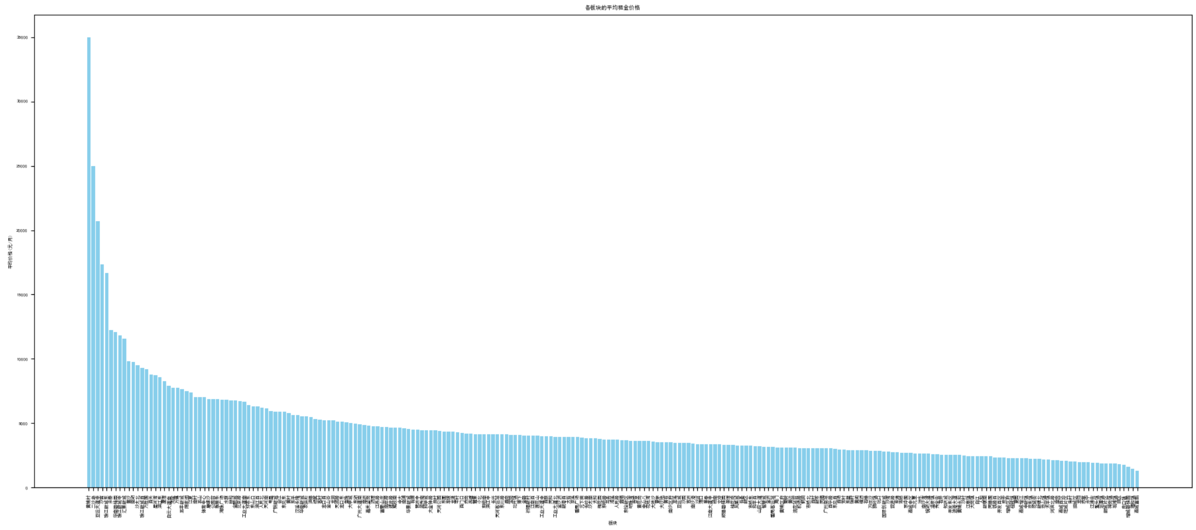
北京



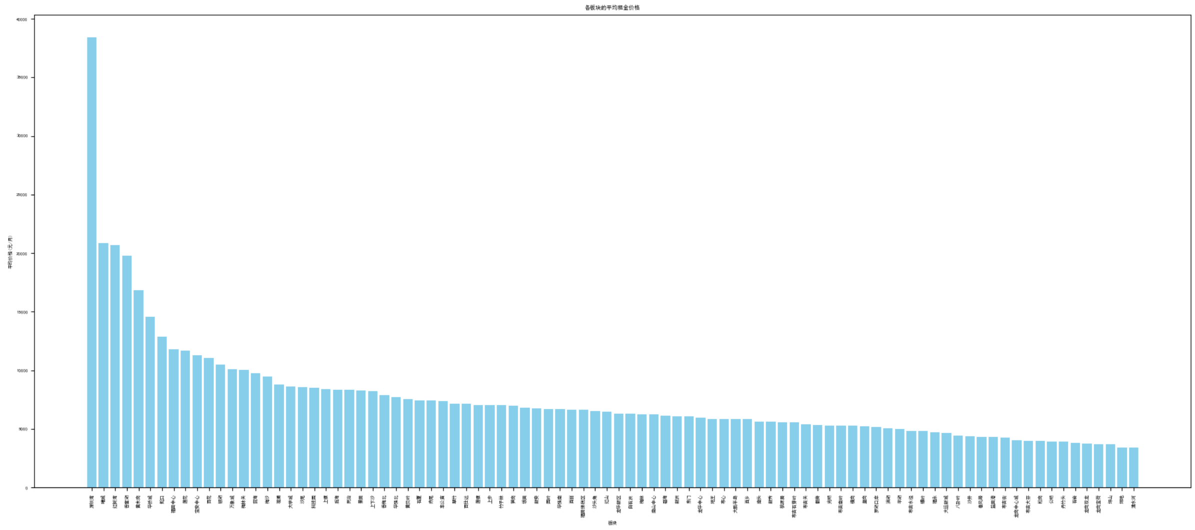
## 上海


各板块的平均租金价格
平均价格(元/月)
板块
0
5000
10000
15000
20000
25000

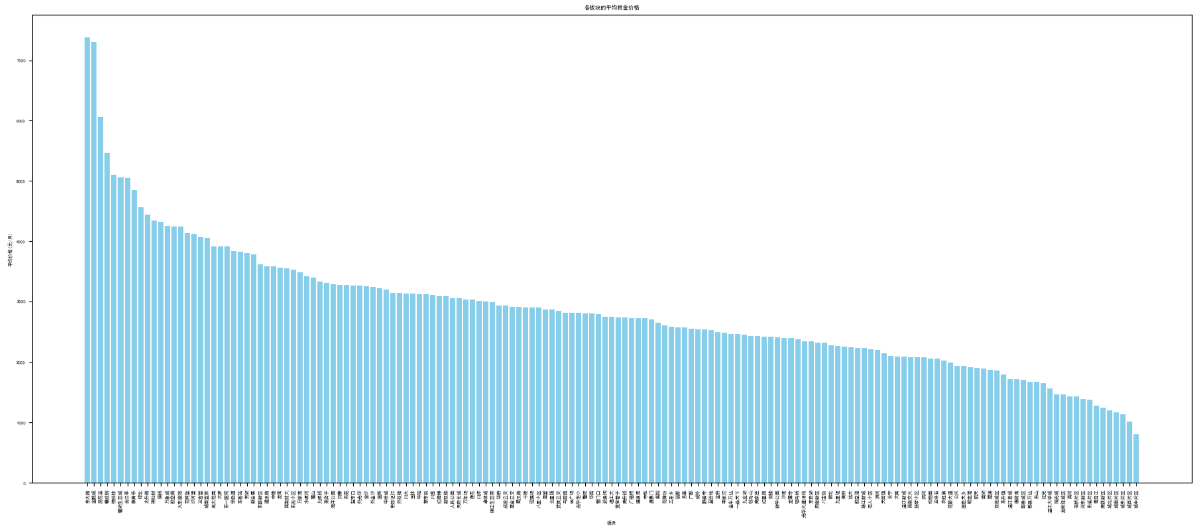
## 广州


各板块的平均租金价格
平均价格(元/月)
板块
0
5000
10000
15000
20000
25000
30000
35000

## 深圳


各板块的平均租金价格
平均价格(元/月)
板块
0
5000
10000
15000
20000
25000
30000
35000
40000

成都



## 6. 朝向比较

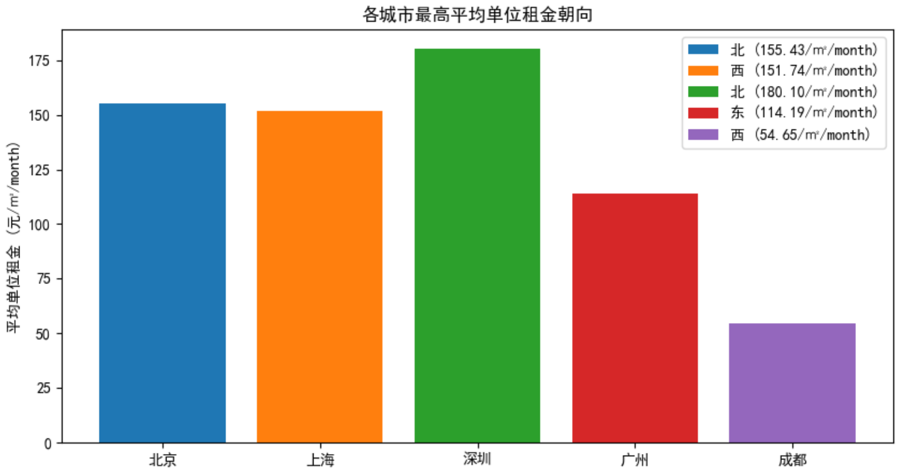
1. 设置合法朝向为"东南西北", 进行数据清洗并结合单位面积价格进行分组

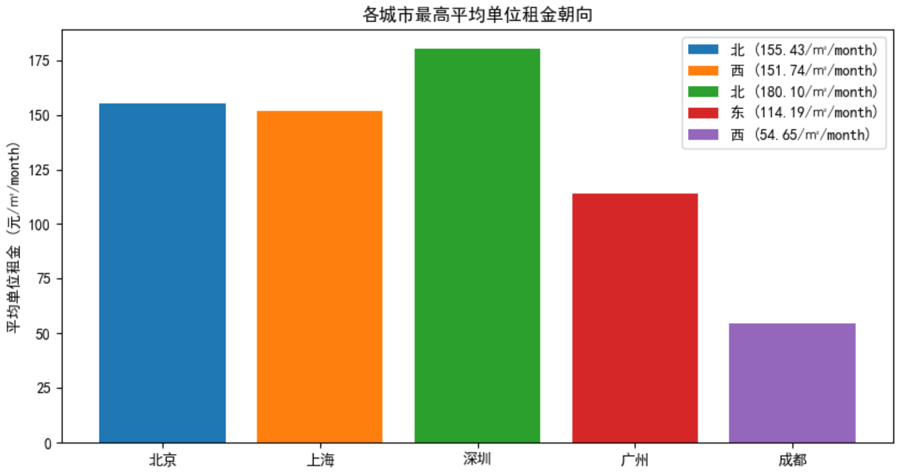
```
def analyze_rent_data(file_path):
    # Load the data
    data = pd.read_csv(file_path)

    # Remove rows with missing data
    data_clean = data.dropna()

    # Keep only rows with a single direction in the orientation (朝向)
column
    valid_directions = ["东", "南", "西", "北"]
    data_clean = data_clean[data_clean["朝向"].isin(valid_directions)]

    # Calculate and return the average unit area rent for each direction
    return data_clean.groupby("朝向")["单位面积价格(元/平米/月)"].mean()
```

2. 计算各个城市最高价方向和最低价方向, 并可视化

```
def plot_rent_directions(cities_data):
    highest_rent_directions = {}
    lowest_rent_directions = {}

    for city, data in cities_data.items():
        highest_rent_direction = data.idxmax()
        lowest_rent_direction = data.idxmin()
        highest_rent_directions[city] = (
            highest_rent_direction,
            data[highest_rent_direction],
        )
        lowest_rent_directions[city] = (
            lowest_rent_direction,
            data[lowest_rent_direction],
        )
    plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False

# Plotting the highest average rent directions
plt.figure(figsize=(10, 5))
plt.title("各城市最高平均单位租金朝向")
for city, (direction, rent) in highest_rent_directions.items():
    plt.bar(city, rent, label=f"{direction} ({rent:.2f}/㎡/month)")
plt.ylabel("平均单位租金 (元/㎡/month)")
plt.legend()
plt.show()

# Plotting the lowest average rent directions
plt.figure(figsize=(10, 5))
plt.title("各城市最低平均租金朝向")
for city, (direction, rent) in lowest_rent_directions.items():
    plt.bar(city, rent, label=f"{direction} ({rent:.2f}/㎡/month)")
plt.ylabel("平均单位租金 (元/㎡/month)")
plt.legend()
plt.show()
```

