数据来源：房天下

主要爬取房天下中，铜川二手房信息，网址为：<https://tongchuan.esf.fang.com/house/i32/>



一共有455套二手房信息



在页面上有这样的信息，我们再看看详情信息，



总共就有这些信息，我们需要批量爬取以上信息，需要借助一些工具

需要工具:

1. python模块中的requests模块，借助这个模块，我们设置好headers，可有模拟浏览器对网站进行访问，这样就不容易被网站认为我们是自动化工具，再利用time模块的定时功能，避免访问速度过快，ip容易被封。
2. 借助bs4中的BeautifulSoup模块，解析response对象，将它解析成xml格式，之后借助find()和find\_all()方法，找到我们需要的数据的标签，获取标签的内容
3. 借助pandas库，将我们获取到的数据保存成DataFrame类型，之后将DataFrame输出到csv文件，保存。

开始分析：

1. 首先我们二手房的数据有455条



一共有8页，我们爬取4页即可，那我们就先要分析页码和网址的关系，先看第一页的网址，

第一页网址为：<https://tongchuan.esf.fang.com/>

第二页网址为：<https://tongchuan.esf.fang.com/house/i32/>

第三页网址为：<https://tongchuan.esf.fang.com/house/i33/>

这样我们就可以发现网址和页码的关系了，只有一个数字发生了变化

之后设置headers

打开f12，如图所示





获取cookie和user-agent，设置好以后，就可以开始写代码抓取数据了



在上图，我们可以看到房源信息的网址，通过find()方法获取即可。

之后访问房源详情界面，在界面上获取主要信息



开始爬取，出现问题，无论爬取什么内容，最后显示都是[]，也就是空，意味着我们并没有爬取到数据，根据我们的观察，我们访问的网址并不是数据存储的网址

打开f12看看，找到网站详情信息



看到这个红色框中的网址，通过浏览器访问，结果如下



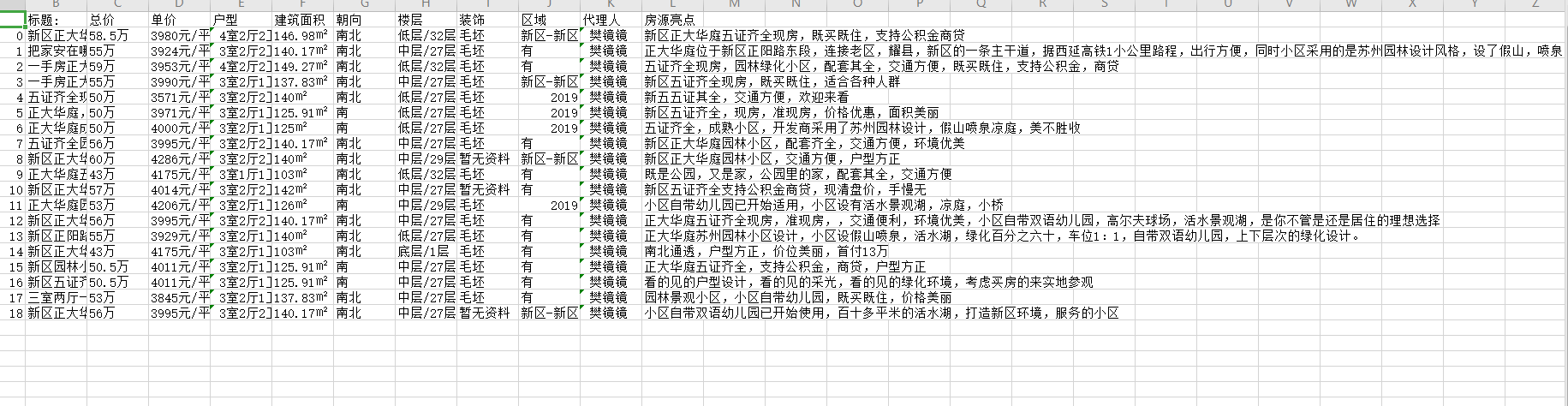
发现我们需要的数据，尝试这个网站能否爬取，最后验证成功，可以通过这个网站获取我们需要的数据。

