上海市二手房市场价格影响因素探究

——基于分类模型和文本挖掘

吕港 2015111600

2018年6月7日星期四

目录

[摘要: 5](#_Toc503171502)

[一、问题描述 5](#_Toc503171503)

[二、数据收集、说明及处理 5](#_Toc503171504)

[（1）数据源选择 5](#_Toc503171505)

[（2）数据说明 6](#_Toc503171506)

[（3）数据收集 6](#_Toc503171507)

[（3）数据处理 7](#_Toc503171508)

[1．爬取过程中的处理 7](#_Toc503171509)

[2．数据爬取后的处理 8](#_Toc503171510)

[3．文本挖掘数据处理 9](#_Toc503171511)

[三、描述性分析 11](#_Toc503171512)

[四、文本挖掘 15](#_Toc503171513)

[五、分类采用的模型及原因 16](#_Toc503171514)

[（1）线性回归模型 16](#_Toc503171515)

[（2）神经网络 16](#_Toc503171516)

[（3）支持向量机 16](#_Toc503171517)

[六、分类模型使用python的Sklearn库 16](#_Toc503171518)

[七、数据挖掘建模过程 17](#_Toc503171519)

[（1）数据特征分析 17](#_Toc503171520)

[（2）模型调参 17](#_Toc503171521)

[（3）结果分析及模型对比 17](#_Toc503171522)

[八、非线性模型建模 18](#_Toc503171523)

[（1）非线性决策树 18](#_Toc503171524)

[（2）结论 19](#_Toc503171525)

[九、房价查询界面 19](#_Toc503171526)

[十、缺陷和改进措施 20](#_Toc503171527)

[10.1缺陷： 20](#_Toc503171528)

[10.2改进措施 21](#_Toc503171529)

[（1）收集更多数据 21](#_Toc503171530)

[（2）寻找更多特征 22](#_Toc503171531)

[（3）分析方法的结合 22](#_Toc503171532)

[十一、附录 23](#_Toc503171533)

[（1） 爬虫代码 23](#_Toc503171535)

[（2） 分类模型和Python的Gui图形界面代码 27](#_Toc503171536)

[（3） 上海市二手房描述性分析（R） 31](#_Toc503171537)

[（4） 上海市二手房文本挖掘数据清洗（R） 33](#_Toc503171538)

[（5）上海市二手房文本挖掘分析建模（R） 34](#_Toc503171539)

# 摘要:

本文主要分析影响上海市二手房市场房价的影响因素，数据来源为链家网，分类模型的使用中，采用了三种线性模型、一种非线性模型，以及对每个房源的介绍进行了文本挖掘和对自变量和因变量进行可视化处理，最后得到在上海市二手房市场中：房子的大小、房子的位置、房子的建造年份以及房子的高度对房价影响较大；房子的单价与房子的总面积整体上呈现反比与上海每个区的房价的箱线图；二手房的 “交通优势”、“户型优势”对房子的单价和总价有着较大影响。

# 一、问题描述

随着近十年来中国房产市场中的供不应求的现象的影响，导致中国房价持续高涨，特别是上海、北京等一线城市，房价更是高的离谱，令人震叹。那么搞清楚在中国房产市场中决定一个房子的价格的因素中，哪些因素占了主要的地位，将对于我们选购房子和避免花冤枉钱有着积极、重要的作用。通过这些因素和信息又如何让想买房的人快速获取大概的房价信息也是本文的主要解决的问题。针对以上问题本文介绍的就是如何用分类模型和文本挖掘去训练上海房价信息并生成模型然后进行分析的过程。

# 二、数据收集、说明及处理

## （1）数据源选择

通过在网上对几个主流的房产信息网站的二手房价格的比较(如图)，



安居客二手房房价信息



房天下二手房房价信息



链家二手房房价信息

可以看到，其实各个房产网站展示的二手房的房价信息差别不大，另外根据百度上查到链家网的房价与实际的房价差距较小，所以就把链家网的房价数据作为数据源，以供爬虫爬取信息。

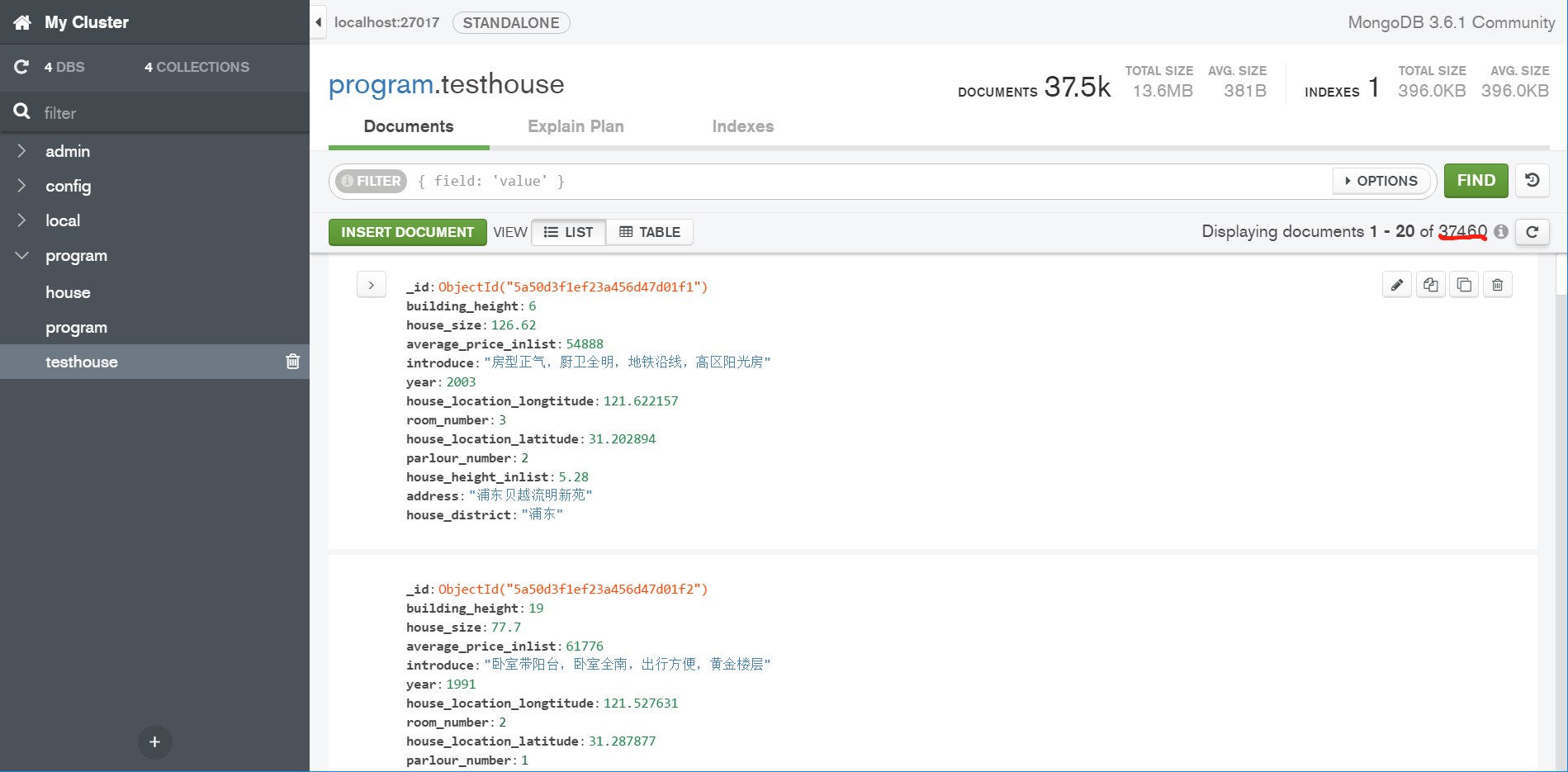
本文采用上海市二手房作为数据集的主要原因在于：目前上海市的新房房源数量较少。链家网上上海市的新房房源信息只有约400套，数据集过小，不适合进行训练和挖掘，所以选择了二手房信息作为数据集进行训练与挖掘。在链家网上，上海二手房数据有几万套，可以进行数据挖掘从而得到有用的信息。

## （2）数据说明



## （3）数据收集

使用爬虫从链家网上爬取房价信息的数据，爬虫使用python编写，并存入mongoDB数据库，以备之后的训练，共计爬取二手房房源37460条。（爬虫代码见附录，画红线的为爬取的二手房房源的总计，详细数据可见电子版的secondHouse.csv文件。）



## （3）数据处理

### 1．爬取过程中的处理

由之前各个房产网站的二手房房源信息的截图可见，房价信息数据还不能直接用于数据挖掘，所以在爬取信息时对其中的一些数据进行处理来使得更适合挖掘。

首先是房型



为了更好的分类，我们把房型拆分为几个卧室，几个客厅，只保留数字。



对于楼层高度，总共有6层高，房子位于中层(中区)，那么就用6\*0.5表示房子的高度。同样，对于低层和高层的房子，分别用楼高乘以0.2和0.8表示房子的高度。



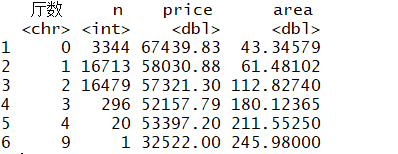
对于地址位置，调用了高德地图的地址查询api把它转换成了利于计算的经纬度信息。

其他的数据信息都比较标准，在爬取之后不需要进行额外的处理。

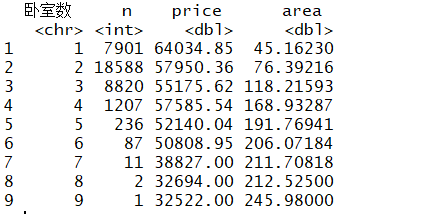
### 2．数据爬取后的处理

先将爬取并存入到mongoDB数据库中的数据导入到一个csv文件中，运用R进行数据处理。而处理的方式针对模型的应用有所不同，本文中的处理分别针对数据挖掘中的分类模型和文本挖掘两类。（R代码见附录）

