北京郵電大學

Beijing University of Posts and Telecommunications

Python 程序设计

[数据预处理实验]

学院：计算机学院
2020211376 马天成
2022 年 12 月 11 日

北京邮电大学《Python程序设计》课程实验报告

实验名称	数据预处理	学院	计算机	指导教师	王晶
班 级	班内序号	学 号	学生姓名	成绩	
2020211305	2	2020211376	马天成		
	作业 1：爬取并存储链家的新房数据，并进行预处理。 作业 2：分析处理 2015 年北京市 PM2.5 指数数据集空值				
学生实验报告	(详见“实验报告和源程序”册)				
课程设计成绩评定	评语: 成绩: 指导教师签名: 年 月 日				

注：评语要体现每个学生的工作情况，可以加页。

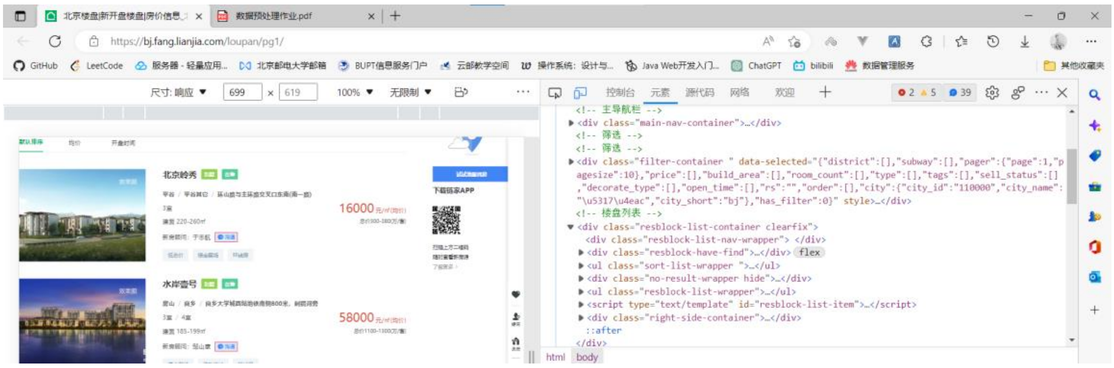
目录

1. 实验目的和要求

实验内容	作业 1：爬取并存储链家的新房数据，并进行预处理。 （1）爬取起始网页：https://bj.fang.lianjia.com/loupan/ （2）爬取信息的提取及存储要求（单条数据示例在第 3 页） （4）异常值处理 • 列出总价在均值三倍标准差以外的房屋，展示其基本信息（如果太 多可以只展示一部分） （3）数据统计 • 找出总价最贵和最便宜的房子，以及总价的中位数 • 找出均价最贵和最便宜的房子，以及均价的中位数 （5）离散化处理 • 对房屋的均价进行离散化处理，自行设定每个区间的长度并给出设 置的理由，给出每个区间的房屋数量和所占比例 作业 2：分析处理 2015 年北京市 PM2.5 指数数据集空值 （1）原始数据集：BeijingPM20100101_20151231.csv（列信息见第 5 页 说明） （2）数据抽取及存储：从原始数据集中抽取 2015 年度数据，存储为新 的 csv 文件 （3）找出空值：对新的 csv 文件，找出存在的空值列及相应的空值数量 （4）空值处理方法：对所有存在空值的列，给出空值的处理方法及理 由，要求处理方法必须可在本数据集范围内执行 （5）空值处理并存储：按照自己的处理方法，通过 pandas、numpy 或 python 方法对空值进行处理，完成后给出新的空值列信息，并将处理后 的数据（不涉及空值的列应原样保留）存储为新的 csv 文件

2. 爬取房屋数据

2.1 数据观测



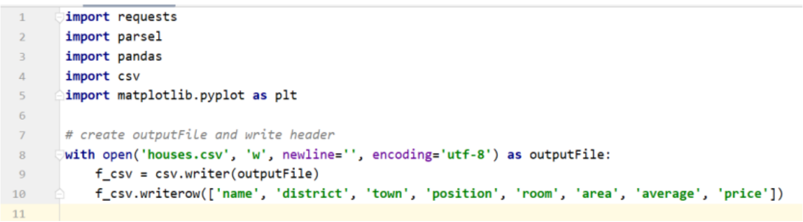
```
<!-- 主导航栏 -->
▶<div class="main-nav-container">…</div>
<!-- 筛选 -->
<!-- 筛选 -->
▶<div class="filter-container " data-selected="{"district":[],"subway":[],"pager":{"page":1,"pagesize":10},"price":[],"build_area":[],"room_count":[],"type":[],"tags":[],"sell_status":[],"decorate_type":[],"open_time":[],"rs":"","order":[],"city":{"city_id":"110000","city_name":"\u5317\u4eac","city_short":"bj"},"has_filter":0}" style>…</div>
<!-- 楼盘列表 -->
▼<div class="resblock-list-container clearfix">
   <div class="resblock-list-nav-wrapper"> </div>
  ▶<div class="resblock-have-find">…</div> flex
  ▶<ul class="sort-list-wrapper ">…</ul>
  ▶<div class="no-result-wrapper hide">…</div>
  ▶<ul class="resblock-list-wrapper">…</ul>
  ▶<script type="text/template" id="resblock-list-item">…</script>
  ▶<div class="right-side-container">…</div>
   ::after
</div>
```

