

# python 数据处理

姓名：郑金亮

班级：2019211301

学号：2019211168

## 题目一

### 题目要求：

通过爬虫爬取链家的新房数据，并进行预处理

- 最终的csv文件，应包括以下字段：名称，地理位置（3个字段分别存储），房型（只保留最小房型），面积（按照最小值），总价（万元，整数），均价（万元，保留小数点后4位）；
- 对于所有字符串字段，要求去掉所有的前后空格；
- 如果有缺失数据，不用填充。
- 找出总价最贵和最便宜的房子，以及总价的中位数 • 找出单价最贵和最便宜的房子，以及单价的中位数

### 爬虫部分

- 程序框架：
  - `chromedriver.exe`：浏览器驱动，用于本地渲染
  - `scrapy.cfg`：配置
  - `myScrapy`：
    - `spiders`
      - `spider.py`：主要负责对返回的HTML网页进行爬取，并存储到item中
    - `item.py`：定义存储数据结构
    - `middlewares.py`：编写中间件
    - `pipelines.py`：主要负责对spider返回的item进行统一存储管理
    - `setting.py`：框架设置
    - `start.py`：编写的程序运行脚本

`lianjia_spider.py`

根据爬取的html网页和xpath提取的数据并存储到数据结构中

```
1 class LianjiaspiderSpider(scrapy.Spider):
2     name = 'lianjia_spider'
3     allowed_domains = ['bj.fang.lianjia.com']
4     start_urls = ['https://bj.fang.lianjia.com/loupan/']
5
6
7     csv_header = ['name', 'location_area', 'location_town',
8                  'location_exact', 'type', 'area', 'price', 'price_type']
9     def parse(self, response, **kwargs):
10         #爬取当前页面信息并存储
```

```

11         item = HomeItem()
12         for each in response.xpath("/html/body/div[3]/ul[2]/*"):
13             try:
14                 item["name"] = each.xpath("div/div[1]/a/text()").get()
15                 item["location_area"] =
each.xpath("div/div[2]/span[1]/text()").get()
16                 item["location_town"] =
each.xpath("div/div[2]/span[2]/text()").get()
17                 item["location_exact"] =
each.xpath("div/div[2]/a/text()").get()
18                 item["type"] = each.xpath("div/a/span/text()").get()
19                 item["area"] = each.xpath("div/div[3]/span/text()").get()
20                 item["price"] =
each.xpath("div/div[6]/div[1]/span[1]/text()").get()
21                 item["price_type"] =
each.xpath("div/div[6]/div[1]/span[2]/text()").get()
22                 yield item
23             except:
24                 print("get an error:", Exception)
25         if "cur_page" not in kwargs:
26             kwargs["cur_page"] = 1
27
28         #休眠防止触发人机认证
29         time.sleep(5)
30
31
32         #根据网址手动翻页
33         if kwargs["cur_page"] < 22:
34             yield response.follow(url =
"https://bj.fang.lianjia.com/loupan/pg" + str(kwargs["cur_page"] + 1) + '/',
callback = self.parse, cb_kwargs = {'cur_page': kwargs["cur_page"] + 1})

```

item.py

## 定义数据结构

```

1 class HomeItem(scrapy.Item):
2     name = scrapy.Field()
3     location_area = scrapy.Field()
4     location_town = scrapy.Field()
5     location_exact = scrapy.Field()
6     type = scrapy.Field()
7     area = scrapy.Field()
8     price = scrapy.Field()
9     price_type = scrapy.Field()
10    pass

```

middlewares.py

## 编写中间件，在本地加载动态网页

```

1 def process_response(self, request, response, spider):
2     # Called with the response returned from the downloader.
3
4     # Must either;
5     # - return a Response object
6     # - return a Request object

```

```

7         # - or raise IgnoreRequest
8         options = webdriver.ChromeOptions()
9         options.add_argument('--window-position=0,0')
10        options.add_argument('--window-size=1080,800')
11        browser =
webdriver.Chrome(executable_path="D:\\homework\\Python\\myScrapy\\chromedriver.e
xe",chrome_options=options)
12        browser.get(request.url)
13        browser.execute_script('window.scrollTo(0,
document.body.scrollHeight)')
14        # 等待页面加载, 防止触发人机认证
15        browser.implicitly_wait(5)