3. 图表展示





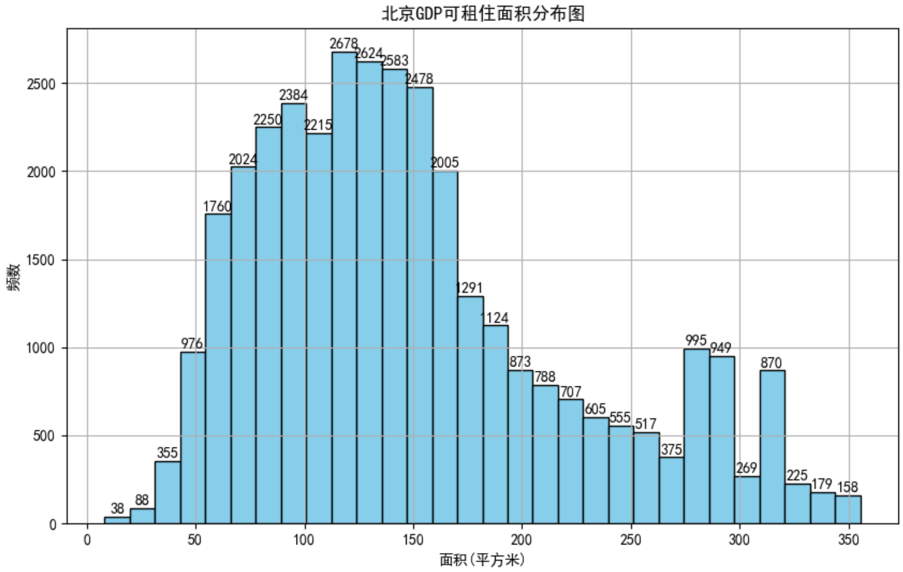
4. 分析: 各个城市并不一致, 我认为该数据与当地的纬度及太阳朝向有关

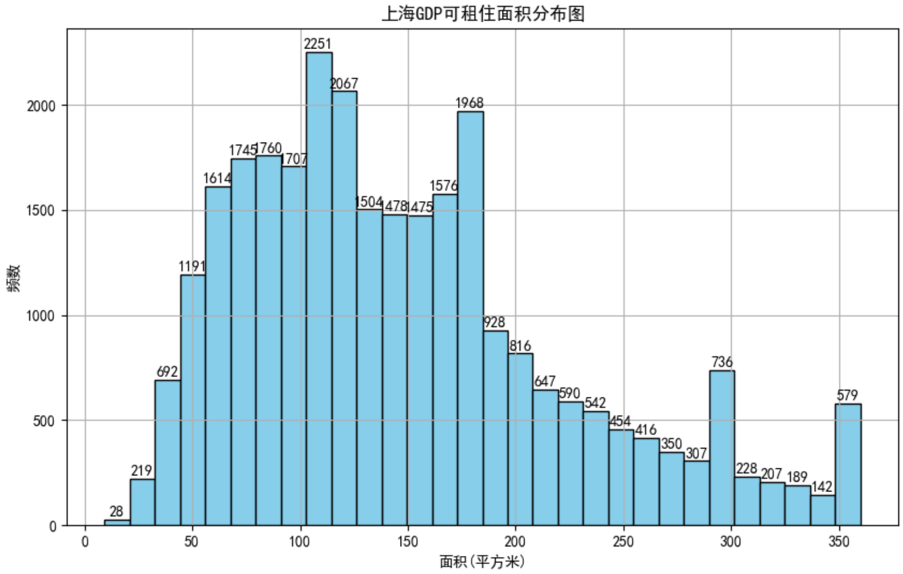
## 7. 人均GDP与单位面积租金的关系

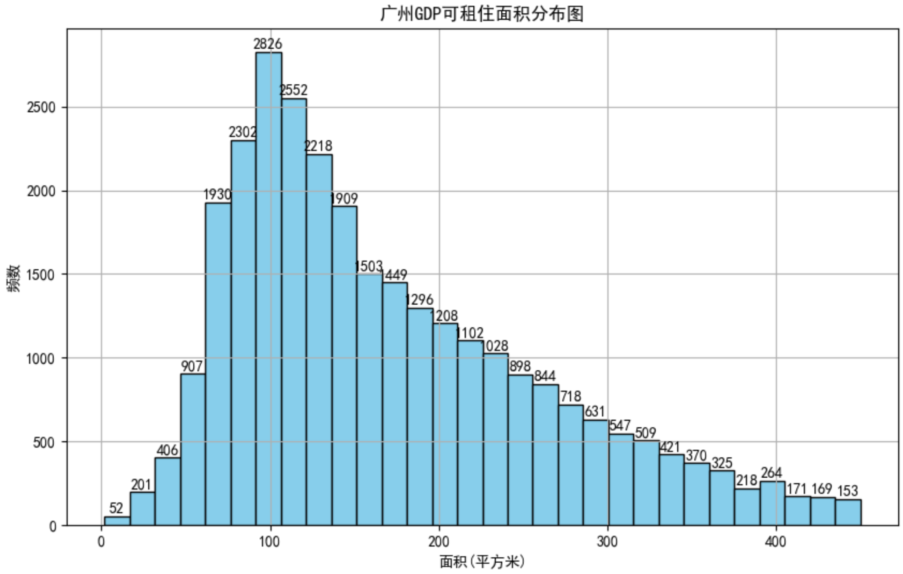
1. 上网查询各地的人均GDP
2. 计算人均GDP与各地房租的比值, 记为GDP可租住面积, 记录频数生成GDP可租住面积分布图, 通过查看高频面积区间来判断该城市的生活压力