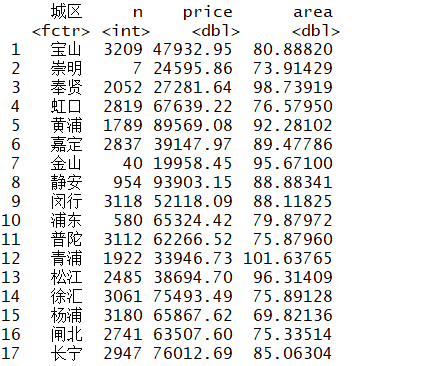
（1）数据挖掘的分类模型数据预处理



对房源的厅数进行分析，将3厅及3厅以上的数据合并为3+厅



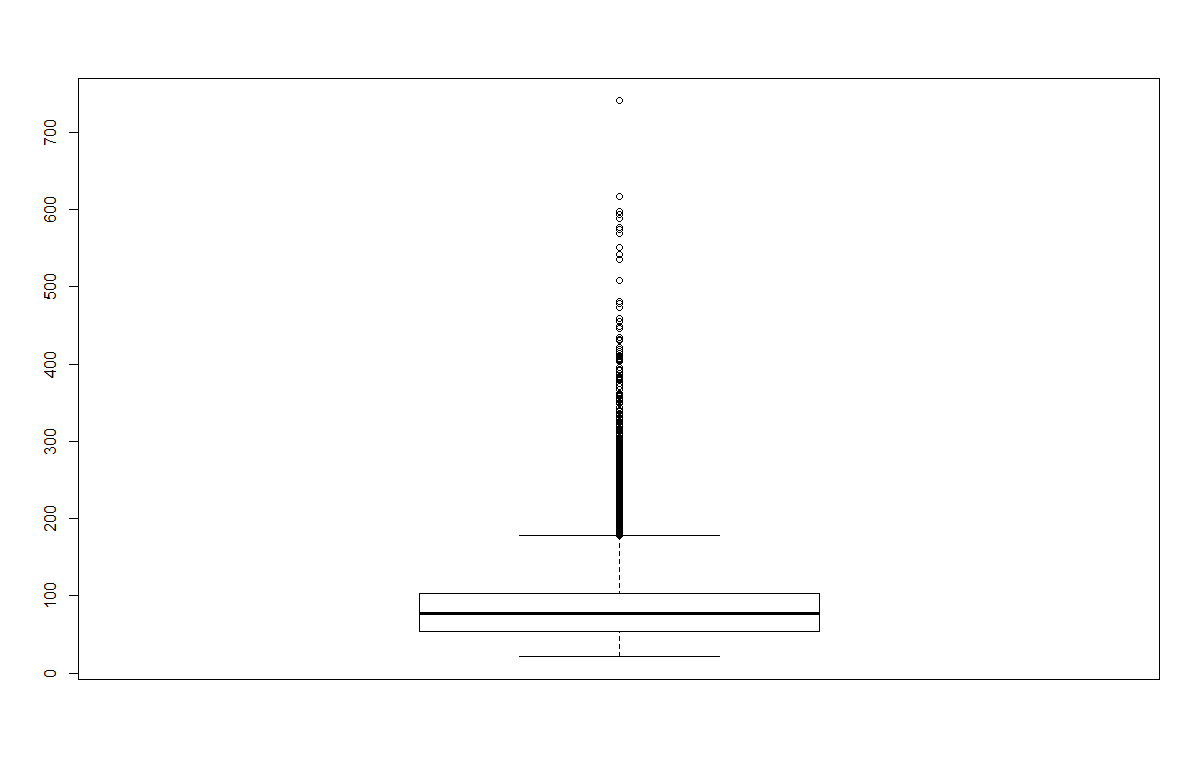
对房源的卧室数进行分析，对卧室数为5个和5个以上的记为5+卧室



对每个区的的房源数进行分析，虽然崇明和金山的房源数偏少，但是为了对每个区的单价情况进行处理，这里就先不做处理。

房源建造时间的处理同上。

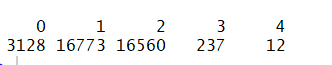
### 3．文本挖掘数据处理



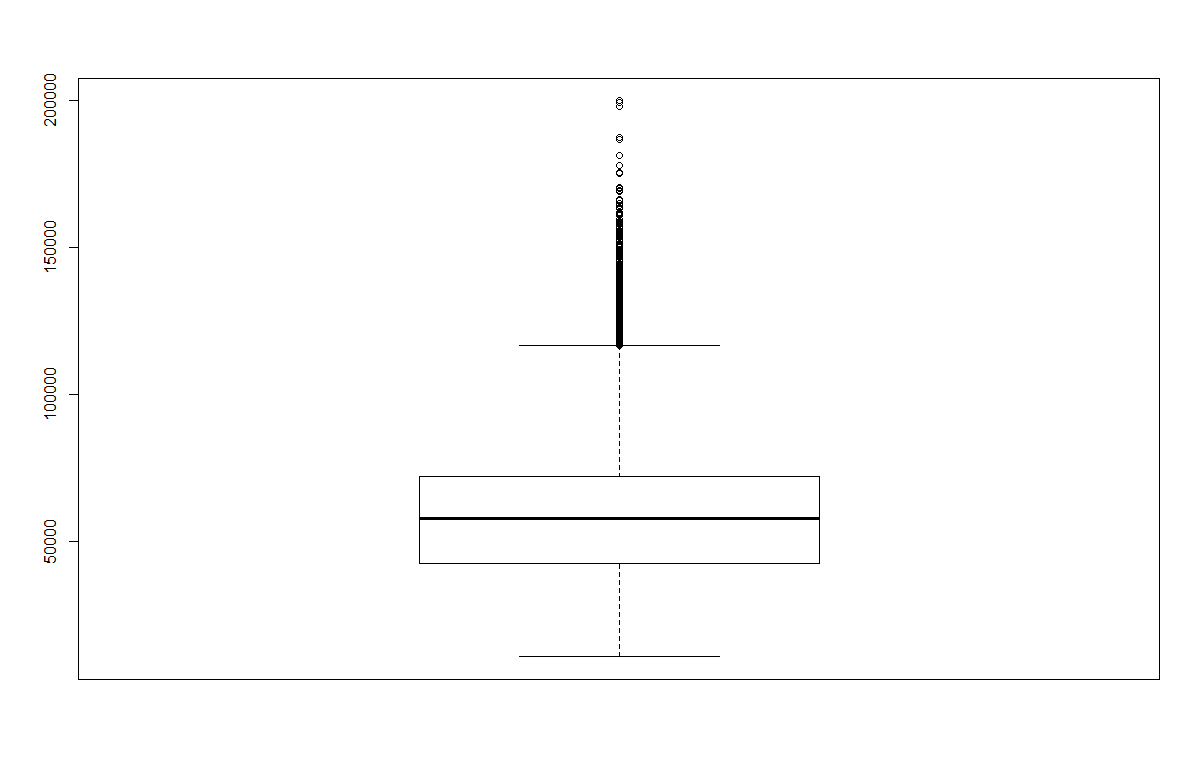
对房源的总面积采用箱线图来表示，可以发现面积这个参数中还是存在许多的较多的离群点，所以首先假设房屋面积符合卡方分布（第三部分的描述性分析中的图会给出这种假设的原因），选取其中的1%（30平方米）和99%（230平方米）的数据。



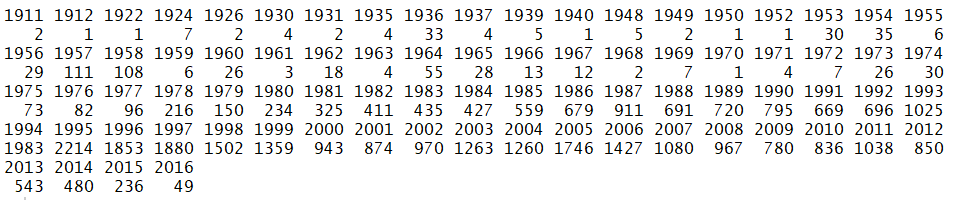
对经过面积筛选的二手房房源进行卧室数的分析，将样本量较少的7个及七个卧室数以上的房源去掉。



对经过面积和卧室筛选后的二手房中进行厅数的分析，将样本中较少的4厅数进行删除。



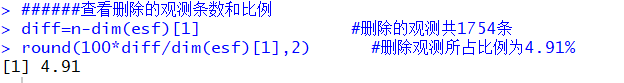
对经过面积、卧室数和厅数筛选后的二手房房源的房屋单价进行分析，从箱线图中我们可以很容易的发现有很多的离群点，所以首先假设二手房的房屋单位面积价格服从正态分布（第三部分的描述性分析中的图会给出这种假设的原因），再根据采用正态分布的3σ原理，选取房屋价格中的1%（20541）和99%（122448）的数据。



对经过面积、卧室数、厅数和单价筛选后二手房房源的建造时间的分析中，可以发现其中有许多年限的二手房的数量较少，存在特殊和偶然性，所以去掉相应年段中数量少于30个房源的数据。