很显然可以观察到展示数据的模块。

我们需要做的就是根据库把所有 div 里的数值拿出来。

2.2 代码编写

2.2.1 准备输出文件

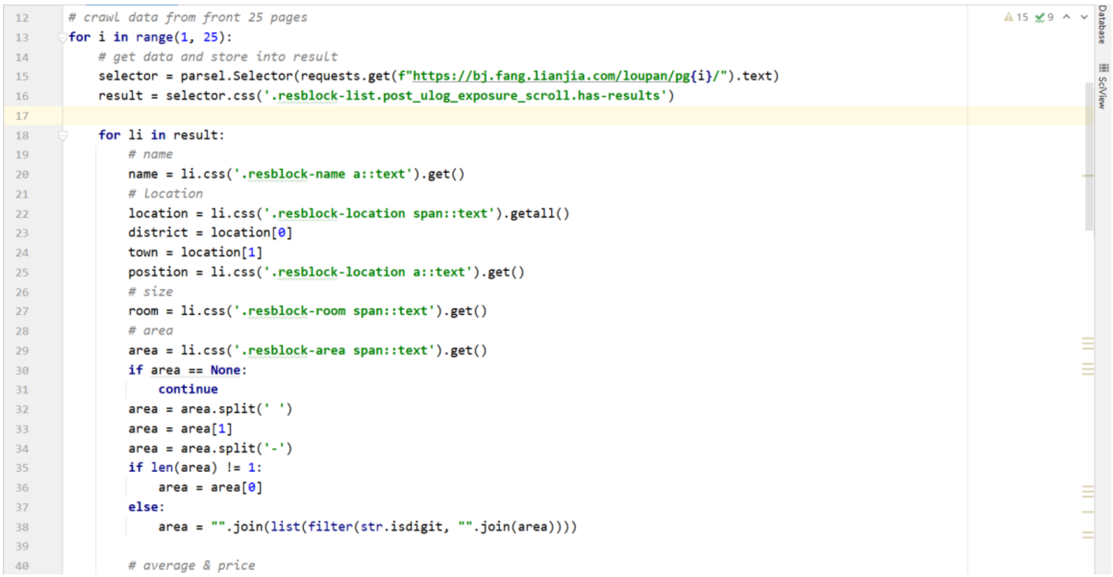


```python
import requests
import parsel
import pandas
import csv
import matplotlib.pyplot as plt

# create outputFile and write header
with open('houses.csv', 'w', newline='', encoding='utf-8') as outputFile:
    f_csv = csv.writer(outputFile)
    f_csv.writerow(['name', 'district', 'town', 'position', 'room', 'area', 'average', 'price'])

```

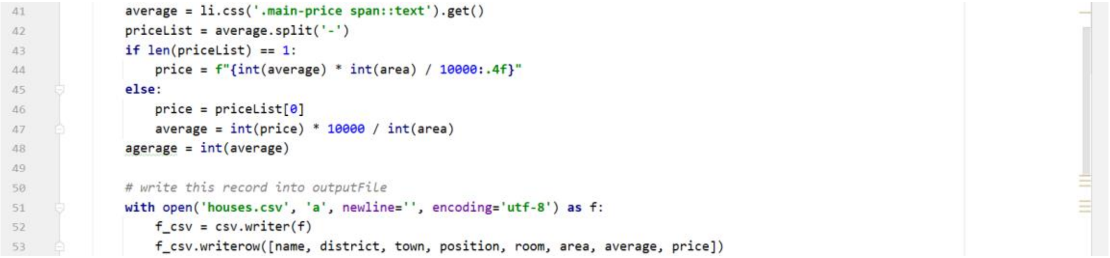
是准备 output 的文件。这里使用 csv 来存储。

字段根据英文名就可以知道了吧。

2.2.2 读取前 25 页数据到输出文件

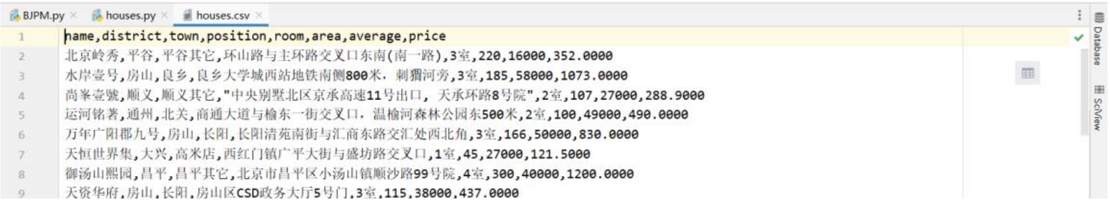


```python
# crawl data from front 25 pages
for i in range(1, 25):
    # get data and store into result
    selector = parsel.Selector(requests.get(f"https://bj.fang.lianjia.com/loupan/pg{i}/").text)
    result = selector.css('.resblock-list.post_ulog_exposure_scroll.has-results')

    for li in result:
        # name
        name = li.css('.resblock-name a::text').get()
        # location
        location = li.css('.resblock-location span::text').getall()
        district = location[0]
        town = location[1]
        position = li.css('.resblock-location a::text').get()
        # size
        room = li.css('.resblock-room span::text').get()
        # area
        area = li.css('.resblock-area span::text').get()
        if area == None:
            continue
        area = area.split(' ')
        area = area[1]
        area = area.split('-')
        if len(area) != 1:
            area = area[0]
        else:
            area = "".join(list(filter(str.isdigit, "".join(area))))

        # average & price
```



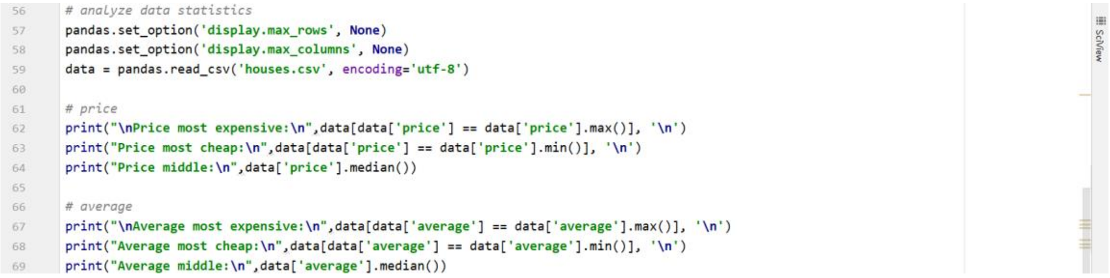
```python
        average = li.css('.main-price span::text').get()
        priceList = average.split('-')
        if len(priceList) == 1:
            price = f"{int(average) * int(area) / 10000:.4f}"
        else:
            price = priceList[0]
            average = int(price) * 10000 / int(area)
        agerage = int(average)

        # write this record into outputFile
        with open('houses.csv', 'a', newline='', encoding='utf-8') as f:
            f_csv = csv.writer(f)
            f_csv.writerow([name, district, town, position, room, area, average, price])
```