```

pipelines.py

```

1 class DataProcessPipeline:
2     def open_spider(self, spider):
3         self.file = open("spider_result.csv", "w", encoding="utf-8-sig",
newline='')
4         self.csv_write = csv.writer(self.file)
5         self.csv_write.writerow(spider.csv_header)
6
7     def process_item(self, item, spider):
8         list_item = list(item.values())
9         self.csv_write.writerow(list_item)
10        return item
11
12    def close_spider(self, spider):
13        self.file.close()

```

start.py

```

1 def main():
2     cmdline.execute("scrapy crawl lianjia_spider".split())
3
4 if __name__ == '__main__':
5     main()

```

## 爬取数据截图

name	location_area	location_town	location_exact	type	area	price	price_type
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	建面 185-199m <sup>2</sup>	58000	元/m <sup>2</sup> (均价)
观唐云鼎	密云	溪翁庄镇	溪翁庄镇密溪路39号院 (云佛山度假村对面)	3室	建面 172m <sup>2</sup>	30000	元/m <sup>2</sup> (均价)
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	建面 100-166m <sup>2</sup>	49000	元/m <sup>2</sup> (均价)
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	建面 166-228m <sup>2</sup>	50000	元/m <sup>2</sup> (均价)
首开璞堤公馆	丰台	方庄	紫芳园五区	3室	建面 203-236m <sup>2</sup>	106000	元/m <sup>2</sup> (均价)
华远豪马四季	门头沟	大峪	增产路16号院	3室	建面 156-191m <sup>2</sup>	55000	元/m <sup>2</sup> (均价)
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	建面 300-536m <sup>2</sup>	40000	元/m <sup>2</sup> (均价)
华远和墅	大兴	南中轴机场商务区	南六环磁各庄桥沿南中轴向南2公里	5室	建面 295m <sup>2</sup>	54000	元/m <sup>2</sup> (均价)
天资华府	房山	长阳	房山区CSD政务大厅5号门	3室	建面 115-293m <sup>2</sup>	38000	元/m <sup>2</sup> (均价)
檀香府	门头沟	门头沟其它	京潭大街与潭柘十街交叉口	3室	建面 208-320m <sup>2</sup>	45000	元/m <sup>2</sup> (均价)
韩建·观山源墅	房山	良乡	阳光北大街与多宝路交汇处西南 (理工大学北校区西侧)	3室	建面 290-330m <sup>2</sup>	40000	元/m <sup>2</sup> (均价)
首城汇景墅	平谷	平谷其它	金河北街6号院, 金河北街8号院	3室	建面 360m <sup>2</sup>	25000	元/m <sup>2</sup> (均价)
中国铁建花语金郡	大兴	瀛海	南海子公园西侧(南五环旧忠桥向南第二个红绿灯西300米)	3室	建面 150m <sup>2</sup>	70000	元/m <sup>2</sup> (均价)
北辰墅院1900	顺义	马坡	顺兴街11号院望尊园	4室	建面 251-282m <sup>2</sup>	42000	元/m <sup>2</sup> (均价)
首创天阅西山	海淀	海淀北部新区	海淀区丰秀东路9号院, 永丰路与北清路交汇处东北角, 中关村壹号北	4室	建面 175-245m <sup>2</sup>	80000	元/m <sup>2</sup> (均价)