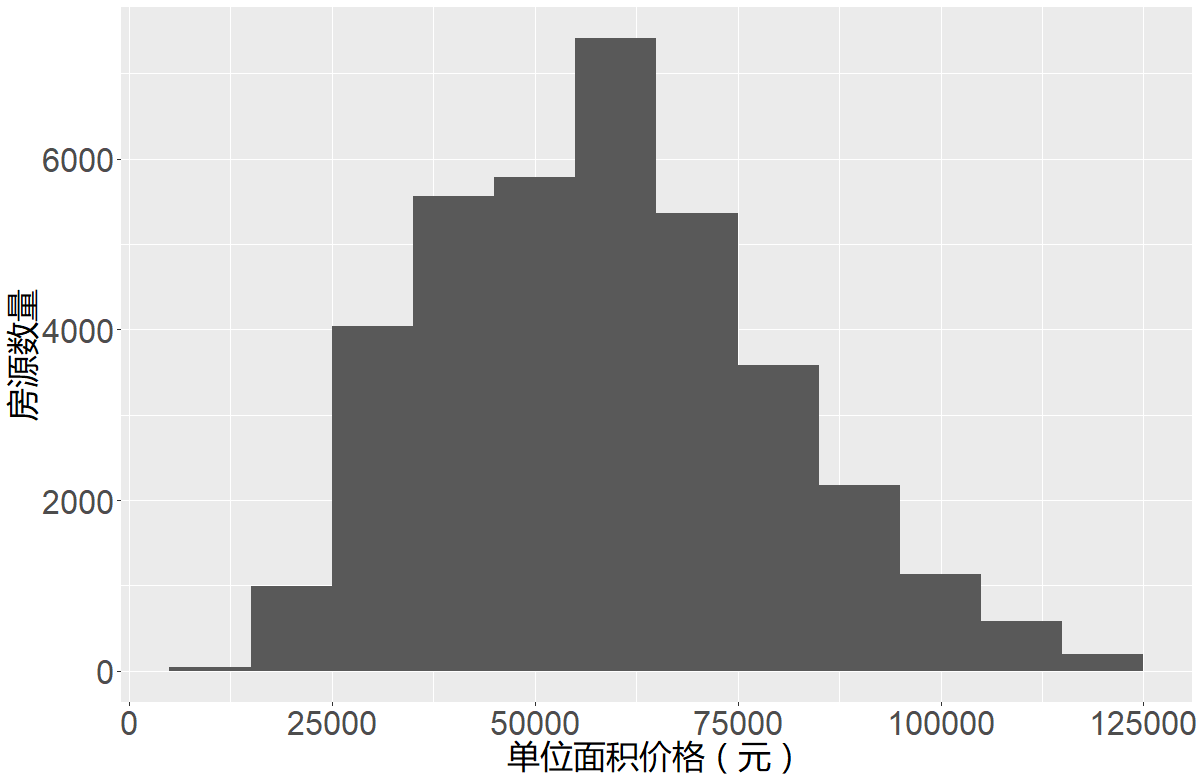


对经过面积、卧室数、厅数、单价和建造时间筛选后的二手房的房源数据进行分析，可以发现其中崇明和金山的房源数据偏少，所以将崇明和金山的数据从数据集中去掉。

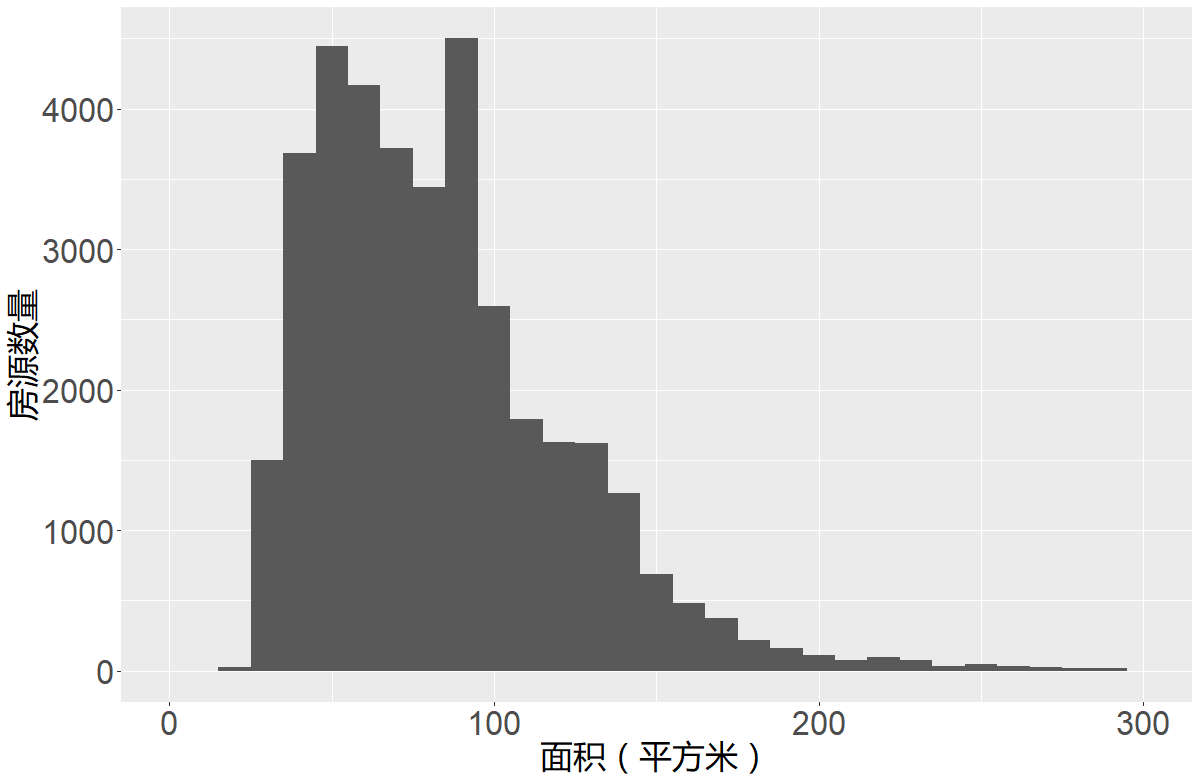


在上海市二手房房源进行文本挖掘时共删除了1754条数据，占总处理后的数据集的百分比为4.91%。

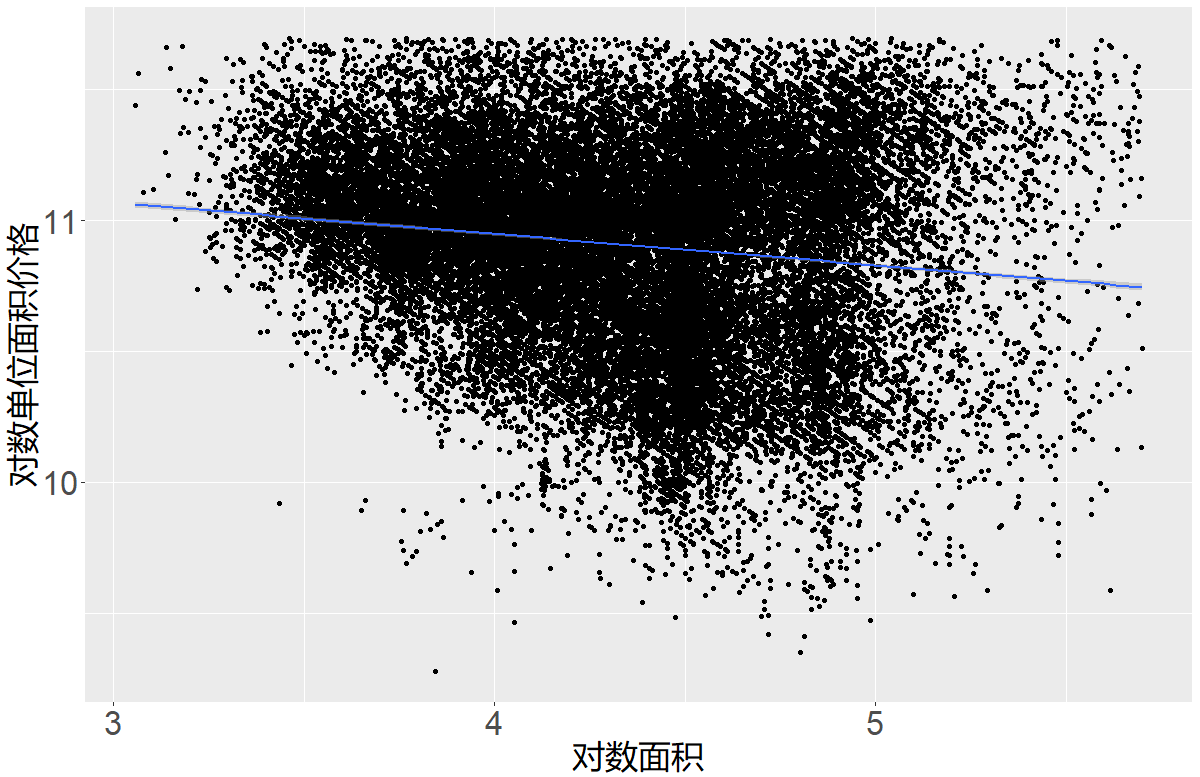
# 三、描述性分析



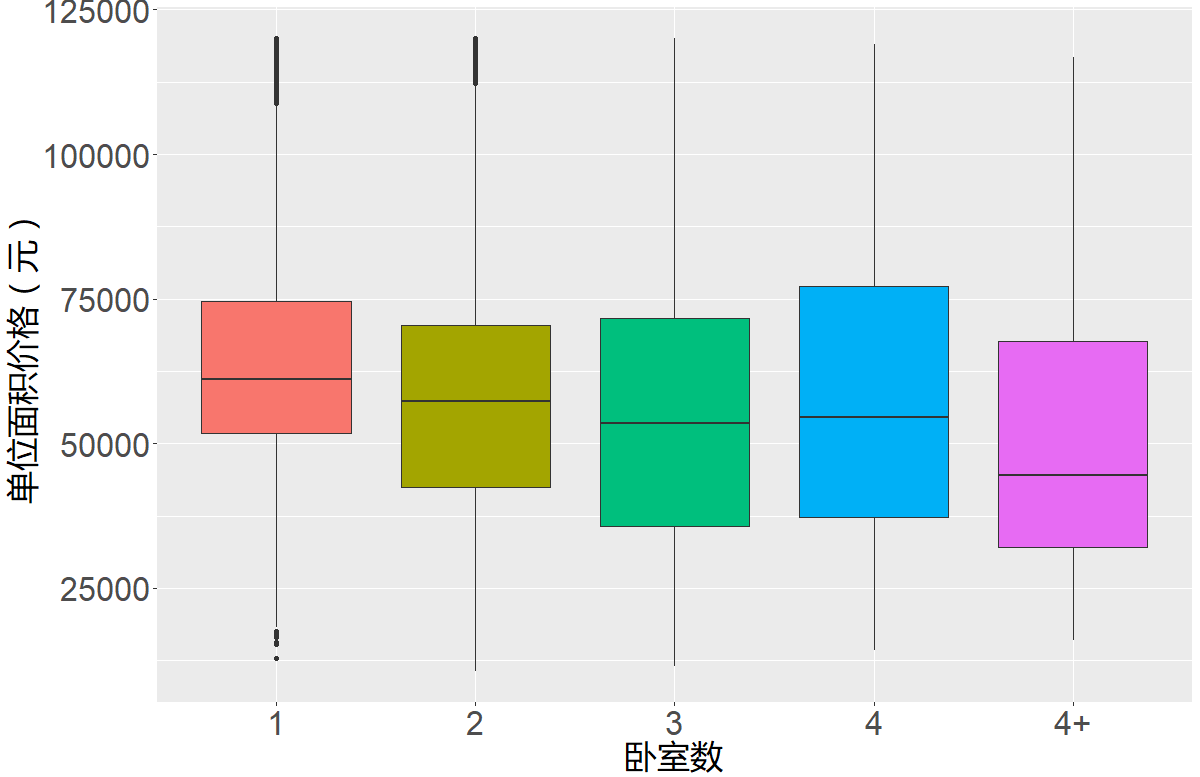
将进行处理后的二手房的房源数据进行柱形图展示可以发现，房源数量随单位面积价格呈现正态分布的样式，表明着上海市二手房的房价大概在62500元/m2。