这里也非常简单，只不过点繁琐。根据 div 的属性把所有的值取出来，适当处理后形成需要的记录存储到 csv。这就是数据预处理。输出文件长这样：



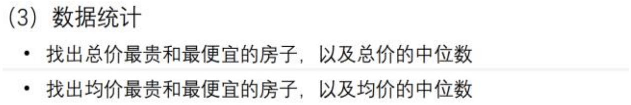
```
name,district,town,position,room,area,average,price
北京岭秀,平谷,平谷其它,环山路与主环路交叉口东南(南一路),3室,220,16000,352.0000
水岸壹号,房山,良乡,良乡大学城西站地铁南侧800米，刺猬河旁,3室,185,58000,1073.0000
尚峯壹號,顺义,顺义其它,"中央别墅北区京承高速11号出口， 天承环路8号院",2室,107,27000,288.9000
运河铭著,通州,北关,商通大道与榆东一街交叉口，温榆河森林公园东500米,2室,100,49000,490.0000
万年广阳郡九号,房山,长阳,长阳清苑南街与汇商东路交汇处西北角,3室,166,50000,830.0000
天恒世界集,大兴,高米店,西红门镇广平大街与盛坊路交叉口,1室,45,27000,121.5000
御汤山熙园,昌平,昌平其它,北京市昌平区小汤山镇顺沙路99号院,4室,300,40000,1200.0000
天资华府,房山,长阳,房山区CSD政务大厅5号门,3室,115,38000,437.0000
```

2.2.3 分析数据



```
# analyze data statistics
pandas.set_option('display.max_rows', None)
pandas.set_option('display.max_columns', None)
data = pandas.read_csv('houses.csv', encoding='utf-8')

# price
print("\nPrice most expensive:\n",data[data['price'] == data['price'].max()], '\n')
print("Price most cheap:\n",data[data['price'] == data['price'].min()], '\n')
print("Price middle:\n",data['price'].median())

# average
print("\nAverage most expensive:\n",data[data['average'] == data['average'].max()], '\n')
print("Average most cheap:\n",data[data['average'] == data['average'].min()], '\n')
print("Average middle:\n",data['average'].median())
```

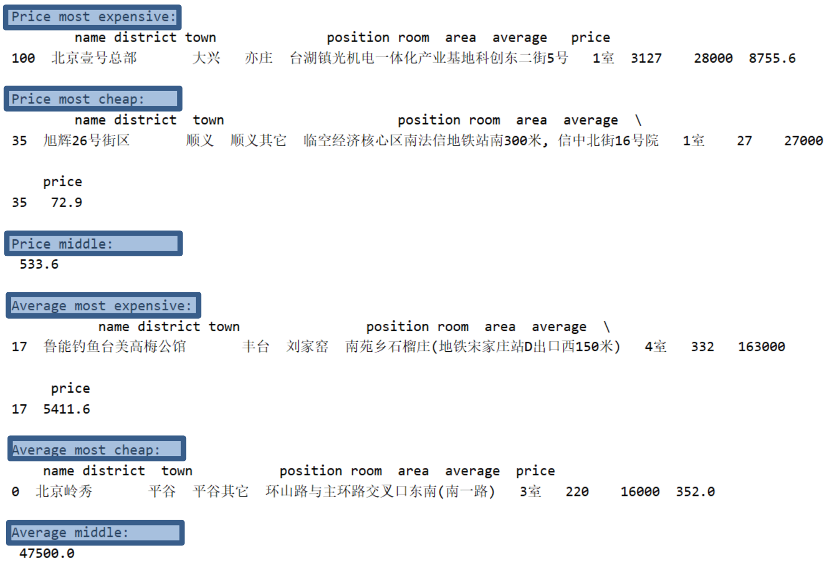
这里是寻找一些特殊的记录展示出来。这里分别是找到：



(3) 数据统计

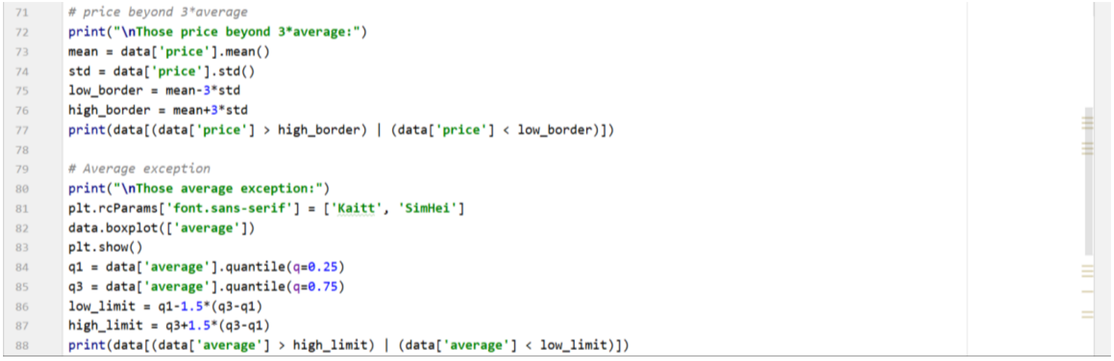
- 找出总价最贵和最便宜的房子，以及总价的中位数
- 找出均价最贵和最便宜的房子，以及均价的中位数

所以 6 个记录分别是：



```
Price most expensive:
        name district town                 position room  area  average   price
100  北京壹号总部        大兴    亦庄  台湖镇光机电一体化产业基地科创东二街5号   1室  3127    28000  8755.6

Price most cheap:
        name district  town                          position room  area  average  \
35  旭辉26号街区        顺义  顺义其它  临空经济核心区南法信地铁站南300米, 信中北街16号院   1室    27    27000

    price
35   72.9

Price middle:
 533.6

Average most expensive:
           name district town                      position room  area  average  \
17  鲁能钓鱼台美高梅公馆        丰台  刘家窑  南苑乡石榴庄(地铁宋家庄站D出口西150米)   4室   332   163000

     price
17  5411.6

Average most cheap:
    name district  town              position room  area  average  price
0  北京岭秀        平谷  平谷其它  环山路与主环路交叉口东南(南一路)   3室   220    16000  352.0

Average middle:
 47500.0
```