翡翠公园	昌平	北七家	北七家京承高速北七家出口向西3公里，七星路与七北路交汇处	4室	建面 98-140m <sup>2</sup>	61000 元/m <sup>2</sup> (均价)
北科建泰禾丽春湖院子	昌平	沙河	中关村北延新核心，沙河水库边（地铁昌平线沙河站向南800米）	4室	建面 379-800m <sup>2</sup>	50000 元/m <sup>2</sup> (均价)
绿地海珀云翡	大兴	大兴其它	兴亦路京开高速东侧（黄村镇第一中心小学对面）	2室	建面 102-182m <sup>2</sup>	65000 元/m <sup>2</sup> (均价)
郁丽华府	平谷	平谷其它	新平南路与林荫南街交汇处向西100米	2室	建面 94-283m <sup>2</sup>	29000 元/m <sup>2</sup> (均价)
中粮京西祥云	房山	长阳	地铁稻田站北800米，西邻京深路	4室	建面 115-140m <sup>2</sup>	58000 元/m <sup>2</sup> (均价)
燕西华府	丰台	丰台其它	王佐镇青龙湖公园东1500米，	4室	建面 60-288m <sup>2</sup>	42000 元/m <sup>2</sup> (均价)
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	建面 122-153m <sup>2</sup>	43000 元/m <sup>2</sup> (均价)
紫辰院	丰台	岳各庄	岳各庄北桥东北角200米处	5室	建面 266-345m <sup>2</sup>	128000 元/m <sup>2</sup> (均价)
鲁能格拉斯小镇	通州	通州其它	北京市通州区宋庄镇格拉斯小镇营销中心	3室	建面 246-850m <sup>2</sup>	60000 元/m <sup>2</sup> (均价)
兴创荣墅	大兴	大兴新机场洋房别	北京市大兴区育胜街	3室	建面 240-411m <sup>2</sup>	23000 元/m <sup>2</sup> (均价)
温哥华森林	昌平	北七家	北五环外紧邻立汤路，北七家建材城北第一个路口200米路东，枫树家4室	建面 460-661m <sup>2</sup>	43478 元/m <sup>2</sup> (均价)	
润泽御府	朝阳	北苑	北京市朝阳区北五环顾家庄桥向北约2.6公里	4室	建面 540-1600m <sup>2</sup>	110000 元/m <sup>2</sup> (均价)
中骏西山天璟	门头沟	城子	西山永定楼北300米	4室	建面 117-190m <sup>2</sup>	65000 元/m <sup>2</sup> (均价)
炫立方	顺义	顺义其它	金关北二路2号院			30000 元/m <sup>2</sup> (均价)
国瑞熙墅	昌平	北七家	北七家镇岭上西路与定泗路交汇处东南角	3室	建面 314-457m <sup>2</sup>	48000 元/m <sup>2</sup> (均价)

## 数据处理部分

```
1 import pandas as pd
2 import numpy as np
3 import re
```

- 字符串数据处理和空格删除：

```
1 # 1.打开csv文件
2 file_str = "spider_result.csv"
3 df = pd.read_csv(file_str, encoding='utf-8-sig')
4
5
6 # 2.去除csv文件中的空行
7 df = df.dropna(axis = 0, how="all")
8
9 # 3.删除所有字符串字段的前后空格 - name - location_area - location_town -
location_exact - type
10 df['name'] = list(map(lambda x: x.strip() if isinstance(x, str) else x,
df['name']))
11 df['location_area'] = list(map(lambda x: x.strip() if isinstance(x, str)
else x, df['location_area']))
12 df['location_town'] = list(map(lambda x: x.strip() if isinstance(x, str)
else x, df['location_town']))
13 df['location_exact'] = list(map(lambda x: x.strip() if isinstance(x,
str) else x, df['location_exact']))
14 df['type'] = list(map(lambda x: x.strip() if isinstance(x, str) else x,
df['type']))
```

- 面积和价格字段，取最小值

```
1 def get_digit(s):
2     '''
3     overview:
4         提取字符串s中的所有数字，并组成列表返回
5     example:
6         s = '12and345and678'
7         return [12, 345, 678]
8     '''
9     return list(map(int, re.findall(r'\d+', s)))
```

```

1 # 4. 处理面积, 提取字符串中数字并保留最小值
2 index = df.dropna(axis = 0, how="any", subset=['area']).index
3 df.loc[index, 'area'] = list(map(lambda x: min(get_digit(x)),
4                                 df.loc[index, 'area']))
5
6 # 4. 处理价格
7 index = df.dropna(axis = 0, how="any", subset=['price']).index
8 df.loc[index, 'price'] = list(map(lambda x: min(get_digit(x)),
9                                 df.loc[index, 'price']))

```

- 价格处理

```

1 def get_price(kwarg):
2     if "万" in kwarg[2]: # 单位换算
3         kwarg[0] *= 10000
4     if "均价" in kwarg[2]:
5         return int(kwarg[0]*kwarg[1]), kwarg[0]
6     else:
7         return kwarg[0], kwarg[0]/kwarg[1]