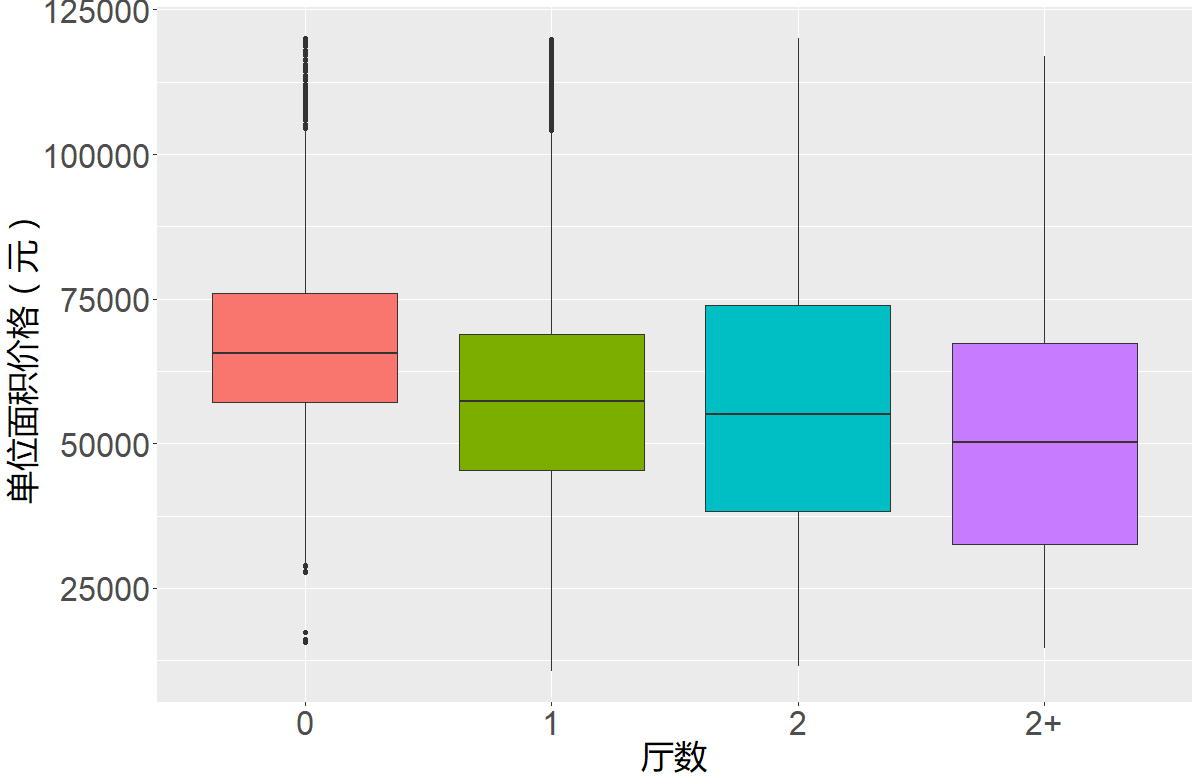


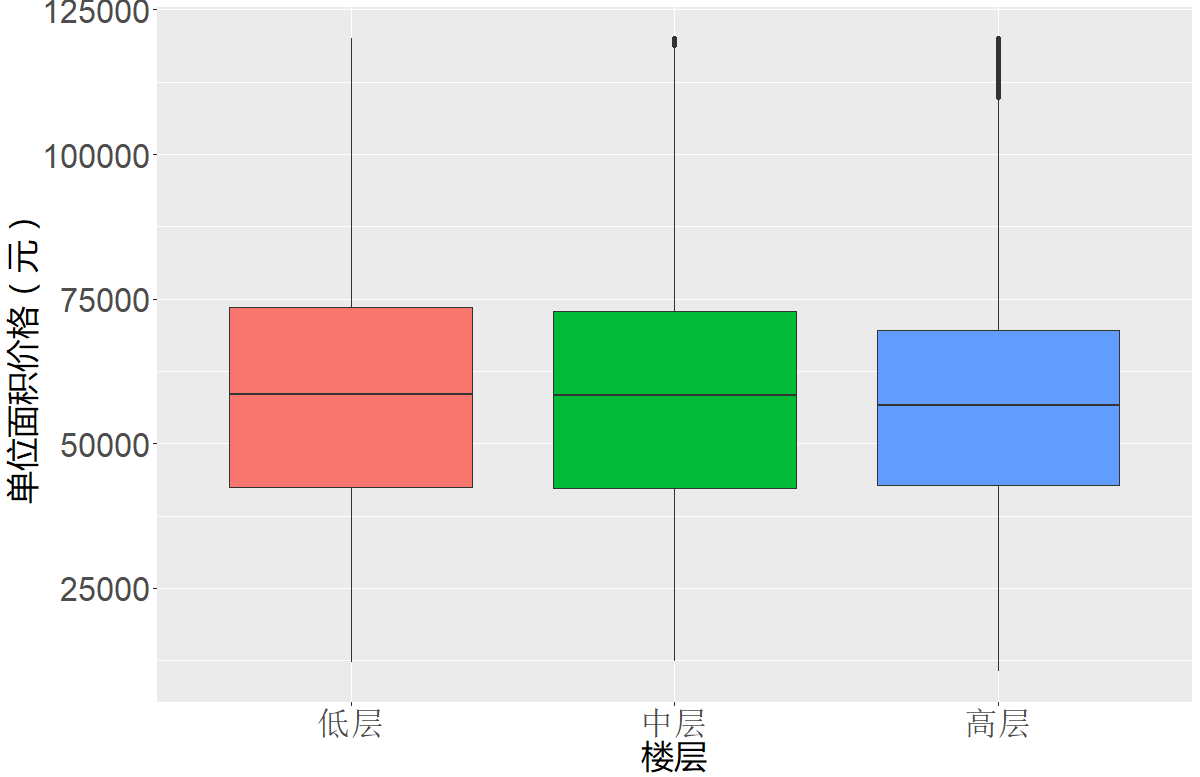
对上海市的二手房房源的面积进行分析的时候，可以发现上海市二手房的房源面积比较符合卡方分布，其面积大概在150平方米以下，尤其是100平方米以下占了大多数，这样的房源面积图示也与上海房价奇高有着不可分割的关系，据此估计上海市二手房的面积均值为75m2。



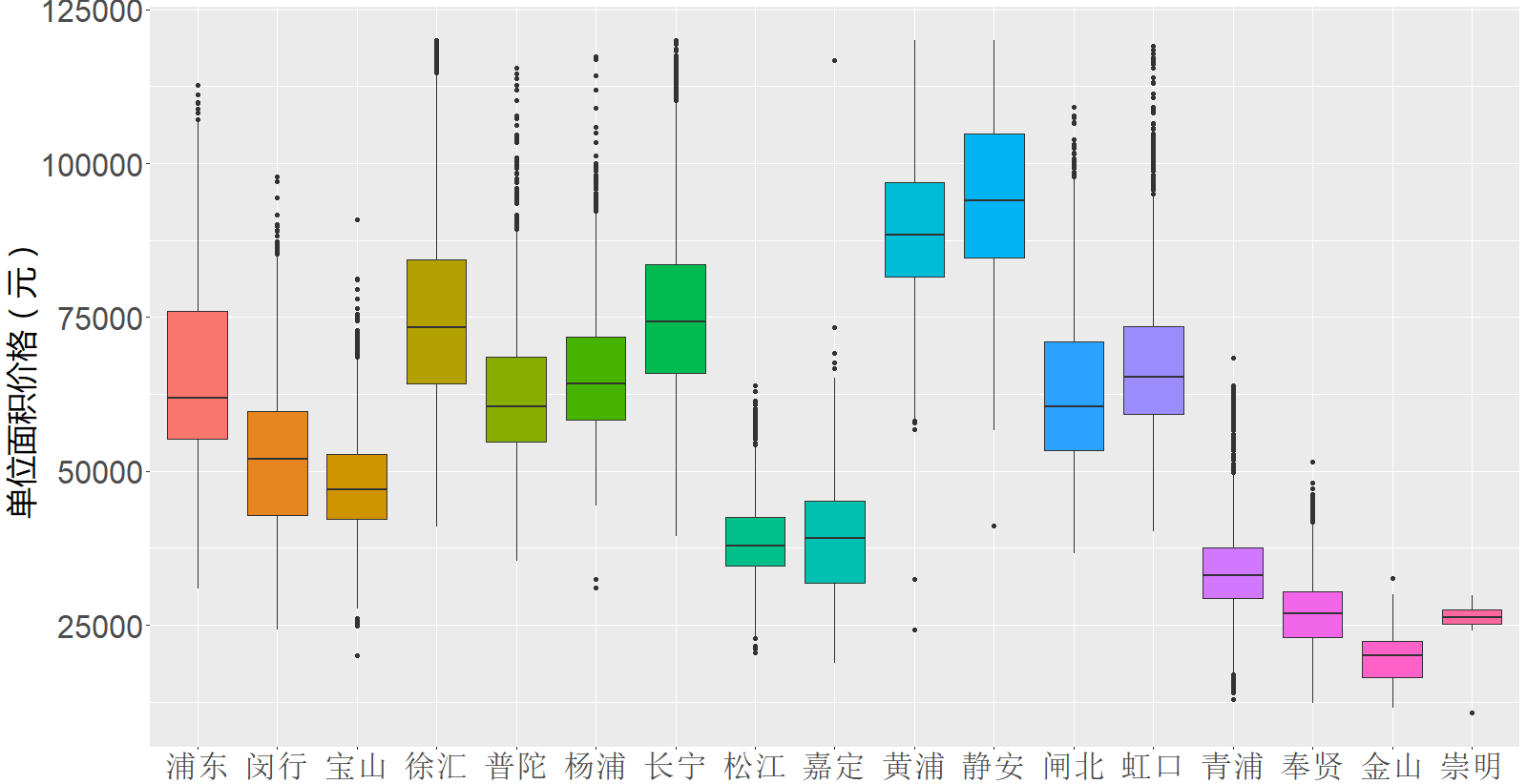
取面积对数和单位面积价格的对数进行作图，可以发现在上海市二手房的房源的单位面积价格是与面积成反比，这与编者最开始认为会是正比的想法有所出入，其产生的原因可能与房源总价有关系，因为太高少有人能负担的起所以才会产生这种反比现象，数据挖掘的科学性和事实性改变我对上海市二手房的走向的看法，可以很好的应用于日后买房，甚至可能出现同样的价格买更大的面积的现象出现。







从上面三个箱线图中可以发现卧室数、厅数以及楼层数都与单位面积价格（元）呈现反比趋势，但是卧室数、厅数对单位面积影响较大，而楼层对单位面影响相对较小。由此可以得出在购买房屋的时候应主要考虑卧室数和厅数，楼层可以少做或者不做考虑，也可以直接选取中间楼层或者高层（低层）的房屋来挑选，不用考虑层级对于房价的影响。

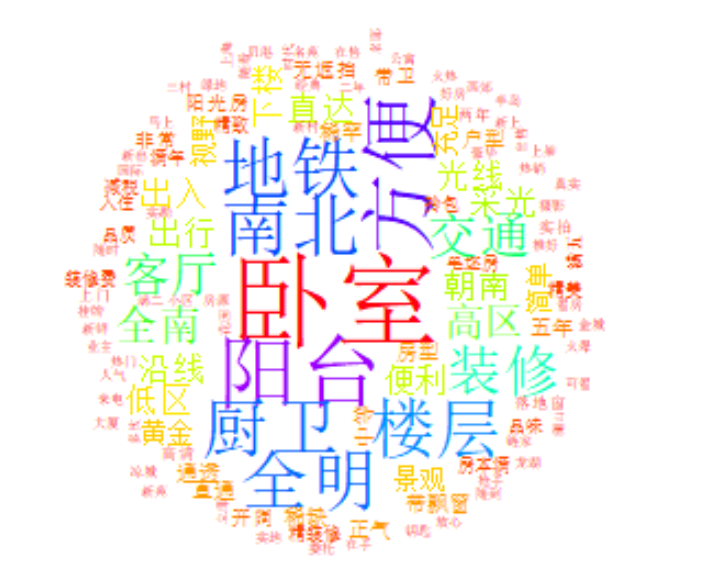


在对整个以处理后的数据集进行箱线图展示的时候，可以看出每个区的单位面积价格走势，其中以静安、黄浦的单位面积价格最贵，金山、崇明的最低。也可以从这个图中大概看出上海市的区经济发展状况和一级、二级市中心的位置，还可以辅助决策上海市该大力扶持哪个市区等。

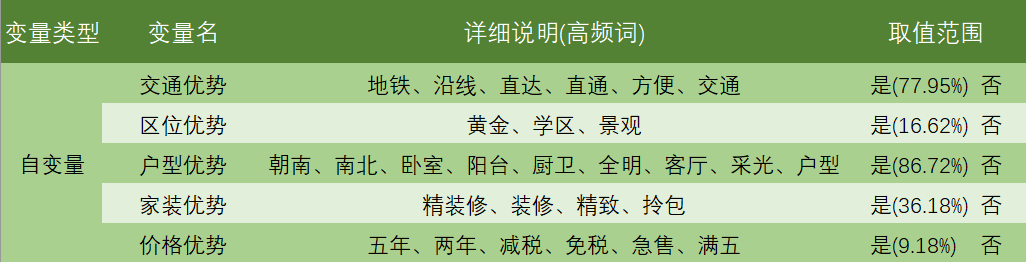
# 四、文本挖掘

在我们处理后的数据集中我们可以很容易的发现链家对在售二手房的广告宣传用语。比如：“卧室带阳台，卧室全南，地铁房，低楼层”，“卧室带阳台，卧室全南，地铁直达，采光好”等标语。这些广告用于可以用于挖掘出二手房市场中市场消费者的需求条件和二手房的优势因素。

首先我们采用我们利用RStudio中的jiebaR程序包对文本进行分词，并利用wordcloud程序包绘制出高频词(120个)的词云。



通过对该词云的高频词进行分析和提炼，建立如下所示的标签字典（相关的高频词与相应的标签对应）。



其中取值范围代表的是此类优势在发布的二手房房源中的比例。比如说交通优势，代表的是在已发布的二手房房源中有77.95%的房源广告中强调了交通优势，剩下的22.05%则没有强调交通优势。通过这些优势变量可以大致得出在上海市二手房市场中占优势的因素即客户的喜好程度，对于售房经理来说，当新挂上的房源有着更多的对于交通、户型和家装的描述的话将更利于二手房被客户所关注和购买，而且卖房用户和买房客户还可以针对交通便利度、户型样式和家庭装修情况等来对房价的现价和走势进行估计。