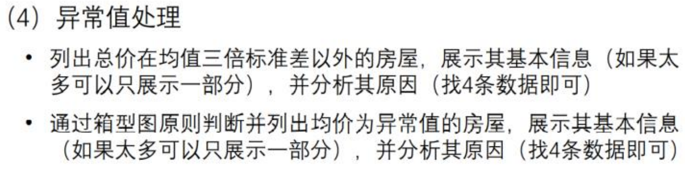
2.2.4 异常处理



```
# price beyond 3*average
print("\nThose price beyond 3*average:")
mean = data['price'].mean()
std = data['price'].std()
low_border = mean-3*std
high_border = mean+3*std
print(data[(data['price'] > high_border) | (data['price'] < low_border)])

# Average exception
print("\nThose average exception:")
plt.rcParams['font.sans-serif'] = ['Kaitt', 'SimHei']
data.boxplot(['average'])
plt.show()
q1 = data['average'].quantile(q=0.25)
q3 = data['average'].quantile(q=0.75)
low_limit = q1-1.5*(q3-q1)
high_limit = q3+1.5*(q3-q1)
print(data[(data['average'] > high_limit) | (data['average'] < low_limit)])
```

这里就是很显然的数据处理了。

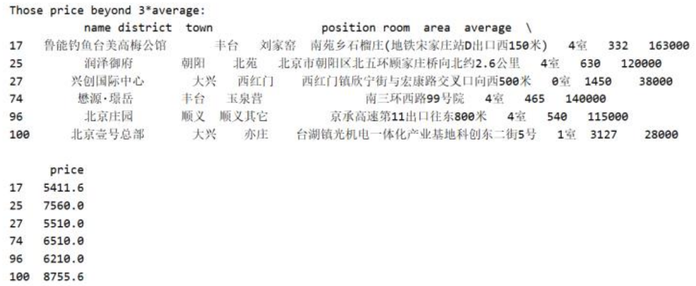
找出的是以下属性的记录：



（4）异常值处理

- 列出总价在均值三倍标准差以外的房屋，展示其基本信息（如果太多可以只展示一部分），并分析其原因（找4条数据即可）
- 通过箱型图原则判断并列出均价为异常值的房屋，展示其基本信息（如果太多可以只展示一部分），并分析其原因（找4条数据即可）

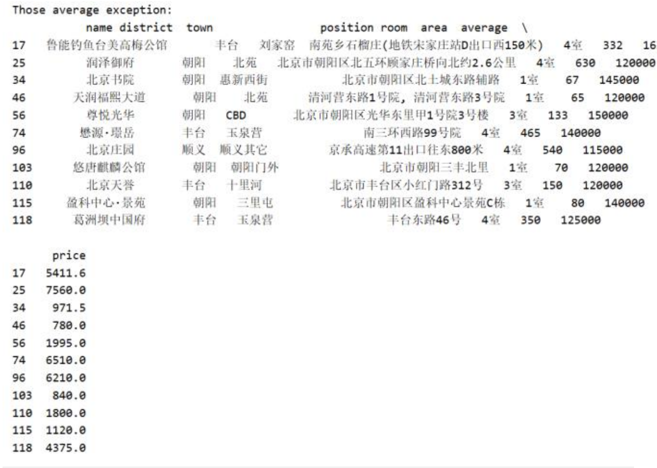
1. 超三倍的异常记录如下：



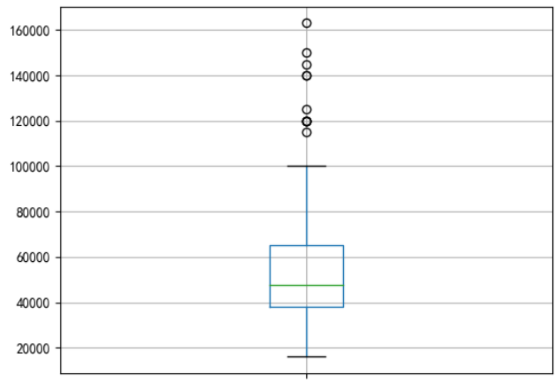
```
Those price beyond 3*average:
            name district  town                    position room  area  average  \
17   鲁能钓鱼台美高梅公馆        丰台    刘家窑  南苑乡石榴庄(地铁宋家庄站D出口西150米)   4室   332   163000
25        润泽御府       朝阳     北苑   北京市朝阳区北五环顺家庄桥向北约2.6公里   4室   630   120000
27      兴创国际中心       大兴    西红门    西红门镇欣宁街与宏康路交叉口向西500米   0室  1450    38000
74       懋源·璟岳       丰台    玉泉营                南三环西路99号院   4室   465   140000
96        北京庄园       顺义   顺义其它          京承高速第11出口往东800米   4室   540   115000
100     北京壹号总部       大兴     亦庄   台湖镇光机电一体化产业基地科创东二街5号   1室  3127    28000

      price
17   5411.6
25   7560.0
27   5510.0
74   6510.0
96   6210.0
100  8755.6
```

在这个纪录中，可看到这些圆点。这些数据的共同点为面积大，均价高，房屋所占面积越大总价越高。

2. 均价异常的房屋如下：

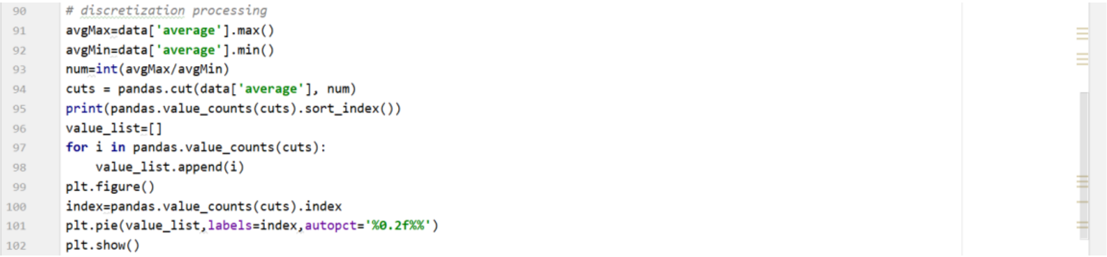


```
Those average exception:
            name district  town                    position room  area  average  \
17   鲁能钓鱼台美高梅公馆        丰台    刘家窑  南苑乡石榴庄(地铁宋家庄站D出口西150米)   4室   332   163000
25        润泽御府       朝阳     北苑   北京市朝阳区北五环顺家庄桥向北约2.6公里   4室   630   120000
34        北京书院       朝阳   惠新西街           北京市朝阳区北土城东路辅路   1室    67   145000
46      天润福熙大道       朝阳     北苑     清河营东路1号院，清河营东路3号院   1室    65   120000
56        尊悦光华       朝阳    CBD      北京市朝阳区光华东里甲1号院3号楼   3室   133   150000
74       懋源·璟岳       丰台    玉泉营                南三环西路99号院   4室   465   140000
96        北京庄园       顺义   顺义其它          京承高速第11出口往东800米   4室   540   115000
103     悠唐麒麟公馆       朝阳   朝阳门外                北京市朝阳三丰北里   1室    70   120000
110       北京天誉       丰台    十里河          北京市丰台区小红门路312号   3室   150   120000
115     盈科中心·景苑       朝阳    三里屯          北京市朝阳区盈科中心景苑C栋   1室    80   140000
118     葛洲坝中国府       丰台    玉泉营                  丰台东路46号   4室   350   125000

      price
17   5411.6
25   7560.0
34    971.5
46    780.0
56   1995.0
74   6510.0
96   6210.0
103   840.0
110  1800.0
115  1120.0
118  4375.0
```