```

```

1 # 5. 计算总结和均价
2 index = df.dropna(axis = 0, how="any", subset=['area', 'price',
3         'price_type']).index
4 price = np.array(list(map(get_price, np.array([df.loc[index, 'price'],
5         df.loc[index, 'area'], df.loc[index, 'price_type']))).transpose((1,0))))
6 tot_price = price[:, 0]
7 avg_price = price[:, 1]
8 df.loc[index, 'tot_price'], df.loc[index, 'avg_price'] = tot_price,
9         avg_price
10
11 # 6. 总价为整数, 均价保留四位小数, 删除列price, price_type
12 df.loc[index, 'tot_price'] = df.loc[index, 'tot_price'].astype('int')
13 df.loc[index, 'avg_price'] = df.loc[index, 'avg_price'].round(4)
14 df = df.drop(columns=['price', 'price_type'])

```

- 结果输出及文件保存

```

1 # 7. 输出统计信息
2 tot_max_id = df.loc[index, 'tot_price'].idxmax()
3 tot_min_id = df.loc[index, 'tot_price'].idxmin()
4 tot_mid = df.loc[index, 'tot_price'].describe()["50%"]
5
6 avg_max_id = df.loc[index, 'avg_price'].idxmax()
7 avg_min_id = df.loc[index, 'avg_price'].idxmin()
8 avg_mid = df.loc[index, 'avg_price'].describe()["50%"]
9
10 print("总价最高", df.loc[tot_max_id])
11 print("总价最低", df.loc[tot_min_id])
12 print("中位数", tot_mid)
13
14 print("单价最高", df.loc[avg_max_id])
15 print("单价最低", df.loc[avg_min_id])

```

```

16 print("中位数", avg_mid)
17
18
19 # 保存最后结果
20 df.to_csv("process_result.csv")

```

## 数据处理结果

```

总价最高 name                    北京壹号总部
location_area                    大兴
location_town                    亦庄
location_exact                    台湖镇光机电一体化产业基地科创东二街5号
type                            1室
area                            3127
tot_price                        87556000.0
avg_price                        28000.0
Name: 274, dtype: object
总价最低 name                    长海御墅
location_area                    房山
location_town                    房山其它
location_exact                    长沟国家湿地公园南侧
type                            1室
area                            70
tot_price                        1050000.0
avg_price                        15000.0
Name: 292, dtype: object
中位数 5520000.0
单价最高 name                    北京庄园
location_area                    顺义
location_town                    顺义其它
location_exact                    京承高速第11出口往东800米
type                            4室
area                            460
tot_price                        76820000.0
avg_price                        167000.0
Name: 256, dtype: object
单价最低 name                    长海御墅
location_area                    房山
location_town                    房山其它
location_exact                    长沟国家湿地公园南侧
type                            1室
area                            70
tot_price                        1050000.0
avg_price                        15000.0
Name: 292, dtype: object
中位数 47000.0

```

## 题目二

### 题目要求:

1. 汇总计算PM指数年平均值的变化情况
2. 汇总计算10-15年PM指数和温度月平均数据的变化情况

## 数据观察与要求分析:

1. 数据中有四列与PM相关数据与一列与温度相关数据
2. 根据题目要求需要对数据针对年, 月分组

- 以如下逻辑进行数据处理

- 将四列PM值均加入求均值
- 忽略所有nan值

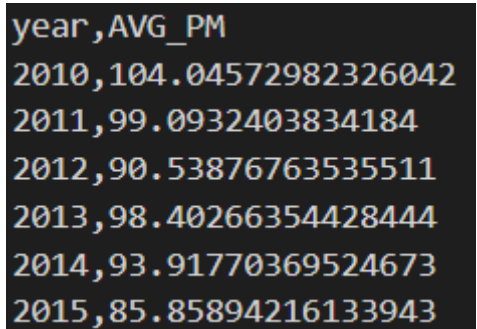
## 源代码

```
1 import numpy as np
2 import pandas as pd
3 from matplotlib import pyplot as plt
4
5 # 1. 打开CSV文件, 创建结果文件
6 fileNameStr = 'BeijingPM20100101_20151231.csv'
7 df = pd.read_csv(fileNameStr, encoding='utf-8')
8
9 # 2. 分组求PM年平均值的变化情况 - 分三地分别处理
10 df['AVG_PM'] = df[['PM_Dongsi', 'PM_Dongsihuan', 'PM_Nongzhanguan', 'PM_US
    Post']].mean(axis=1)
11 year_result = df[['year', 'AVG_PM']].groupby('year')['AVG_PM'].mean()
12
13 # 3. 分组求PM及温度月平均的变化情况
14 month_result = df[['year', 'month', 'AVG_PM', 'TEMP']].groupby(['year',
    'month'])[['AVG_PM', 'TEMP']].mean()
15
16 # 4. 将结果保存至文件
17 year_result.to_csv("year_PM_result.csv")
18 month_result.to_csv("month_PM_result.csv")
19
20 year_result.plot()
21 plt.savefig('year_result.png')
22 month_result.plot()
23 plt.savefig('month_result.png')
```