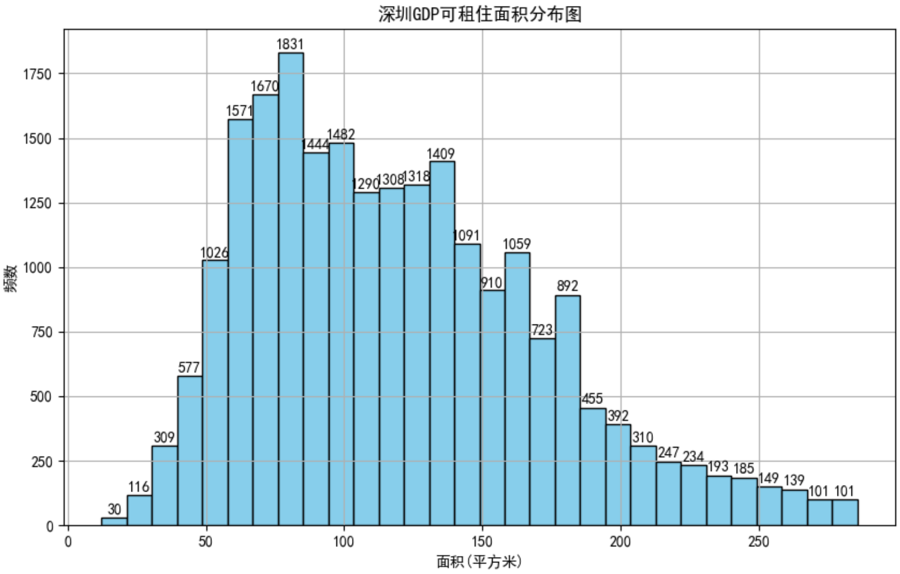
```
 GDP = {
        "北京": 190000,
        "上海": 180000,
        "广州": 150000,
        "深圳": 180000,
        "成都": 100000,
    }
    gdp_per_capita = GDP[city_name]  # in Yuan

    # Calculating the number of square meters of housing that can be afforded for a year with the GDP
    data["affordable_area_sqm"] = gdp_per_capita / (data["单位面积价格(元/平米/月)"] * 12)

    # Removing extreme points (outliers) for a clearer view
    # Using the Interquartile Range (IQR) method to identify outliers
    Q1 = data["affordable_area_sqm"].quantile(0.25)
    Q3 = data["affordable_area_sqm"].quantile(0.75)
    IQR = Q3 - Q1

    # Defining outliers as points outside of Q1 - 1.5*IQR and Q3 + 1.5*IQR
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Filtering the data to exclude outliers
    filtered_data = data[
        (data["affordable_area_sqm"] >= lower_bound)
        & (data["affordable_area_sqm"] <= upper_bound)
    ]

    # Creating a histogram to show the distribution of affordable housing area
    plt.rcParams["font.sans-serif"] = ["SimHei"]
    plt.rcParams["axes.unicode_minus"] = False

    plt.figure(figsize=(10, 6))
    plt.hist(
        filtered_data["affordable_area_sqm"],
        bins=30,
        color="skyblue",
        edgecolor="black",
    )

    # Adding value labels to each bar
    for rect in plt.gca().patches:
        height = rect.get_height()
        plt.gca().text(
            rect.get_x() + rect.get_width() / 2,
            height + 5,
            f"{int(height)}",
            ha="center",
            va="bottom",
        )

    plt.title(f"{city_name}GDP可租住面积分布图")
    plt.xlabel("面积(平方米)")
    plt.ylabel("频数")
    plt.grid(True)
    plt.show()
```