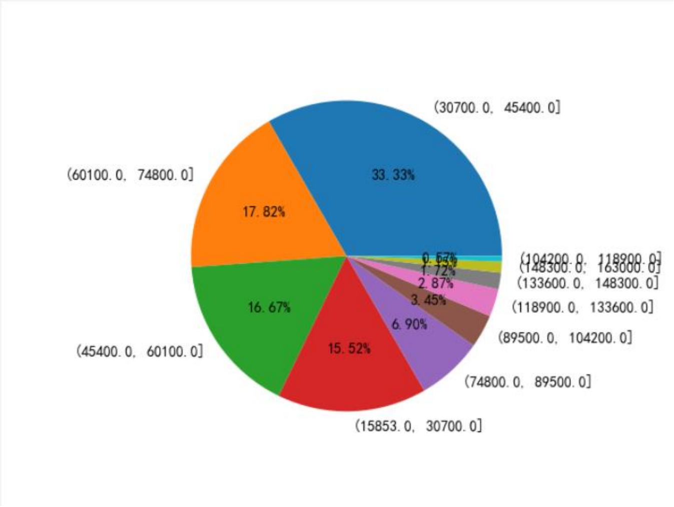
这些都是因为单价很高或 price 很高的房屋，导致最终的 average 计算出来高。

2.2.5 离散化处理



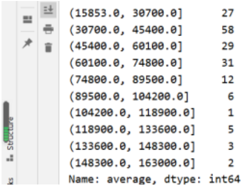
```python
# discretization processing
avgMax=data['average'].max()
avgMin=data['average'].min()
num=int(avgMax/avgMin)
cuts = pandas.cut(data['average'], num)
print(pandas.value_counts(cuts).sort_index())
value_list=[]
for i in pandas.value_counts(cuts):
    value_list.append(i)
plt.figure()
index=pandas.value_counts(cuts).index
plt.pie(value_list,labels=index,autopct='%0.2f%%')
plt.show()
```

直接把东西贴出来完事。就是简单把这些归类分组然后展示成饼状图。



分组依据：用均价最高值除以均价最低值获取划分个数。

分组结果：

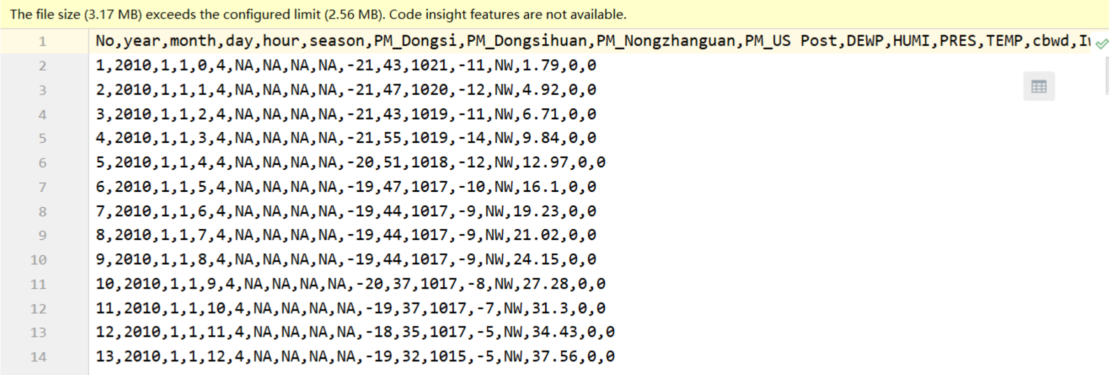


```
(15853.0, 30700.0]      27
(30700.0, 45400.0]      58
(45400.0, 60100.0]      29
(60100.0, 74800.0]      31
(74800.0, 89500.0]      12
(89500.0, 104200.0]      6
(104200.0, 118900.0]     1
(118900.0, 133600.0]     5
(133600.0, 148300.0]     3
(148300.0, 163000.0]     2
Name: average, dtype: int64
```

由上图可知，呈现一个类似正态分布的形状。中间值大约在 20700 到 45400 中间偏大的位置。

3. 分析北京市数据

3.1 数据观测



The file size (3.17 MB) exceeds the configured limit (2.56 MB). Code insight features are not available.

```
No,year,month,day,hour,season,PM_Dongsi,PM_Dongsihuan,PM_Nongzhanguan,PM_US Post,DEWP,HUMI,PRES,TEMP,cbwd,I
1,2010,1,1,0,4,NA,NA,NA,NA,-21,43,1021,-11,NW,1.79,0,0
2,2010,1,1,1,4,NA,NA,NA,NA,-21,47,1020,-12,NW,4.92,0,0
3,2010,1,1,2,4,NA,NA,NA,NA,-21,43,1019,-11,NW,6.71,0,0
4,2010,1,1,3,4,NA,NA,NA,NA,-21,55,1019,-14,NW,9.84,0,0
5,2010,1,1,4,4,NA,NA,NA,NA,-20,51,1018,-12,NW,12.97,0,0
6,2010,1,1,5,4,NA,NA,NA,NA,-19,47,1017,-10,NW,16.1,0,0
7,2010,1,1,6,4,NA,NA,NA,NA,-19,44,1017,-9,NW,19.23,0,0
8,2010,1,1,7,4,NA,NA,NA,NA,-19,44,1017,-9,NW,21.02,0,0
9,2010,1,1,8,4,NA,NA,NA,NA,-19,44,1017,-9,NW,24.15,0,0
10,2010,1,1,9,4,NA,NA,NA,NA,-20,37,1017,-8,NW,27.28,0,0
11,2010,1,1,10,4,NA,NA,NA,NA,-19,37,1017,-7,NW,31.3,0,0
12,2010,1,1,11,4,NA,NA,NA,NA,-18,35,1017,-5,NW,34.43,0,0
13,2010,1,1,12,4,NA,NA,NA,NA,-19,32,1015,-5,NW,37.56,0,0
```