有些房源信息如下：



我们可以做一个判断，如果字符串“很抱歉，该房源已删除”在html中，则我们就跳出这条数据爬取循环，爬取下一条数据。

查看爬取好的数据



发现数据量并不是我们所要的，这是为什么呢？

我们需要做一下测试，测试发现，除了上面的十几条数据是存在的，其他的房源网站都是提示抱歉，已删除，我们需要重新对网址进行分析

发现有些房源详情网站格式是这样的，https://tongchuan.esf.fang.com/chushou/3\_154052145.htm（命名类型3）

而有些房源网站是这样的https://tongchuan.esf.fang.com/chushou/16\_201374.htm（命名类型16）

它们网址结构苏安然不一样，但是我们都可以获取他的网址，这并不影响，之后我们在去看它们标签元素的结构是否一样

先看第一种，



再看第二种，



很显然，它们标题的标签结构完全不一样，所以我们需要对分开进行分析

除了小区、区域、标题、房源亮点外，其他数据的标签结构都是一样的，可以以同一种方式获取，而像小区、区域、标题、房源亮点，我们就需要先判断是类型3还是类型16的网址，之后在根据不同结构获取数据。

数据爬取过程图片：



爬取过程中又遇到一个问题，有些房源信息有亮点信息，有些没有亮点信息，我们就需要判断它是否有亮点信息，如果没有，就将亮点信息设置为无。如下图



有些房源是没有区域信息的，如下



我们也需要进行判断，如果没有就设置为无。

之后查看我们获取到的数据



之后进行数据分析与处理

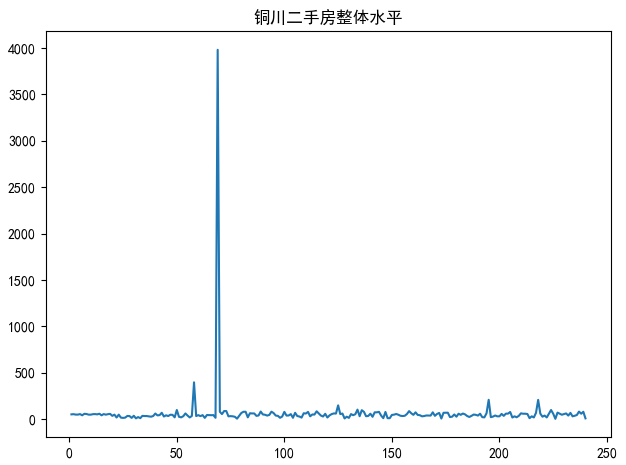
数据分析：

1铜川房价的总体水平如何

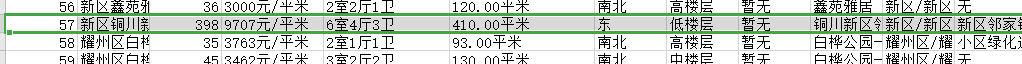
代码如下：

def get\_tongchuan\_price\_sum(data):  
 price\_sum = data['总价'].values  
 x\_label = np.linspace(1,len(price\_sum),len(price\_sum))  
 plt.plot(x\_label,price\_sum)  
 plt.title('铜川二手房整体水平')  
 plt.show()