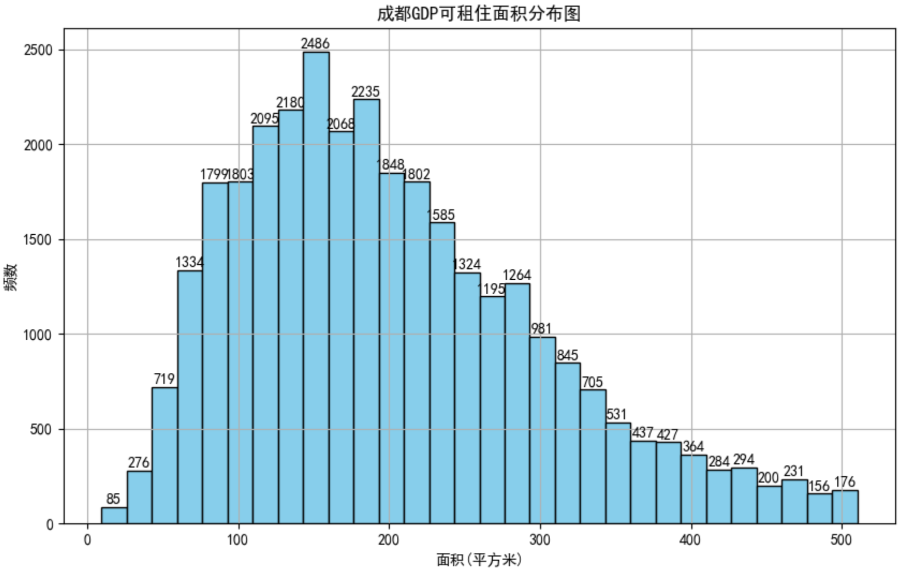
3. 图表展示:


北京GDP可租住面积分布图
频数
面积(平方米)
0
500
1000
1500
2000
2500
0
50
100
150
200
250
300
350
38
88
355
976
1760
2024
2250
2384
2215
2678
2624
2583
2478
2005
1291
1124
873
788
707
605
555
517
375
995
949
269
870
225
179
158


上海GDP可租住面积分布图
频数
面积(平方米)
0
500
1000
1500
2000
0
50
100
150
200
250
300
350
28
219
692
1191
1614
1745
1760
1707
2251
2067
1504
1478
1475
1576
1968
928
816
647
590
542
454
416
350
307
736
228
207
189
142
579


广州GDP可租住面积分布图
频数
面积(平方米)
0
500
1000
1500
2000
2500
0
100
200
300
400
52
201
406
907
1930
2302
2826
2552
2218
1909
1503
1449
1296
1208
1102
1028
898
844
718
631
547
509
421
370
325
218
264
171
169
153