# 五、分类采用的模型及原因

笔者在进行数据挖掘的时候，所有的训练数据集和测试数据集都是把已有的已预处理过的数据集按照80%作为训练集，20%作为测试集来进行模型的训练的。

## （1）线性回归模型

线性回归模型是比较基本和经典的房价训练模型，在预测波士顿房价的这个数据挖掘案例中使用的就是线性回归模型。并且，给每个房价的特征赋予一个系数来生成具体的房价是一个直观上感觉很正确的方法。

## （2）神经网络

神经网络是目前比较流行的数据挖掘方法，对于数据的训练通常会有比较好的结果，因为通过反向传播等算法不断迭代使得预测结果不断逼近实际值，可以与线性回归模型进行对比来看看是否会有更好的表现和结果。

## （3）支持向量机

支持向量机也目前是比较流行的数据挖掘算法，对于线性模型的训练效果一般会优于线性回归模型，使用这个模型也是想看看能不能针对线性回归模型进行一些提高。

# 六、分类模型使用python的Sklearn库

笔者做分类模型的时候使用的是Python2，因为目前Python2对数据挖掘的支持比较好，有大量的数据挖掘的库可以调用，而且笔者的爬虫也是使用Python2编写的，所以笔者决定使用Python2的Sklearn库中的数据挖掘算法来进行训练及预测。

Sklearn作为目前数据挖掘与机器学习中一个常用的python第三方模块，封装了许多数据挖掘和机器学习的算法，它的官网是<http://scikit-learn.org/stable/>，在github上可以看到相应的介绍和源代码。Sklearn库是在2007年Google的一个暑期项目中由David Cournapeau开发并由800多位contributor合力开发出的机器学习库。

# 七、数据挖掘建模过程

## （1）数据特征分析

总共爬取了37460条房源信息，爬取的二手房房源信息的特征如下：



其大部分的特征都是number类型，根据平时对于房价的影响因素的了解以及描述性信息和文本挖掘中得到的信息，目前认为比较重要的特征有房子大小与户型，房子位置，房子的建造年份这几个特征。

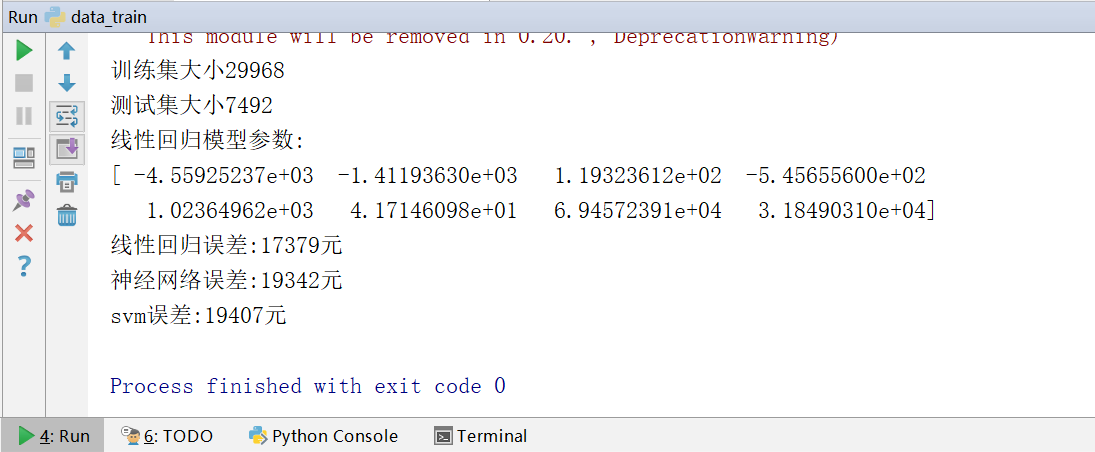
在训练时使用的特征按如下顺序输入：房子中的房间数，房子中的客厅数，房子的大小，房子的建造年份，房子的高度，房子所在那栋楼的高度，房子的经度，房子的纬度。一共八个特征。

## （2）模型调参

在训练的过程中，根据训练出的结果的特征的重要程度不同进行调参，把不重要的特征删除并再次进行训练，直到剩下的特征都对结果有着比较大的影响。

## （3）结果分析及模型对比

对于训练出的结果可以观测到，线性回归模型，神经网络，支持向量机的误差基本都在每平方米17000-20000元之间，这个误差还是比较大的。所以，这个房价的模型很可能是非线性的，才会产生如此大的误差。



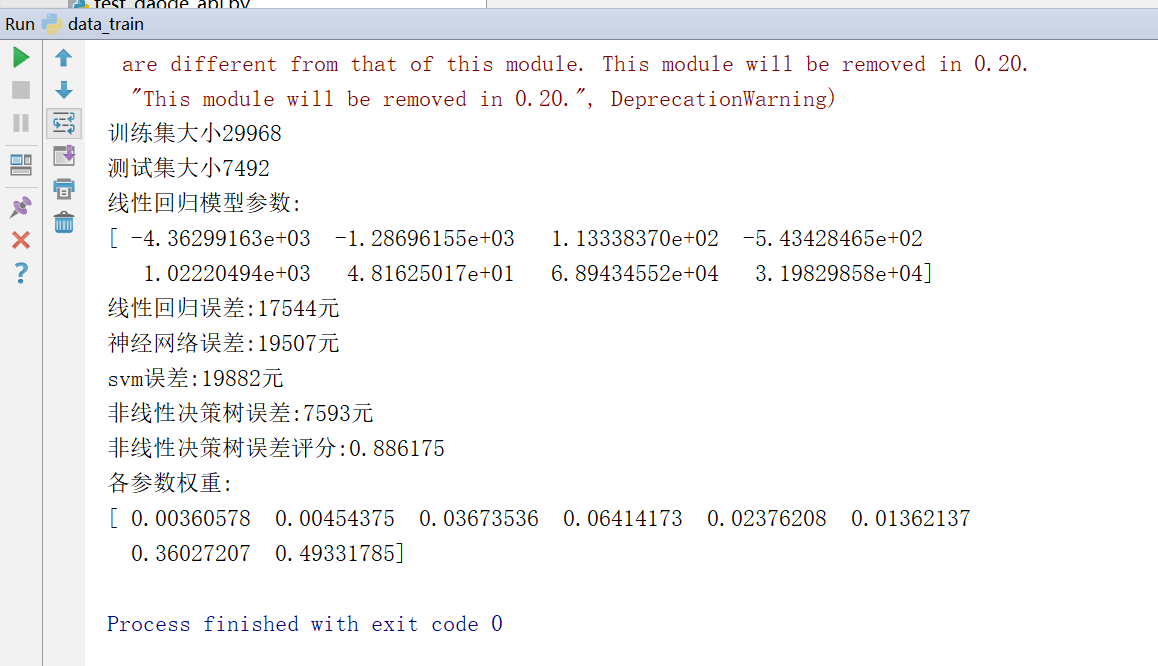
由上图可知，线性回归模型的表现较好，神经网络次之，支持向量机是最差的。另外，从线性回归模型的参数来看，每多一个房间，每多一个客厅，房子建的越晚，都会导致房价变低，这显然是与大众常识不相符合的。所以这个房价模型应该是一个非线性模型。

# 八、非线性模型建模

通过以上的分析和判断大概确定了该模型是一个非线性模型，下面笔者将会采用非线性决策树来进行建模比较。

## （1）非线性决策树

使用了Sklearn库中自带的DecisionTreeRegressor模型，经计算误差在7600元每平左右。评分也达到了0.886（满分1分），因此可认为该模型对于上海市二手房房价模拟的较好。

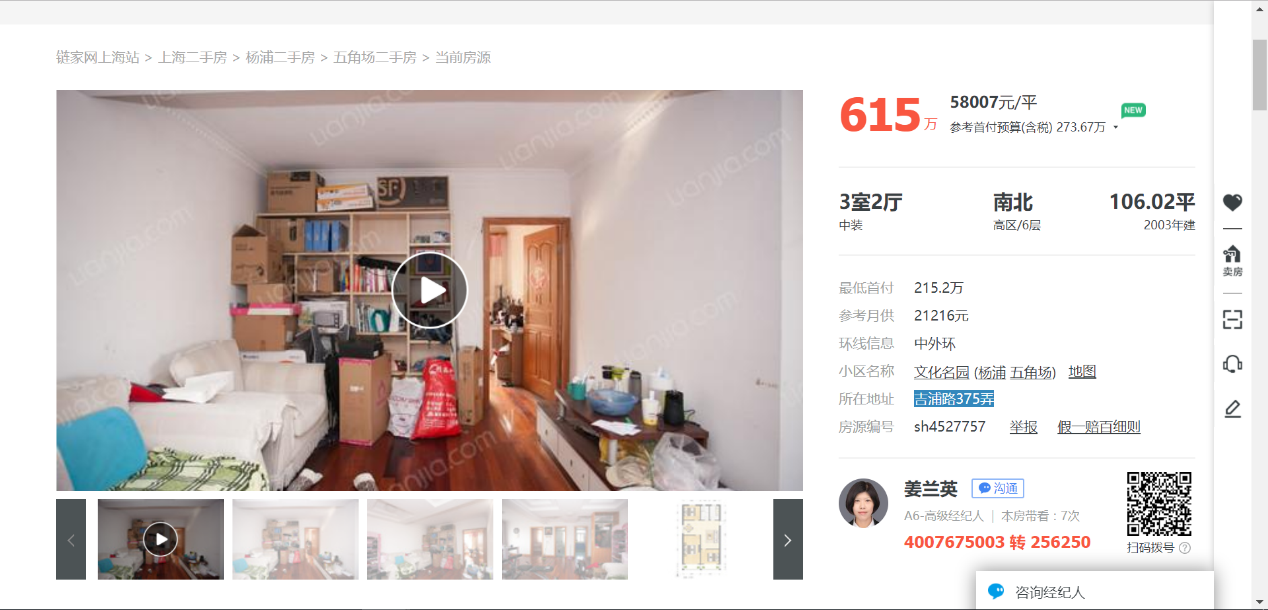


## （2）结论

可以从上图看出，在改用非线性模型之后，模拟的结果有了较大的改善。另外根据输出的模型的参数权重，可以得知，房子所处的位置对房价有较大的影响，这与第三部分所展现的每个区的单位面积价格的箱线图的情况基本符合，也与上海每个区、每个地方的单位房价差距较大的事实基本吻合。

# 九、房价查询界面

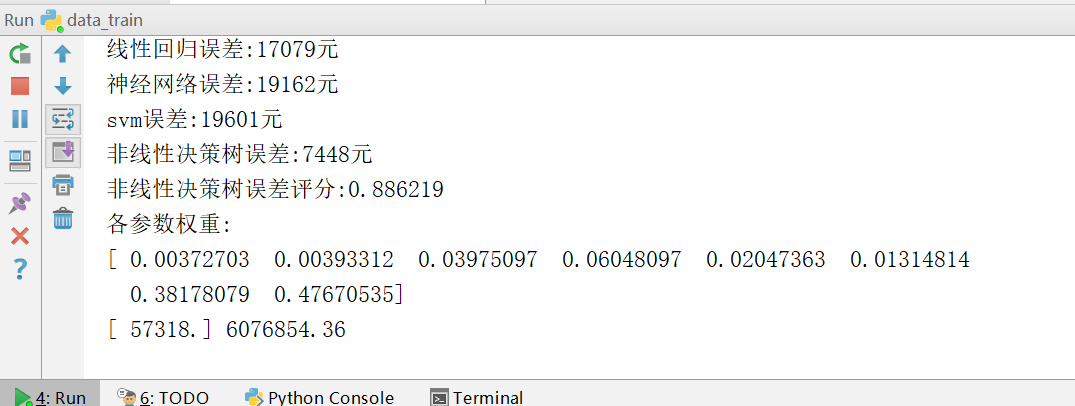
先从链家网上随意找到一条房源信息



输入查询界面



点击查询后进行房价的查询



Pycharm的控制台中会输出房子的每平米价格及总价，可以看到，链家网上所挂出的价格是每平米58007元，模型的预测结果为每平米57318元，总价相差7.31万，基本达到了帮助用户了解房价的需求。