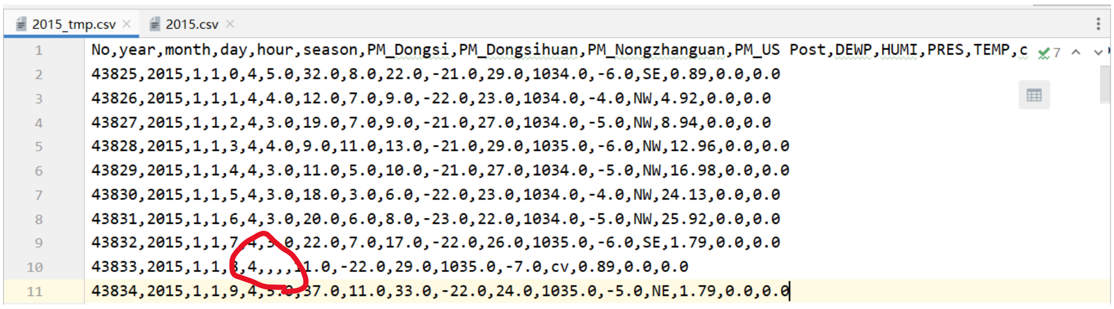
显然 NA 啥的呀 NW 肯定要处理一下。

3.2 代码编写

3.2.1 数据抽取及存储

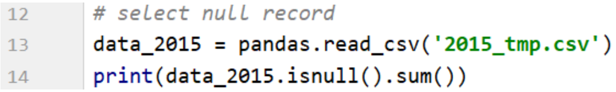
```python
import pandas

# read data
pandas.set_option('display.max_rows',None)
pandas.set_option('display.max_columns',None)
data = pandas.read_csv('BeijingPM20100101_20151231.csv')

# select 2015 into 2015.csv
data_2015 = data[data['year']==2015]
data_2015.to_csv('2015_tmp.csv', index=None)
```

就是把所有的 2015 存到 2015_tmp.csv 中。输出文件如下：



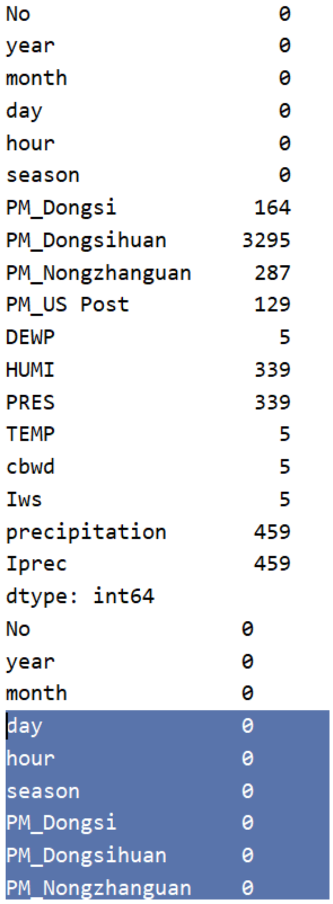
```
No,year,month,day,hour,season,PM_Dongsi,PM_Dongsihuan,PM_Nongzhanguan,PM_US Post,DEWP,HUMI,PRES,TEMP,c
43825,2015,1,1,0,4,5.0,32.0,8.0,22.0,-21.0,29.0,1034.0,-6.0,SE,0.89,0.0,0.0
43826,2015,1,1,1,4,4.0,12.0,7.0,9.0,-22.0,23.0,1034.0,-4.0,NW,4.92,0.0,0.0
43827,2015,1,1,2,4,3.0,19.0,7.0,9.0,-21.0,27.0,1034.0,-5.0,NW,8.94,0.0,0.0
43828,2015,1,1,3,4,4.0,9.0,11.0,13.0,-21.0,29.0,1035.0,-6.0,NW,12.96,0.0,0.0
43829,2015,1,1,4,4,3.0,11.0,5.0,10.0,-21.0,27.0,1034.0,-5.0,NW,16.98,0.0,0.0
43830,2015,1,1,5,4,3.0,18.0,3.0,6.0,-22.0,23.0,1034.0,-4.0,NW,24.13,0.0,0.0
43831,2015,1,1,6,4,3.0,20.0,6.0,8.0,-23.0,22.0,1034.0,-5.0,NW,25.92,0.0,0.0
43832,2015,1,1,7,4,3.0,22.0,7.0,17.0,-22.0,26.0,1035.0,-6.0,SE,1.79,0.0,0.0
43833,2015,1,1,8,4,,,,11.0,-22.0,29.0,1035.0,-7.0,cv,0.89,0.0,0.0
43834,2015,1,1,9,4,5.0,37.0,11.0,33.0,-22.0,24.0,1035.0,-5.0,NE,1.79,0.0,0.0
```

3.2.2 找出空值



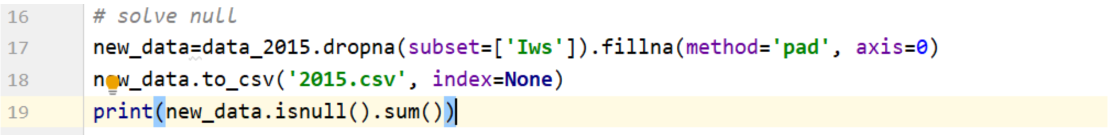
```python
# select null record
data_2015 = pandas.read_csv('2015_tmp.csv')
print(data_2015.isnull().sum())
```

就是把所有空值记录取出输出。



```
No                  0
year                0
month               0
day                 0
hour                0
season              0
PM_Dongsi         164
PM_Dongsihuan    3295
PM_Nongzhanguan   287
PM_US Post        129
DEWP                5
HUMI              339
PRES              339
TEMP                5
cbwd                5
Iws                 5
precipitation     459
Iprec             459
dtype: int64
No               0
year             0
month            0
day              0
hour             0
season           0
PM_Dongsi        0
PM_Dongsihuan    0
PM_Nongzhanguan  0
```