4. 分析: 查看图表可知, 成都的可居住面积较高, 因此在成都租房性价比较高

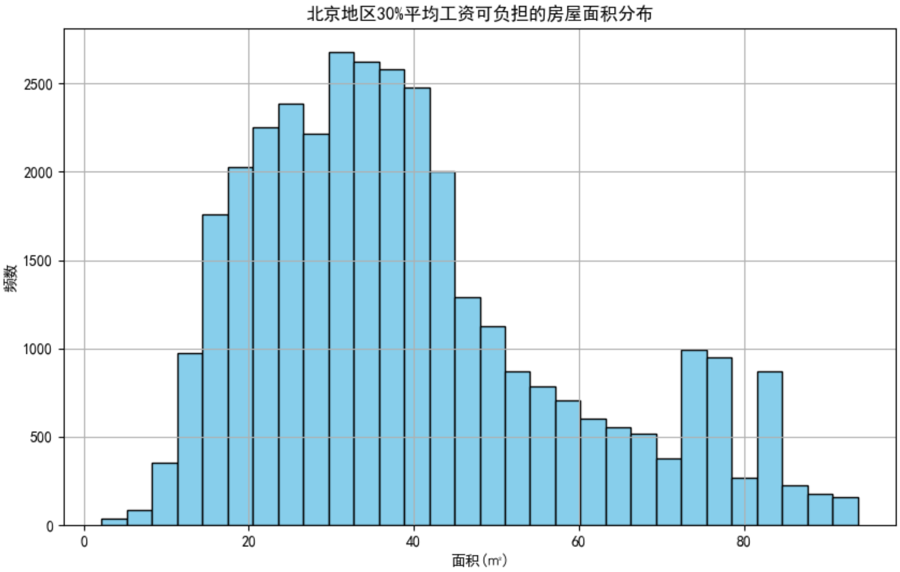
## 8. 平均工资与单位面积租金

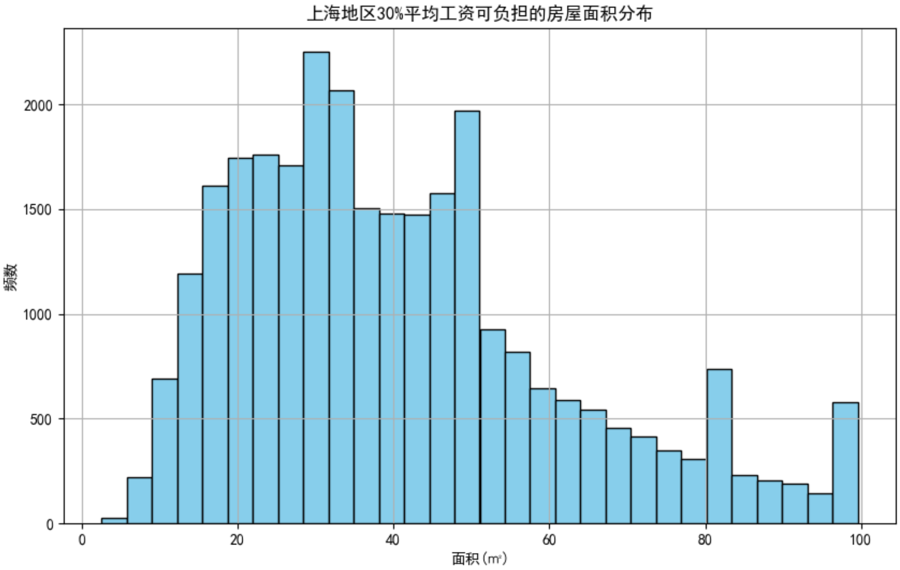
1. 获取各地的平均工资
2. 按国际惯例, 使用30%工资进行租房较为合理, 因此计算30%工资在当地可以租到的平方数, 通过面积数据体现当地租房压力

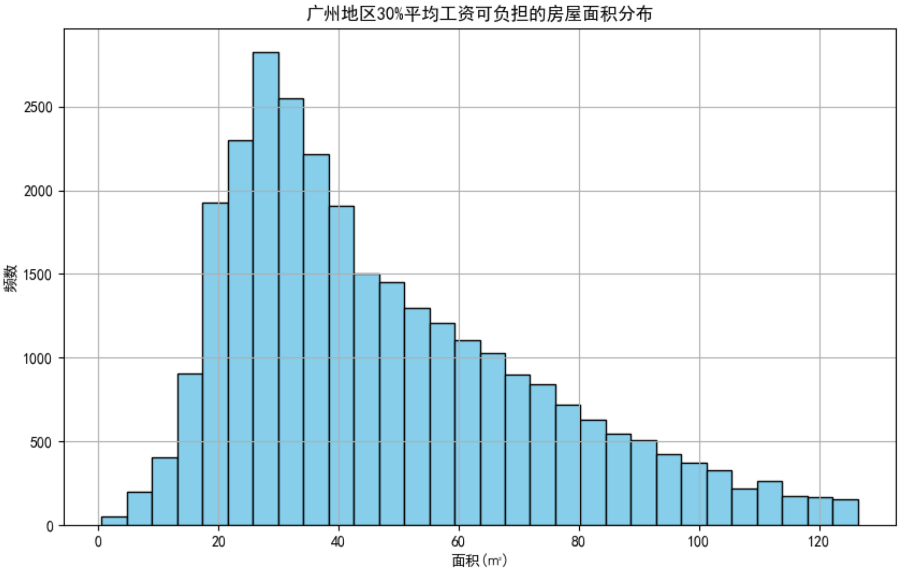
```
def analyze_housing_affordability_salary(file_path, city_name):
    # Load the CSV file
    data = pd.read_csv(file_path)
    salarys = {
        "北京": 13930,
        "上海": 13832,
        "广州": 11710,
        "深圳": 13086,
        "成都": 10039,
    }
    # Calculating the number of square meters of housing that can be
afforded for a year with the average salary
```

```
    data["affordable_area_sqm"] = 0.3 * salarys[city_name] / (data["单位面积价格(元/平米/月)"])

    # Removing extreme points (outliers) for a clearer view
    # Using the Interquartile Range (IQR) method to identify outliers
    Q1 = data["affordable_area_sqm"].quantile(0.25)
    Q3 = data["affordable_area_sqm"].quantile(0.75)
    IQR = Q3 - Q1

    # Defining outliers as points outside of Q1 - 1.5*IQR and Q3 + 1.5*IQR
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Filtering the data to exclude outliers
    filtered_data = data[
        (data["affordable_area_sqm"] >= lower_bound)
        & (data["affordable_area_sqm"] <= upper_bound)
    ]

    # Creating a histogram to show the distribution of affordable housing area based on average salary
    plt.rcParams["font.sans-serif"] = ["SimHei"]
    plt.rcParams["axes.unicode_minus"] = False
    plt.figure(figsize=(10, 6))
    plt.hist(
        filtered_data["affordable_area_sqm"],
        bins=30,
        color="skyblue",
        edgecolor="black",
    )
    plt.title(f"{city_name}地区30%平均工资可负担的房屋面积分布")
    plt.xlabel("面积(㎡)")
    plt.ylabel("频数")
    plt.grid(True)
    plt.show()
```