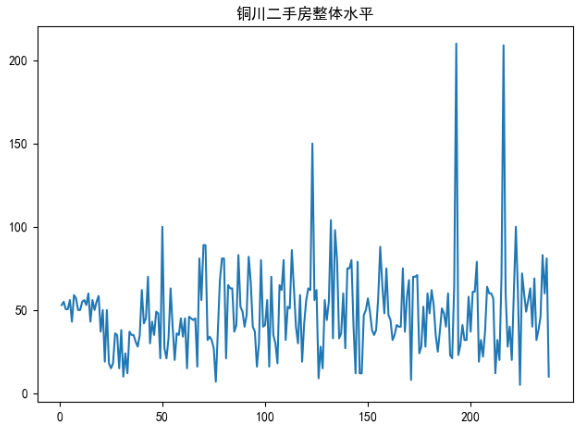
结果如下：



普遍低于500万，有一个差不多4000万，我们回到数据中查看，如下图



这个数据应该是有问题的，我们把它删掉，因为一平方米40万还夸张了，之后再运行代码



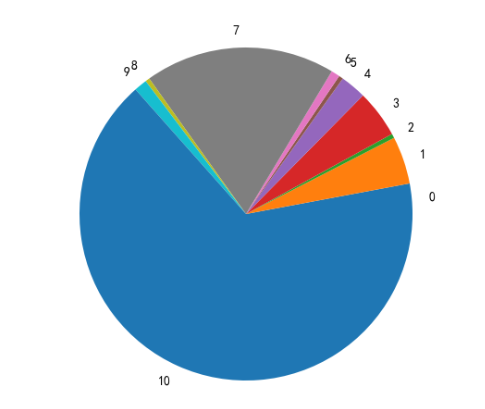
这个就比较正常了，大部分房价在100万以下。

1. 房源朝向分析

代码如下：

def get\_orientation(data):  
 orientation\_map = {vol:ii for ii,vol in enumerate(set(data['朝向']))}  
 print(orientation\_map)  
 data['朝向'] = data['朝向'].map(orientation\_map)  
 orientation\_count\_data = data.groupby(['朝向'], as\_index=False)['朝向'].agg({'cnt': 'count'})  
 orientation = orientation\_count\_data['朝向']  
 count = orientation\_count\_data['cnt']  
 plt.pie(count,labels=orientation)  
 plt.show()

结果如下：



这个0,1,2,3，对应着朝向类型，标签说明{'东': 0, '东南': 1, '暂无': 2, '东西': 3, '西南': 4, '东北': 5, '西': 6, '南': 7, '西北': 8, '北': 9, '南北': 10}。

这个0占得面积最大，也就是南北，说明这些房源大多朝向是南北。

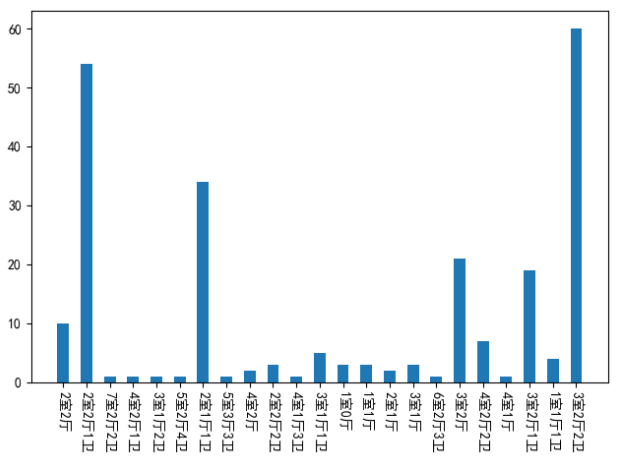
1. 热门户型

我们需要以户型作为索引，然后统计这个户型下所有房源个数的总和，之后以柱形图的形式展现出来。

代码如下：

def get\_hot\_house\_type(data):  
 house\_type\_list = list(set(data['户型']))  
 count = []  
 for house\_type in house\_type\_list:  
 count.append(len(data[data['户型']==house\_type]['户型'].values))  
 plt.bar(house\_type\_list, count,width=0.5)  
 plt.xticks(rotation=270)  
 plt.show()

结果如下：



可以看到，大多数房源属于两房两厅和三房两厅的户型。

1. 区域房价分析

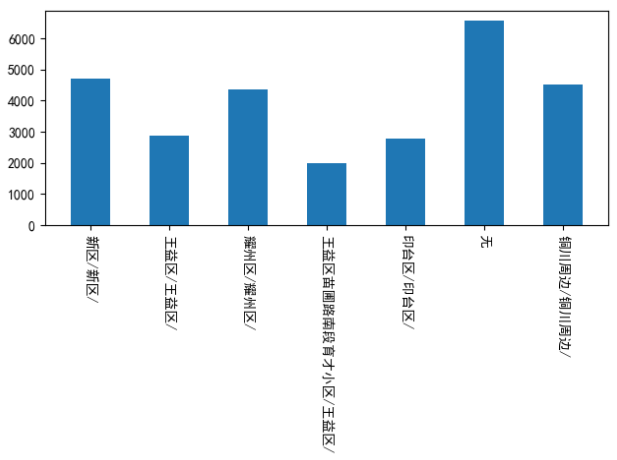
我们分析的是每个区域的平均单价，因为我们爬取的单价数据是字符串，我们需要对它进行处理，删除字符串数据，然后将数据转为浮点型数据数据处理代码如下：

def set\_price\_sum(price\_string):  
 return float(price\_string.replace('元/平米',''))