3.2.3 空值处理方法



```python
# solve null
new_data=data_2015.dropna(subset=['Iws']).fillna(method='pad', axis=0)
new_data.to_csv('2015.csv', index=None)
print(new_data.isnull().sum())
```

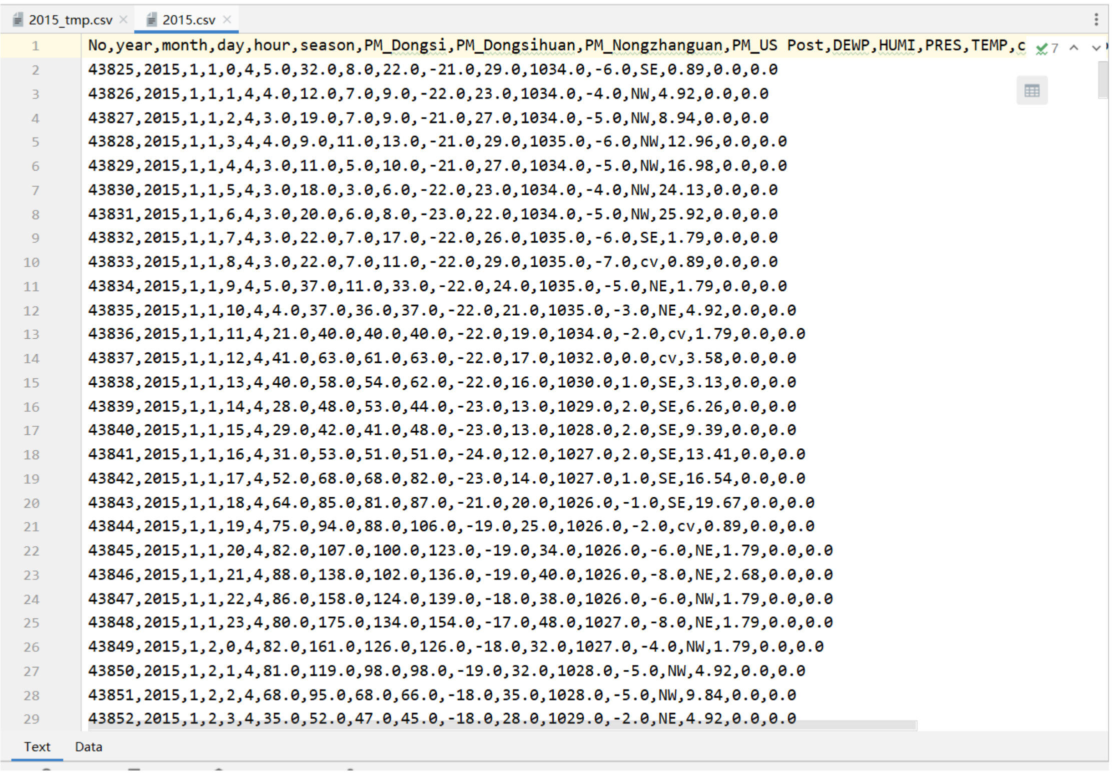
对所有存在空值的列，给出空值的处理方法及理 由，要求处理方法必须可在本数据集范围内执行。

```
No                  0
year                0
month               0
day                 0
hour                0
season              0
PM_Dongsi         164
PM_Dongsihuan    3295
PM_Nongzhanguan   287
PM_US Post        129
DEWP                5
HUMI              339
PRES              339
TEMP                5
cbwd                5
Iws                 5
precipitation     459
Iprec             459
dtype: int64
```

观察原数据，DEWP、TEMP 等空值数量较少的属性的空值都是来自同一组数据，且数量只有 5 组，因此直接将此 5 组数据进行删除。

剩余空值较多的列，考虑到数据采集过程，前后差距不大，默认采用上一条记录数据进行复制填充。

然后把新数据存储到 2015.csv 中。输出文件如下：



```
No,year,month,day,hour,season,PM_Dongsi,PM_Dongsihuan,PM_Nongzhanguan,PM_US Post,DEWP,HUMI,PRES,TEMP,c
43825,2015,1,1,0,4,5.0,32.0,8.0,22.0,-21.0,29.0,1034.0,-6.0,SE,0.89,0.0,0.0
43826,2015,1,1,1,4,4.0,12.0,7.0,9.0,-22.0,23.0,1034.0,-4.0,NW,4.92,0.0,0.0
43827,2015,1,1,2,4,3.0,19.0,7.0,9.0,-21.0,27.0,1034.0,-5.0,NW,8.94,0.0,0.0
43828,2015,1,1,3,4,4.0,9.0,11.0,13.0,-21.0,29.0,1035.0,-6.0,NW,12.96,0.0,0.0
43829,2015,1,1,4,4,3.0,11.0,5.0,10.0,-21.0,27.0,1034.0,-5.0,NW,16.98,0.0,0.0
43830,2015,1,1,5,4,3.0,18.0,3.0,6.0,-22.0,23.0,1034.0,-4.0,NW,24.13,0.0,0.0
43831,2015,1,1,6,4,3.0,20.0,6.0,8.0,-23.0,22.0,1034.0,-5.0,NW,25.92,0.0,0.0
43832,2015,1,1,7,4,3.0,22.0,7.0,17.0,-22.0,26.0,1035.0,-6.0,SE,1.79,0.0,0.0
43833,2015,1,1,8,4,3.0,22.0,7.0,11.0,-22.0,29.0,1035.0,-7.0,cv,0.89,0.0,0.0
43834,2015,1,1,9,4,5.0,37.0,11.0,33.0,-22.0,24.0,1035.0,-5.0,NE,1.79,0.0,0.0
43835,2015,1,1,10,4,4.0,37.0,36.0,37.0,-22.0,21.0,1035.0,-3.0,NE,4.92,0.0,0.0
43836,2015,1,1,11,4,21.0,40.0,40.0,40.0,-22.0,19.0,1034.0,-2.0,cv,1.79,0.0,0.0
43837,2015,1,1,12,4,41.0,63.0,61.0,63.0,-22.0,17.0,1032.0,0.0,cv,3.58,0.0,0.0
43838,2015,1,1,13,4,40.0,58.0,54.0,62.0,-22.0,16.0,1030.0,1.0,SE,3.13,0.0,0.0
43839,2015,1,1,14,4,28.0,48.0,53.0,44.0,-23.0,13.0,1029.0,2.0,SE,6.26,0.0,0.0
43840,2015,1,1,15,4,29.0,42.0,41.0,48.0,-23.0,13.0,1028.0,2.0,SE,9.39,0.0,0.0
43841,2015,1,1,16,4,31.0,53.0,51.0,51.0,-24.0,12.0,1027.0,2.0,SE,13.41,0.0,0.0
43842,2015,1,1,17,4,52.0,68.0,68.0,82.0,-23.0,14.0,1027.0,1.0,SE,16.54,0.0,0.0
43843,2015,1,1,18,4,64.0,85.0,81.0,87.0,-21.0,20.0,1026.0,-1.0,SE,19.67,0.0,0.0
43844,2015,1,1,19,4,75.0,94.0,88.0,106.0,-19.0,25.0,1026.0,-2.0,cv,0.89,0.0,0.0
43845,2015,1,1,20,4,82.0,107.0,100.0,123.0,-19.0,34.0,1026.0,-6.0,NE,1.79,0.0,0.0
43846,2015,1,1,21,4,88.0,138.0,102.0,136.0,-19.0,40.0,1026.0,-8.0,NE,2.68,0.0,0.0
43847,2015,1,1,22,4,86.0,158.0,124.0,139.0,-18.0,38.0,1026.0,-6.0,NW,1.79,0.0,0.0
43848,2015,1,1,23,4,80.0,175.0,134.0,154.0,-17.0,48.0,1027.0,-8.0,NE,1.79,0.0,0.0
43849,2015,1,2,0,4,82.0,161.0,126.0,126.0,-18.0,32.0,1027.0,-4.0,NW,1.79,0.0,0.0
43850,2015,1,2,1,4,81.0,119.0,98.0,98.0,-19.0,32.0,1028.0,-5.0,NW,4.92,0.0,0.0
43851,2015,1,2,2,4,68.0,95.0,68.0,66.0,-18.0,35.0,1028.0,-5.0,NW,9.84,0.0,0.0
43852,2015,1,2,3,4,35.0,52.0,47.0,45.0,-18.0,28.0,1029.0,-2.0,NE,4.92,0.0,0.0
```