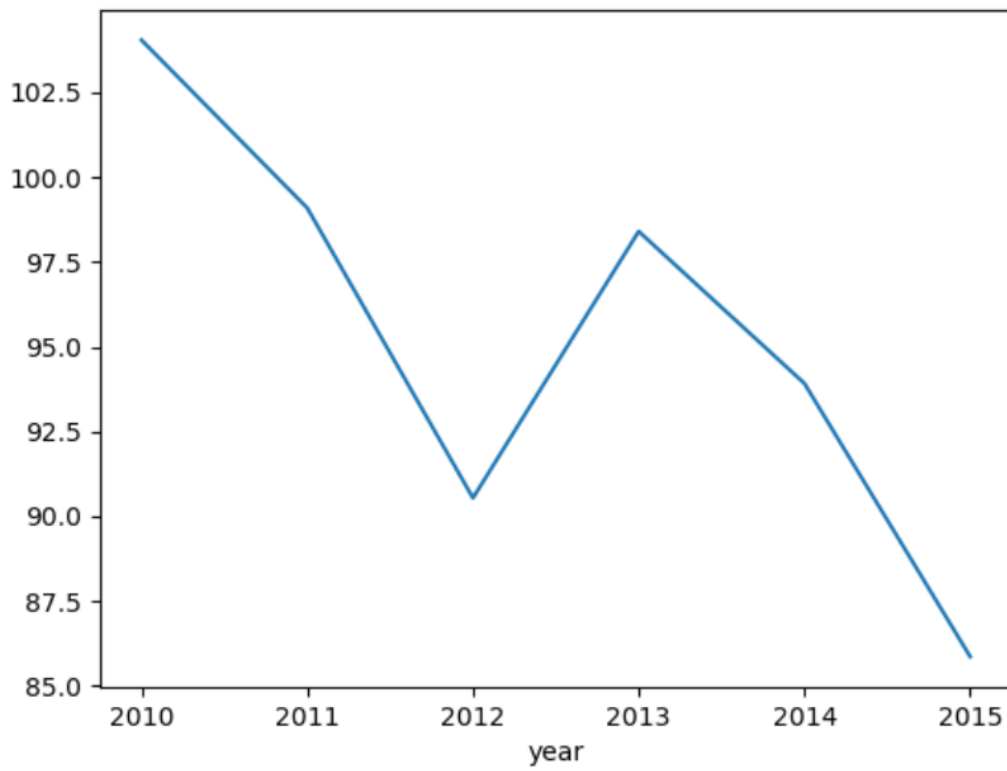
## 结果分析

### 1. 年数据

- CSV文件截图



year	AVG_PM
2010	104.04572982326042
2011	99.0932403834184
2012	90.53876763535511
2013	98.40266354428444
2014	93.91770369524673
2015	85.85894216133943