借助pandas中的apply方法，遍历每一个单价数据进行处理，之后就可以将区域平均单价可视化，代码如下：

def get\_aver\_region(data):  
 region\_list = list(set(data['区域']))  
 aver\_price\_list = []  
 for region in region\_list:  
 aver\_price\_list.append(sum((data[data['区域']==region]['单价'].values)/(len((data[data['区域']==region]['单价'].values)))))  
 plt.bar(region\_list, aver\_price\_list,width=0.5)  
 plt.xticks(rotation=270)  
 plt.show()

结果如下：



数据处理：

1. 单价数据处理

我们在分析区域房价的时候，已经对这个单价数据进行处理了。

1. 户型数据处理

我们需要将户型数据转变为数字，也就是将几房几厅转为0,1,2,3，之类的数字，先创建个字典，然后根据字典键值修改数据

1. 建筑面积处理

建筑面积数据和房源单价数据差不多，数据中多了一部分字符串，我们需要将字符串删掉然后将数据转为浮点型数据

1. 朝向、装饰和户型数据处理

朝向数据、装饰数据和户型数据，我们都可以按照户型数据处理的方式转变为数字，同理，创建字典，然后根据键值索引修改数据值。

数据处理代码：

def set\_data(data):  
 def set\_covered\_area(area\_string):  
 return float(area\_string.replace('平米',''))  
 orientation\_map = {vol:ii for ii,vol in enumerate(set(data['朝向']))}  
 data['朝向'] = data['朝向'].map(orientation\_map)  
 floor\_map = {vol:ii for ii,vol in enumerate(set(data['层']))}  
 data['层'] = data['层'].map(floor\_map)  
 decoration\_map = {vol:ii for ii,vol in enumerate(set(data['装饰']))}  
 data['装饰'] = data['装饰'].map(decoration\_map)  
 data['建筑面积'] = data['建筑面积'].apply(set\_covered\_area)  
 house\_type\_map = {vol:ii for ii,vol in enumerate(set(data['户型']))}  
 data['户型'] = data['户型'].map(house\_type\_map)  
 return data

模型建立：

1特征选取：

我们可以先查看数据之间的相关性

代码如下：

def get\_correlation\_coefficient(data):  
 cols = ['总价', '单价', '户型', '建筑面积', '朝向','层','装饰']  
cm = np.corrcoef(data[cols].values.T)#corrcoef方法按行计算皮尔逊相关系数,cm是对称矩阵  
sns.set(font='SimHei',font\_scale=0.9) #font\_scale设置字体大小  
sns.set(style='whitegrid', context='notebook')  
hm = sns.heatmap(cm, cbar=True, annot=True,  
 cbar\_ax=None,ax=None,  
 square=False,  
 fmt='.2f',  
 linewidths=0.05,  
 annot\_kws={'size': 12},  
 yticklabels=cols,  
 xticklabels=cols)  
plt.show()

相关性图如下：



可以看到单价和总价、户型、建筑面积和层关系较大，但是我们的单价是根据总价和建筑面积获取的，所以我们选取特征的时候不能选取总价。也就是我们最后的特征为户型、建筑面积、朝向、层和装饰数据。

1. 训练集测试集划分

我们在模型训练之前，需要划分好数据集，百分之三十是测试集，百分之七十是训练集

代码如下：

x\_train,t\_test,y\_train,y\_test = train\_test\_split(x\_data,y\_data,test\_size=0.3,random\_state=30)

1. 模型训练

我们建立随机森林回归模型

代码如下：

def train\_test(data):  
 feature = ['户型', '建筑面积', '朝向','层','装饰']  
 x\_data = data[feature]  
 y\_data = data['单价']  
 x\_train,t\_test,y\_train,y\_test = train\_test\_split(x\_data,y\_data,test\_size=0.3,random\_state=30)  
 model = RandomForestRegressor()  
 model.fit(x\_train,y\_train)  
 return model

之后使用训练好的模型去测试测试集数据，对比真实数据与模型预测结果

代码如下：

model,x\_test,y\_test = train\_test(data)  
y\_pred = model.predict(x\_test)  
x\_label = np.linspace(1,len(y\_pred),len(y\_pred))  
plt.plot(x\_label,y\_test)  
plt.plot(x\_label,y\_pred)  
plt.legend(['真实值','预测值'])  
plt.show()

预测结果和真实值对比图



模型拟合效果还不错。