4：代码部分

4.1 houses.py

```python
import requests
import parsel
import pandas
import csv
import matplotlib.pyplot as plt

# create outputFile and write header
with open('houses.csv', 'w', newline='', encoding='utf-8') as outputFile:
    f_csv = csv.writer(outputFile)
    f_csv.writerow(['name', 'district', 'town', 'position', 'room', 'area',
'average', 'price'])

# crawl data from front 25 pages
for i in range(1, 25):
    # get data and store into result
    selector =
parsel.Selector(requests.get(f"https://bj.fang.lianjia.com/loupan/pg{i}/").text)
    result = selector.css('.resblock-list.post_ulog_exposure_scroll.has-results')

    for li in result:
        # name
        name = li.css('.resblock-name a::text').get()
        # location
        location = li.css('.resblock-location span::text').getall()
        district = location[0]
        town = location[1]
        position = li.css('.resblock-location a::text').get()
        # size
        room = li.css('.resblock-room span::text').get()
        # area
        area = li.css('.resblock-area span::text').get()
        if area == None:
            continue
        area = area.split(' ')
        area = area[1]
        area = area.split('-')
        if len(area) != 1:
            area = area[0]
        else:
            area = "".join(list(filter(str.isdigit, "".join(area))))

        # average & price
        average = li.css('.main-price span::text').get()
        priceList = average.split('-')
        if len(priceList) == 1:
            price = f"{int(average) * int(area) / 10000:.4f}"
        else:
            price = priceList[0]
            average = int(price) * 10000 / int(area)
        agerage = int(average)

        # write this record into outputFile
        with open('houses.csv', 'a', newline='', encoding='utf-8') as f:
            f_csv = csv.writer(f)
            f_csv.writerow([name, district, town, position, room, area, average,
price])


# analyze data statistics
pandas.set_option('display.max_rows', None)
pandas.set_option('display.max_columns', None)
data = pandas.read_csv('houses.csv', encoding='utf-8')

# price
print("\nPrice most expensive:\n",data[data['price'] == data['price'].max()], '\n')
print("Price most cheap:\n",data[data['price'] == data['price'].min()], '\n')
print("Price middle:\n",data['price'].median())

# average
print("\nAverage most expensive:\n",data[data['average'] == data['average'].max()],
'\n')
print("Average most cheap:\n",data[data['average'] == data['average'].min()], '\n')
print("Average middle:\n",data['average'].median())

# price beyond 3*average
print("\nThose price beyond 3*average:")
mean = data['price'].mean()
std = data['price'].std()
low_border = mean-3*std
high_border = mean+3*std
print(data[(data['price'] > high_border) | (data['price'] < low_border)])
```

```
# Average exception
print("\nThose average exception:")
plt.rcParams['font.sans-serif'] = ['Kaitt', 'SimHei']
data.boxplot(['average'])
plt.show()
q1 = data['average'].quantile(q=0.25)
q3 = data['average'].quantile(q=0.75)
low_limit = q1-1.5*(q3-q1)
high_limit = q3+1.5*(q3-q1)
print(data[(data['average'] > high_limit) | (data['average'] < low_limit)])

# discretization processing
avgMax=data['average'].max()
avgMin=data['average'].min()
num=int(avgMax/avgMin)
cuts = pandas.cut(data['average'], num)
print(pandas.value_counts(cuts).sort_index())
value_list=[]
for i in pandas.value_counts(cuts):
    value_list.append(i)
plt.figure()
index=pandas.value_counts(cuts).index
plt.pie(value_list,labels=index,autopct='%0.2f%%')
plt.show()
```

4.2 BJPM.py

```
import pandas

# read data
pandas.set_option('display.max_rows',None)
pandas.set_option('display.max_columns',None)
data = pandas.read_csv('BeijingPM20100101_20151231.csv')

# select 2015 into 2015.csv
data_2015 = data[data['year']==2015]
data_2015.to_csv('2015_tmp.csv', index=None)

# select null record
data_2015 = pandas.read_csv('2015_tmp.csv')
print(data_2015.isnull().sum())

# solve null
```

```
new_data=data_2015.dropna(subset=['Iws']).fillna(method='pad', axis=0)
new_data.to_csv('2015.csv', index=None)
print(new_data.isnull().sum())
```