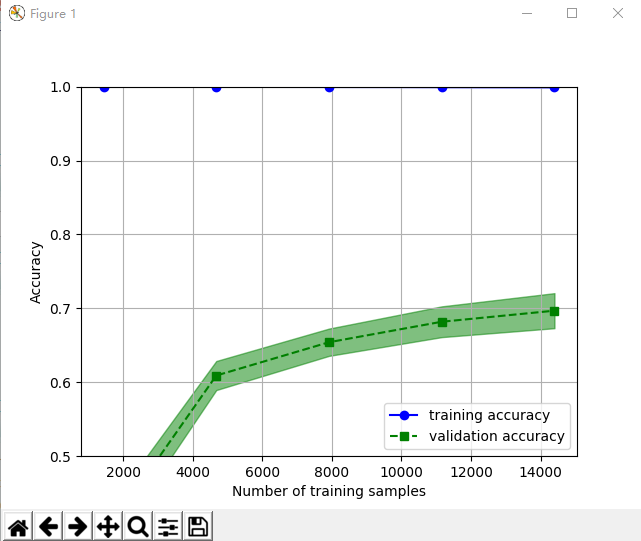
# 十、缺陷和改进措施

## 10.1缺陷：

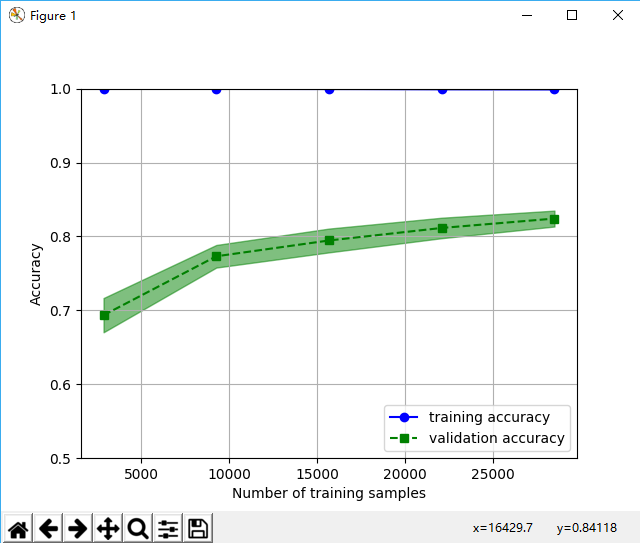
虽然采用非线性决策树模型模拟房价的效果较好，但还是有较大的缺陷，该模型对于房价不超过700万的房子模拟较好，误差一般在15万以内，当对于700万以上尤其是850万以上的房子模拟较差，对于850万以上的部分房子误差高达100万以上，得到的结果不是特别理想。

## 10.2改进措施

### （1）收集更多数据



数据集在20000条以下时的训练学习曲线



数据集在40000条以下时的训练学习曲线

从上面的两张图可以看出，随着数据集变大，验证集的准确率不断上升，所以需要收集更多的训练数据来精化模型。目前的房价数据还不是非常的多，容易受到部分异常房价信息的影响

### （2）寻找更多特征

影响房价的因素还有很多，比如房子周围是否交通便利，房子周围是否有大型商业圈，房子的装修好差以及产权和车位情况等，都会对房价有所影响。如果可以在房子的模型信息中加入这些特征，那么也可以提升模型精度来解决850万以上的房价估值不准确的问题。

### （3）分析方法的结合

虽然本文笔者采用了分类模型和文本挖掘两大技术来对上海市二手房房产市场进行了分析，但其实两者的结合使用很少，比如说文本挖掘中得到的二手房的“交通优势”很少在分类模型中被应用出来；文本挖掘中也发现的“家装优势”也在进行分类模型训练时被笔者给忽略了……这些文本挖掘中得到的结果如果被用于分类模型训练中的话，模拟得到的结果应该会更好，850万以上的房价估值不准确的问题应该会得到很大程度的改善。

# 十一、附录

### 代码已上传至码云：<https://gitee.com/LvXiaoXiang/intellectual>

### 爬虫代码

*# coding:utf-8 ctrl+/ 注释***import** urllib2  
**import** time  
**import** bs4  
**from** bs4 **import** BeautifulSoup  
**import** sys  
**import** pymongo  
**import** requests  
  
  
**def** geocode(address):  
 parameters = {**'address'**: address, **'key'**: **'8ac4f59c23c73f503f350494ff9310d3'**, **'city'**: **'上海'**}  
 base = **'http://restapi.amap.com/v3/geocode/geo'  
 try**:  
 response = requests.get(base, parameters, timeout=1)  
 **except**:  
 **return** {}  
 *# print response.elapsed.microseconds* answer = response.json()  
 **return** answer  
  
  
reload(sys)  
sys.setdefaultencoding(**"utf-8"**)  
  
connection = pymongo.MongoClient()  
tdb = connection.program  
post\_info = tdb.testhouse  
  
  
*# 链家网d***def** find\_data(tmp\_url, tmp\_district, lists):  
 count = 0  
 *# 每个区的最大显示页数为100页* **for** page\_Num **in** range(1, 100):  
 f\_url = tmp\_url + tmp\_district + **"/d"** + str(page\_Num)  
 **print "第"**+str(page\_Num)+**"页"  
 print** f\_url  
  
  
  
 *# print f\_url* f\_page = urllib2.urlopen(f\_url)  
 f\_soup = BeautifulSoup(f\_page, **"html.parser"**)  
 page\_soup = f\_soup.find(class\_=**"m-list"**)  
 *# print page\_soup* ul\_soup = page\_soup.find(**'ul'**)  
 *# print ul\_soup* li\_list = ul\_soup.findAll(**'li'**)[0:]  
  
 houseNum=0.0  
 *# print page\_Num* **for** tr **in** li\_list:  
 houseNum = houseNum + 1  
 **print "第"**+str(page\_Num)+**"页"**+**"已完成"** + str(float(houseNum/len(li\_list)\*100))[0:4] + **"%,共100页"** getTitle = tr.find(class\_=**"prop-title"**).text  
 introduce=getTitle.strip()  
  
 info = tr.findAll(class\_=**"info-row"**)[0:]  
 row1 = info[0].find(class\_=**"info-col row1-text"**).text  
  
 row1 = row1.strip()  
 row1 = row1.replace(**' '**, **''**)  
 *# print row1* cut\_1 = row1.index(**'|'**)  
 *# 几室几厅* house\_type = row1[0:cut\_1]  
 *#print house\_type* cut\_2 = row1[cut\_1 + 1:].index(**'平'**)  
 *# 房屋大小* house\_size = float(row1[cut\_1 + 1:cut\_1 + cut\_2 + 1])  
 *#print house\_size* **try**:  
 cut\_3 = row1.index(**'/'**)  
 **except** ValueError:  
 **continue** cut\_4 = row1.index(**'层'**)  
 *# 建筑总层高* building\_height = float(row1[cut\_3 + 1:cut\_4])  
 *#print building\_height* cut\_5 = row1.index(**'区'**)  
 *# 房屋层高* house\_height = row1[cut\_5 - 1:cut\_5]  
 *#print house\_height* row2 = info[1].find(class\_=**"info-col row2-text"**)  
 *# 均价* average\_price = info[1].find(class\_=**"info-col price-item minor"**).text.strip()  
 *# print average\_price  
 # 位置* location = row2.findAll(**'a'**)[0:]  
  
 **try**:  
 year\_1 = row2.text.index(**'年建'**)  
 year = row2.text[year\_1 - 4:year\_1]  
 **except** ValueError:  
 **continue** *# print year  
 # 小区* housing\_estate = location[0].text  
 *#print housing\_estate  
 # 区县* house\_district = location[1].text  
 *#print house\_district* count = count + 1  
 *# print count  
 #  
 #  
 #整理出room和parlour数量* room\_1=house\_type.index(**'室'**)  
 room\_number=int(house\_type[0:room\_1])  
 parlour\_1=house\_type.index(**'厅'**)  
 parlour\_number=int(house\_type[room\_1+1:parlour\_1])  
  
 *#判断房子的具体高度* **if**(house\_height==**'中'**):  
 house\_height\_inlist=building\_height\*0.5  
 **elif**(house\_height==**'高'**):  
 house\_height\_inlist = building\_height \* 0.88  
 **elif**(house\_height == **'低'**):  
 house\_height\_inlist = building\_height \* 0.23  
  
 *#整理出具体房价* price\_1=average\_price.index(**'价'**)  
 price\_2=average\_price.index(**'元'**)  
 average\_price\_inlist=float(average\_price[price\_1+1:price\_2])  
  
 *#计算地址经纬度* address = house\_district+housing\_estate  
 *# print address* house\_first\_location=geocode(address)  
  
 **if "geocodes" not in** house\_first\_location.keys():  
 *# print "no geocodes"* **continue** house\_first\_location=house\_first\_location[**"geocodes"**]  
  
 *#print house\_first\_location* **if**(house\_first\_location==[]):  
 **continue** house\_location=house\_first\_location[0][**'location'**].encode(**'utf-8'**)  
 *# print house\_location* house\_location\_1=house\_location.index(**','**)  
 house\_location\_longtitude=float(house\_location[0:house\_location\_1])  
 *# print house\_location\_longtitude* house\_location\_latitude=float(house\_location[house\_location\_1+1:])  
 *# print house\_location\_latitude  
 #house\_location\_latitude,house\_location\_longtitude,* list\_use = [room\_number,parlour\_number, house\_size, building\_height, house\_height\_inlist,float(year),house\_location\_longtitude,house\_location\_latitude,house\_district,address,introduce,  
 average\_price\_inlist]  
  
 *#print list\_use* lists.extend([list\_use])  
  
 *# print "{\"户型\":\"%s\", \"大小\":\"%s\",\"楼高\":\"%s\",\"层高\":\"%s\",\"小区名\":\"%s\",\"市区\":\"%s\",\"均价\":\"%s\"}" % (  
 # house\_type, house\_size, building\_height, house\_height, housing\_estate, house\_district, average\_price)* data = {**"room\_number"**: room\_number, **"parlour\_number"**: parlour\_number,**"house\_size"**:house\_size,**"building\_height"**: building\_height,**"house\_height\_inlist"**: house\_height\_inlist,**"year"**: float(year),**"house\_location\_longtitude"**: house\_location\_longtitude,**"house\_location\_latitude"**:house\_location\_latitude,**"house\_district"**:house\_district,**"address"**:address,**"average\_price\_inlist"**: average\_price\_inlist,**"introduce"**:introduce}  
 post\_info.save(data)  
 *# print "end"  
# print "all\_end"***def** use(district):  
 lists = []  
 j=0  
 **for** i **in** district:  
 j=j+1  
 **print "第"**+str(j)+**"个区:"**+i  
 find\_data(**'http://sh.lianjia.com/ershoufang/'**, i, lists)  
 **print**(**"全部完成"**)  
 **return** lists  
  
  
b = [**"jinshan"**, **"chongming"**]  
a = [**"pudongxinqu"**, **"minhang"**, **"baoshan"**, **"xuhui"**, **"putuo"**, **"yangpu"**, **"changning"**, **"songjiang"**, **"jiading"**, **"huangpu"**,  
 **"jingan"**, **"zhabei"**, **"hongkou"**, **"qingpu"**, **"fengxian"**, **"jinshan"**, **"chongming"**]  
dataset = use(b)