**分析:**

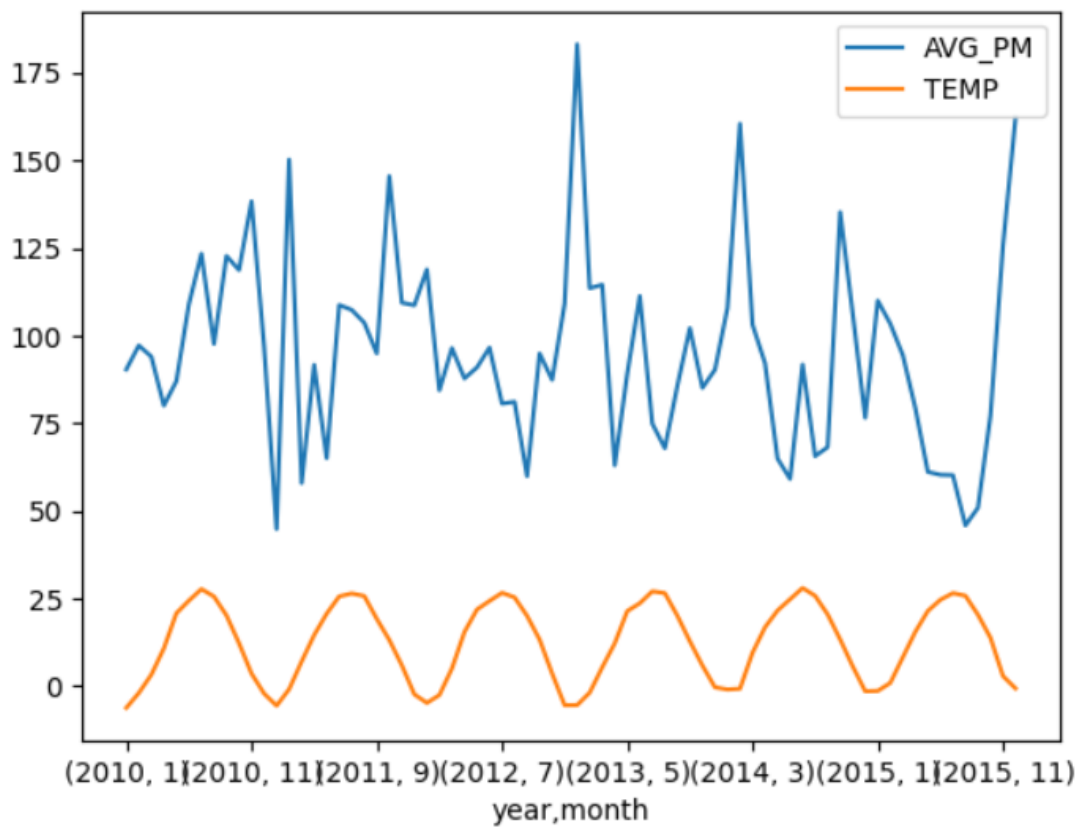
PM指数在2013年有所回弹，但总体呈现下降趋势，指数年均值位于80-110之间

2. 月数据

(截图前30条数据)



```
year,month,AVG_PM,TEMP
2010,1,90.40366972477064,-6.162634408602151
2010,2,97.23994038748137,-1.9226190476190477
2010,3,94.04654442877292,3.293010752688172
2010,4,80.0724233983287,10.806944444444444
2010,5,87.0719131614654,20.831989247311828
2010,6,109.03893805309734,24.434722222222224
2010,7,123.4260752688172,27.72983870967742
2010,8,97.68343195266272,25.611559139784948
2010,9,122.79273504273505,20.21388888888889
2010,10,118.78436657681941,12.299731182795698
2010,11,138.38403614457832,3.609722222222222
2010,12,97.1157469717362,-2.064516129032258
2011,1,44.87369985141159,-5.553763440860215
2011,2,150.29017857142858,-0.8541666666666666
2011,3,57.99198717948718,7.068548387096774
2011,4,91.72067039106145,14.605555555555556
2011,5,65.10814606741573,20.713709677419356
2011,6,108.79465541490858,25.648611111111112
2011,7,107.38648648648649,26.469086021505376
2011,8,103.7338003502627,25.758064516129032
2011,9,94.96940194714882,19.231944444444444
2011,10,145.5568181818182,13.209677419354838
2011,11,109.43496503496503,5.980555555555555
2011,12,108.72139973082099,-2.3024193548387095
2012,1,118.92238805970149,-4.758064516129032
2012,2,84.44202898550725,-2.5114942528735633
2012,3,96.47432432432433,5.07258064516129
2012,4,87.83588317107093,15.473611111111111
```



#### 分析:

PM指数和TEMP数值均与季度有紧密联系，从月数据上较难看出总体趋势，PM指数可以看出波动逐渐加大，TEMP波动逐渐减少。