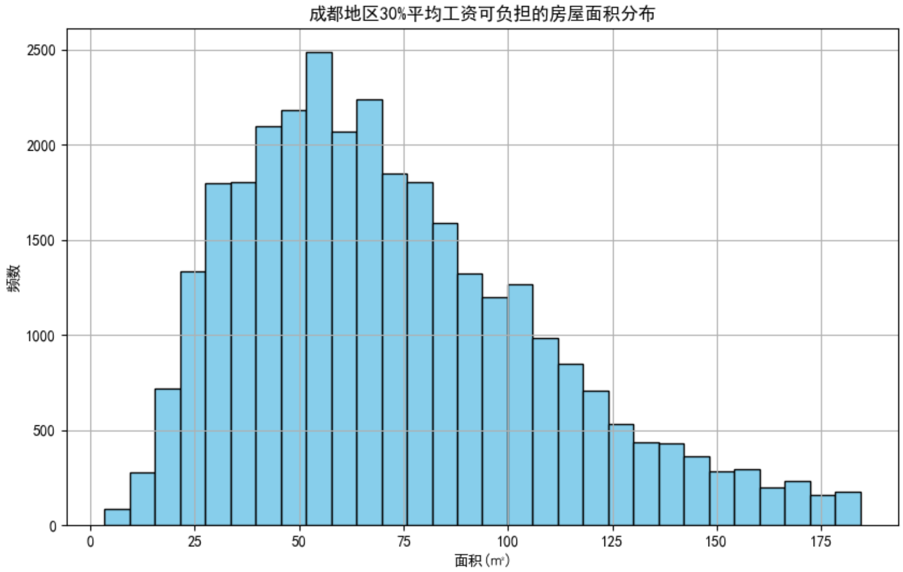
3. 图表展示

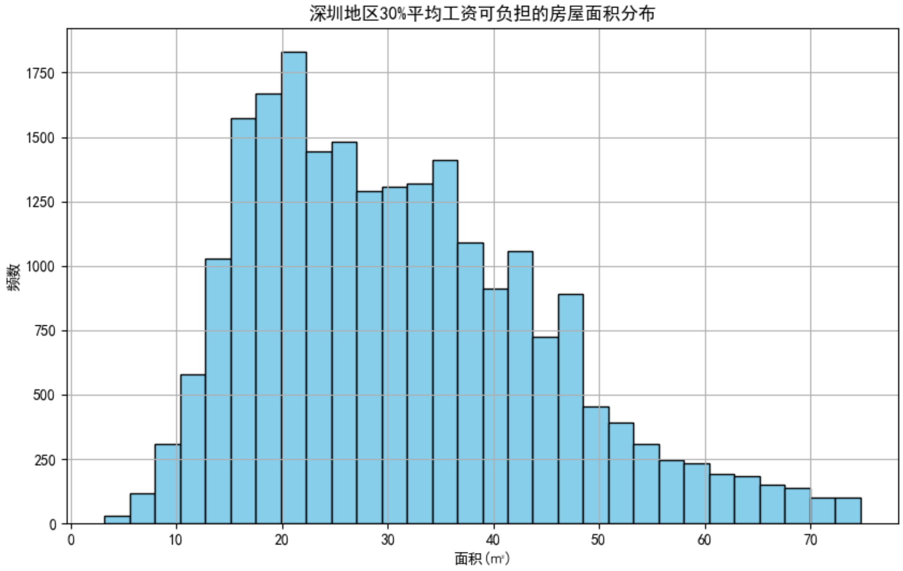






![广州](./assets/广州-1702548389577-44.png





4. 读图分析可知, 在成都, 使用平均工资30%可租住的房子面积最大, 租房负担最小