### 分类模型和Python的Gui图形界面代码

*# coding:utf-8 ctrl+/ 注释***import** sys  
**import** pymongo  
**import** numpy **as** np  
**import** test\_gaode\_api  
  
reload(sys)  
sys.setdefaultencoding(**"utf-8"**)  
  
connection = pymongo.MongoClient()  
tdb = connection.program  
post\_info = tdb.testhouse  
  
lists=[]  
count=0  
  
**for** item **in** post\_info.find():  
 count+=1  
  
 single\_item=[item[**"room\_number"**],item[**"parlour\_number"**],item[**"house\_size"**],item[**"year"**],  
 item[**"building\_height"**],item[**"house\_height\_inlist"**],item[**"house\_location\_longtitude"**],  
 item[**"house\_location\_latitude"**],item[**"average\_price\_inlist"**]]  
 lists.extend([single\_item])  
 *#print count***from** sklearn **import** metrics  
**from** sklearn **import** preprocessing  
**from** sklearn.linear\_model **import** LinearRegression  
**from** sklearn.cross\_validation **import** train\_test\_split  
  
\_array = np.array(lists)  
x = \_array[:, 0:8]  
*#normalized\_X=preprocessing.normalize(x)*y = \_array[:, 8]  
*#normalized\_Y=preprocessing.normalize(y)*X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y,test\_size=0.2)  
  
**print "训练集大小%d"**%(X\_train.shape[0])  
**print "测试集大小%d"**%(X\_test.shape[0])  
  
*#线性回归*clf = LinearRegression()  
clf.fit(X\_train, y\_train)  
**print "线性回归模型参数:"  
print** clf.coef\_  
y\_pred = clf.predict(X\_test)  
  
**print** (**"线性回归误差:%d元"**%(np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred))))  
  
  
*#神经网络***from** sklearn.neural\_network **import** MLPRegressor  
kmodel = MLPRegressor(learning\_rate=**'adaptive'**,max\_iter=2000).fit(X\_train, y\_train)  
y\_kmodel\_pred = kmodel.predict(X\_test)  
**print** (**"神经网络误差:%d元"**%np.sqrt(metrics.mean\_squared\_error(y\_test, y\_kmodel\_pred)))  
  
*#svm模型***from** sklearn **import** svm  
smodel=svm.LinearSVR()  
smodel.fit(X\_train, y\_train)  
y\_smodel\_pred=smodel.predict(X\_test)  
  
**print "svm误差:%d元"**%np.sqrt(metrics.mean\_squared\_error(y\_test, y\_smodel\_pred))  
  
  
*#决策树模型***from** sklearn.tree **import** DecisionTreeRegressor  
dmodel=DecisionTreeRegressor()  
dmodel.fit(X\_train, y\_train)  
  
y\_dmodel\_pred=dmodel.predict(X\_test)  
**print "非线性决策树误差:%d元"**%np.sqrt(metrics.mean\_squared\_error(y\_test, y\_dmodel\_pred))  
**print "非线性决策树误差评分:%f"**%dmodel.score(X\_test,y\_test)  
  
**print "各参数权重:"  
print** dmodel.feature\_importances\_  
*# coding:utf-8 ctrl+/ 注释***import** wx  
  
app=wx.App()  
win =wx.Frame(None,title=**"房价查询"**,size=(420,335))  
win.Show()  
  
btn=wx.Button(win,label=**'查询'**,pos=(315,5),size=(80,25))  
  
  
house\_size\_input=wx.TextCtrl(win,pos=(80,5),size=(210,25))  
house\_size\_input\_label=wx.StaticText(win,label=**'房屋大小'**,pos=(10,10),size=(60,30))  
  
room\_number\_input=wx.TextCtrl(win,pos=(80,45),size=(210,25))  
room\_number\_input\_label=wx.StaticText(win,label=**'房间数'**,pos=(10,50),size=(60,30))  
  
parlour\_number\_input=wx.TextCtrl(win,pos=(80,85),size=(210,25))  
parlour\_number\_input\_label=wx.StaticText(win,label=**'厅数'**,pos=(10,90),size=(60,30))  
  
house\_height\_input=wx.TextCtrl(win,pos=(80,125),size=(210,25))  
house\_height\_input\_label=wx.StaticText(win,label=**'层高'**,pos=(10,130),size=(60,30))  
  
building\_height\_input=wx.TextCtrl(win,pos=(80,165),size=(210,25))  
building\_height\_input\_label=wx.StaticText(win,label=**'楼高'**,pos=(10,170),size=(60,30))  
  
year\_input=wx.TextCtrl(win,pos=(80,205),size=(210,25))  
year\_input\_label=wx.StaticText(win,label=**'建造年份'**,pos=(10,210),size=(60,30))  
  
address\_input=wx.TextCtrl(win,pos=(80,245),size=(210,25))  
address\_input\_label=wx.StaticText(win,label=**'地址'**,pos=(10,250),size=(60,30))  
  
  
**def** search(event):  
 hz = house\_size\_input.Value  
 rn=room\_number\_input.Value  
 pn=parlour\_number\_input.Value  
 hh=house\_height\_input.Value  
 bh=building\_height\_input.Value  
 yi=year\_input.Value  
 ai=address\_input.Value  
 location=test\_gaode\_api.geocode(ai)[**"geocodes"**][0][**'location'**].encode(**'utf-8'**)  
 house\_location\_1=location.index(**','**)  
 house\_location\_longtitude=float(location[0:house\_location\_1])  
 house\_location\_latitude=float(location[house\_location\_1+1:])  
 list=[]  
 list.extend([[hz,rn,pn,hh,bh,yi,house\_location\_longtitude,house\_location\_latitude]])  
 out\_price=dmodel.predict(list)  
 **print** out\_price,float(out\_price[0])\*float(hz)  
 *#print hz,rn,pn,hh,bh,yi,house\_location\_longtitude,house\_location\_latitude*btn.Bind(wx.EVT\_BUTTON,search)  
  
app.MainLoop()

### 上海市二手房描述性分析（R）

rm(list = ls())

house<-read.csv("C:/Users/吕港/Desktop/大三下/人工智能/大作业/房源数据分析/大作业最终文件/secondHouse.csv")

#观察端点值

house[house$average\_price\_inlist==max(house$average\_price\_inlist),]

house[house$average\_price\_inlist==min(house$average\_price\_inlist),]

house[house$total\_price==max(house$total\_price),]

house[house$total\_price==min(house$total\_price),]

house[house$house\_size==max(house$house\_size),]

house[house$house\_size==min(house$house\_size),]

median(house$average\_price\_inlist)

#整理数据集，卧室数厅数只保留数字，楼层只保留四个类别（低层/中层/高层），删去序号列

house$room\_number<-substr(house$room\_number,1,1)

house$parlour\_number<-substr(house$parlour\_number,1,1)

house<-house[house$house\_size<300,c(1:14)] #删去了面积超过300m2的豪宅

house<-house[house$average\_price\_inlist<120000,c(4,13,3,7,9,14,5,12)] #将待用的变量重新组成新的数据框，并且删去了单价超过12万的豪宅

names(house)[1:8]<-c("单价","总价","面积","卧室数","厅数","楼层","建造时间","城区")

#观察各组的情况

library(dplyr)

summarise(group\_by(house,楼层),n=n(),price=mean(单价),area=mean(面积)) #分组还算平均

summarise(group\_by(house,厅数),n=n(),price=mean(单价),area=mean(面积)) #3厅以上不超过400套，于是将3及3以上合并为2+组

summarise(group\_by(house,卧室数),n=n(),price=mean(单价),area=mean(面积))#将5/6/7/8/9合并为4+组

summarise(group\_by(house,城区),n=n(),price=mean(单价),area=mean(面积))

summarise(group\_by(house,建造时间),n=n(),price=mean(单价),area=mean(面积))

#变量分组调整

house$楼层<-factor(house$楼层,order=TRUE,levels=c("低层","中层","高层"))

house$厅数[house$厅数==3|house$厅数==4|house$厅数==6|house$厅数==7|house$厅数==8|house$厅数==9]<-"2+"

house$厅数<-factor(house$厅数,order=TRUE,levels=c("0","1","2","2+"))

house$卧室数[house$卧室数==5|house$卧室数==6|house$卧室数==7|house$卧室数==8|house$卧室数==9]<-"4+"

house$卧室数<-factor(house$卧室数,order=TRUE,levels=c("1","2","3","4","4+"))

house$城区<-factor(house$城区,order=TRUE,levels=c("浦东","闵行","宝山","徐汇","普陀","杨浦","长宁","松江","嘉定","黄浦","静安","闸北","虹口","青浦","奉贤","金山","崇明"))

#设置通用图像样式

library(ggplot2)

windowsFonts(myFont = windowsFont("微软雅黑"))

my.theme<-theme(

axis.text.x=element\_text(size=25),

axis.title.x=element\_text(size=25,family="myFont"),

axis.text.y=element\_text(size=25),

axis.title.y=element\_text(size=25,family="myFont"),

legend.position="none"

)

#观察因变量

price<-ggplot(house,aes(单价))

price+geom\_histogram(binwidth=10000)+xlab("单位面积价格（元）")+ylab("房源数量")+my.theme

#观察自变量-

area<-ggplot(house,aes(面积))

area+geom\_histogram(binwidth=10)+xlim(c(0,300))+xlab("面积（平方米）")+ylab("房源数量")+my.theme

#因变量-自变量关系

price.area<-ggplot(house,aes(log(面积),log(单价)))

price.area+geom\_point()+xlab("对数面积")+ylab("对数单位面积价格")+my.theme+geom\_smooth(method="lm")

price.bedroom<-ggplot(house,aes(卧室数,单价))

price.bedroom+geom\_boxplot(aes(fill=卧室数))+my.theme+ylab("单位面积价格（元）")

last\_plot()+scale\_x\_discrete(labels=c("1","2","3","4","4+"))

price.diner<-ggplot(house,aes(厅数,单价))

price.diner+geom\_boxplot(aes(fill=厅数))+my.theme+ylab("单位面积价格（元）")

last\_plot()+scale\_x\_discrete(labels=c("0","1","2","2+"))

price.floor<-ggplot(house,aes(楼层,单价))

price.floor+geom\_boxplot(aes(fill=楼层))+my.theme+ylab("单位面积价格（元）")

last\_plot()+scale\_x\_discrete(labels=c("低层","中层","高层"))

price.district<-ggplot(house,aes(城区,单价))

price.district+geom\_boxplot(aes(fill=城区))+my.theme+ylab("单位面积价格（元）")+xlab(NULL)

last\_plot()+scale\_x\_discrete(labels=c("浦东","闵行","宝山","徐汇","普陀","杨浦","长宁","松江","嘉定","黄浦","静安","闸北","虹口","青浦","奉贤","金山","崇明"))

### 上海市二手房文本挖掘数据清洗（R）

rm(list = ls())

setwd("C:/Users/吕港/Desktop/大三下/人工智能/大作业/房源数据分析/大作业最终文件") #设置工作路径

######读入数据

#如果你用的是Windows系统，请运行以下代码

esf <- read.csv("secondHouse.csv",header = T) #导入数据

names(esf)[1:15]<-c("序号","具体楼层","总面积.平方米.","单价.元.m2.","建造时间","地理位置经度","卧室数","地理位置经度","厅数","楼层高度","详细地址","城区","总价.元.","楼层","介绍")

#View(esf)

n=dim(esf)[1] #输出样本量。

######通过简单的描述性分析，清洗数据，逐个变量进行处理

#描述分析：房屋介绍

table(esf$介绍) #查看房屋介绍分布，文本型变量，且比较均匀，不做处理

#描述分析：总面积

boxplot(esf$总面积.平方米.) #查看总面积的分布,箱线图显示存在较多离群点

quantile(esf$总面积.平方米.,c(0.01,0.99)) #查看总面积1%和99%分位数

esf=esf[esf[,3]>30&esf[,3]<230,] #根据3σ原理中的1%分位数和99%分位数,将总面积小于30平方米和大于230平方米的观测剔除

#将卧室数和厅数转换为数值型变量

esf$卧室数<-as.numeric(esf$卧室数)

esf$厅数<-as.numeric(esf$厅数)

#描述分析：卧室数和厅数

table(esf$卧室数) #查看频数分布：卧室数为7以上的的观测仅有8个，考虑删除

esf=esf[esf[,7]<7,] #对卧室数变量做部分删除处理

table(esf$厅数) #查看频数分布：厅数为4以上的的观测仅有12个，考虑删除

esf=esf[esf[,9]<4,] #对厅数变量做部分删除处理

#描述分析：房屋单价

boxplot(esf$单价.元.m2.) #查看房屋单价的分布,箱线图显示存在离群点

quantile(esf$单价.元.m2.,c(0.01,0.99)) #查看房屋单价1%和99%分位数

esf=esf[esf[,4]>20541&esf[,4]<122448,] #根据3σ原理中的1%分位数和99%分位数,将房屋单价低于20541元/平方米和高于122448元/平方米的观测剔除

#描述分析：楼层

table(esf$楼层) #查看楼层分布。其中，“低层”、“中层”、“高层”均为普通住宅

#描述分析：建造时间

table(esf$建造时间) #查看建造时间分布,发现原始数据中部分年限观测值很少（该观测样本在前面的清洗工作中已被剔除）,且时间出现断层,考虑删除

esf=esf[esf[,5]>1973|esf[,5]==1936|esf[,5]==1953|esf[,5]==1954|esf[,5]==1957|esf[,5]==1958|esf[,5]==1964,] #对建造时间变量做部分删除处理

table(esf$建造时间)

#“详细地址”亦为文本型变量，不做处理

#描述分析：城区

table(esf$城区) #查看城区分布，崇明仅有6个观测,金山仅有17个观测，考虑删除

esf=esf[esf[,12]!="崇明"&esf[,12]!="金山",] #对城区变量做部分删除处理

table(esf$城区)

######查看删除的观测条数和比例

diff=n-dim(esf)[1] #删除的观测共1754条

round(100\*diff/dim(esf)[1],2) #删除观测所占比例为4.91%

######保存清洗后的数据

write.csv(esf,file="updateHouse.csv")

write.table(esf, file = "updateHouse.txt", fileEncoding = "UTF-8")

### （5）上海市二手房文本挖掘分析建模（R）

#install.packages("jiebaR")

#install.packages("wordcloud2",dep=T)

rm(list = ls())

setwd("C:/Users/吕港/Desktop/大三下/人工智能/大作业/房源数据分析/大作业最终文件") #设置工作路径

######读入数据和基本处理

esf=read.csv("updateHouse.csv",header=T) #读入清洗后的数据

##如果你用的是mac系统，请将上一句代码替换为以下代码

#esf<-read.table(file="esfsj.txt",header=T,sep="\t",fileEncoding="UTF-8") #读入清洗后的数据

#par(family="STXihei") #可实现在统计图中显示中文

#基本处理

esf=esf[,c(-1,-2)] #去掉序号

n=dim(esf)[1] #输出样本量。样本量为35706

summary(esf) #查看数据基本描述

#调整变量单位

esf$单价.元.m2.=esf$单价.元.m2./1000 #价格单位转换成千元

names(esf)[3]<-"单价.千元.m2." #重命名价格变量

esf$总价.元.=esf$总价.元./10000 #总价单位转换成万元

names(esf)[12]<-"总价.万元." #重命名价格变量

#调整城区和楼层的因子水平顺序，以便作图输出美观

esf$城区=factor(esf$城区,levels=c("浦东","闵行","宝山","徐汇","普陀","杨浦","长宁","松江","嘉定","黄浦","静安","闸北","虹口","青浦","奉贤","金山","崇明"))

esf$楼层=factor(esf$楼层,levels=c("高层","中层","低层"))

#将建造时间转换为三个level的factor变量

levels(esf$建造时间)[levels(esf$建造时间) %in% c("1936-2001")] <- "1936-2001" #comment:全部code统一使用“＝”或者是“<-”

levels(esf$建造时间)[levels(esf$建造时间) %in% c("2013","2014","2015","2016")] <- "2013-2016年"

levels(esf$建造时间)[levels(esf$建造时间)!=c("1936-2001","2013-2016")] <- "2002-2012年"

######对"介绍"变量开展文本分析

###第一步：分词

#安装和加载文本分析所需的程序包

##如未安装程序包，请先运行以下代码

#install.packages("jiebaR") #安装程序包

#加载程序包

library(jiebaR)

js=esf$介绍 #将变量“介绍”赋值给新变量“js”

lec=data.frame(js) #定义词语数据框

head(lec) #查看前几行，看是否有字符编码问题

n=length(lec[,1]) #获取数据集长度，n=35706

#文本预处理

res=lec[lec!=" "] #剔除缺失

res=gsub(pattern="http:[a-zA-Z\\/\\.0-9]+","",res) #剔除URL

res=gsub(pattern="[我|你|的|了|是]","",res) #剔除特殊词

res=gsub("[0-9０１２３４５６７８９ < > ~]","",res) #剔除数字等符号

#分词+频数统计+排序(降序)

words=unlist(res) #转化为字符型变量

#View(words)

top <- qseg[words]

#View(top)

top2 <- top[nchar(top)>1] #去除字符长度小于2的词语

toptable <- table(top2) #统计词频

#View(toptable)

top120 <- sort(toptable, decreasing = TRUE)[1:120] #降序排序，并提取出现次数最多的前120个词语

top120 #查看120个词频最高的

#View(top120)

#绘制词云

library(wordcloud)

bmp(file="d:/house.bmp", width = 600, height = 600)

par(bg = "white")

wordcloud(names(top120), top120, colors = rainbow(120), random.order=F)

dev.off()

###第三步：建立标签字典。从top500种提取53个关键词（提取标准是词频大于80），按照字面含义自行拟定7个标签：“交通优势”、“区位优势”、“户型优势”、“家装优势”、“价格优势”、“车位优势”和“产权优势”。关键词及其与标签的对应关系见“关键词对应.xlsx”

#分词+频数统计+排序(降序)

words=unlist(res) #转化为字符型变量

mixseg=worker("mix") #使用混合模型分词方法

word=segment(words,mixseg) #分词

word=lapply(X=word, FUN=strsplit, " ") #转化为列表

v=table(unlist(word)) #频数统计

v=sort(v,decreasing = T) #按词频降序排列

#建立词库

wordsData=data.frame(words=names(v), freq = v) #建立词库（包括词语和词频）

wordsData=subset(wordsData, nchar(as.character(words))>1) #过滤掉一个字的词语

wordsData=wordsData[,c(1,3)]

top500=head(wordsData,500)

keys=top500[top500[,2]>80,]

###第四步：提取衍生标签变量

as.character(esf$介绍) #将变量“介绍”转化为字符型

#提取标签变量“交通优势”

esf<-cbind(esf[,],交通优势=seq(0,by=0,length.out = n)) #在数据集最后一列后添加变量“交通优势”

esf$交通优势<-ifelse(regexpr("直通",esf$介绍)>1|regexpr("地铁",esf$介绍)>1|regexpr("沿线",esf$介绍)>1|regexpr("直达",esf$介绍)>1|regexpr("方便",esf$介绍)>1|regexpr("交通",esf$介绍)>1,1,0) #观测样本在变量“介绍”种若含有关键词“地铁”，其变量“交通优势”的取值为1，否则为0

esf$交通优势=factor(esf$交通优势,levels=c(0,1),labels=c("否","是")) #调整因子水平顺序

table(esf$交通优势)/n\*100 #展示变量频率分布(%)

#提取标签变量“区位优势”

esf<-cbind(esf[,],区位优势=seq(0,by=0,length.out = n)) #在数据集最后一列后添加变量“区位优势”

esf$区位优势<-ifelse(regexpr("学区",esf$介绍)>1|regexpr("黄金",esf$介绍)>1|regexpr("景观",esf$介绍)>1,1,0) #观测样本在变量“介绍”种若含有关键词“学区”等，其变量“区位优势”的取值为1，否则为0

esf$区位优势=factor(esf$区位优势,levels=c(0,1),labels=c("否","是")) #调整因子水平顺序

table(esf$区位优势)/n\*100 #展示变量频率分布(%)

#提取标签变量“户型优势”

esf<-cbind(esf[,],户型优势=seq(0,by=0,length.out = n)) #在数据集最后一列后添加变量“户型优势”

esf$户型优势<-ifelse(regexpr("户型",esf$介绍)>1|regexpr("朝南",esf$介绍)>1|regexpr("南北",esf$介绍)>1|regexpr("卧室",esf$介绍)>1|regexpr("阳台",esf$介绍)>1|regexpr("高区",esf$介绍)>1|regexpr("厨卫",esf$介绍)>1|regexpr("全明",esf$介绍)>1|regexpr("客厅",esf$介绍)>1|regexpr("采光",esf$介绍)>1,1,0) #观测样本在变量“介绍”种若含有关键词“户型”等，其变量“户型优势”的取值为1，否则为0

esf$户型优势=factor(esf$户型优势,levels=c(0,1),labels=c("否","是")) #调整因子水平顺序

table(esf$户型优势)/n\*100 #展示变量频率分布(%)

#提取标签变量“家装优势”

esf<-cbind(esf[,],家装优势=seq(0,by=0,length.out = n)) #在数据集最后一列后添加变量“家装优势”

esf$家装优势<-ifelse(regexpr("精装修",esf$介绍)>1|regexpr("装修",esf$介绍)>1|regexpr("精致",esf$介绍)>1|regexpr("拎包",esf$介绍)>1,1,0) #观测样本在变量“介绍”种若含有关键词“精装修”等，其变量“家装优势”的取值为1，否则为0

esf$家装优势=factor(esf$家装优势,levels=c(0,1),labels=c("否","是")) #调整因子水平顺序

table(esf$家装优势)/n\*100 #展示变量频率分布(%)

#提取标签变量“价格优势”

esf<-cbind(esf[,],价格优势=seq(0,by=0,length.out = n)) #在数据集最后一列后添加变量“价格优势”

esf$价格优势<-ifelse(regexpr("五年",esf$介绍)>1|regexpr("免税",esf$介绍)>1|regexpr("减税",esf$介绍)>1|regexpr("两年",esf$介绍)>1|regexpr("急售",esf$介绍)>1|regexpr("满五",esf$介绍)>1,1,0) #观测样本在变量“介绍”种若含有关键词“可按揭”等，其变量“价格优势”的取值为1，否则为0

esf$价格优势=factor(esf$价格优势,levels=c(0,1),labels=c("否","是")) #调整因子水平顺序

table(esf$价格优势)/n\*100 #展示变量频率分布(%)