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:

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

卧室带阳台，卧室全南，地铁房，低楼层

2室1厅 | 78.16平 | 低区/6层 | 朝南

仁德路67弄10支弄 | 杨浦 | 五角场 | 1997年建

距离3号线江湾镇站303米　满五

480 万

单价61412元/平

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。

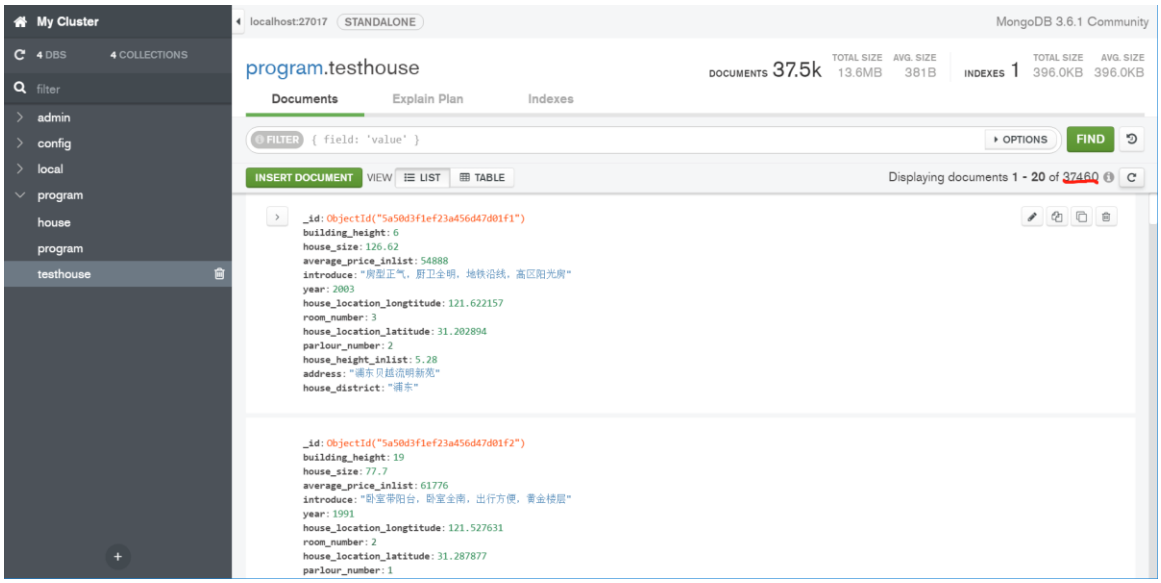
（2）数据说明

变量类型		变量名	详细说明	取值范围
因变量		单位面积房价	单位：元/平方米	20541-122448
		总价	单位：元	500000-120000000
自变量	内部因素	总面积	单位：平方米	21.17-740.75
		总厅数	定性变量	0-9
		总卧室数	定性变量	1-9
		楼层位置	定性变量	低层、中层、高层
		建造时间	单位：年	1936-2016
		详细地址	根据高德api取得的详细经纬度坐标	上海范围内
	外部因素	城区	定性变量：17个区	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、
		中介介绍	由链家网对房源的介绍得来	不定向

（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。）

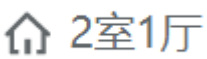


（3）数据处理

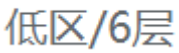
1．爬取过程中的处理

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

	厅数	n	price	area
	<chr>	<int>	<dbl>	<dbl>
1	0	3344	67439.83	43.34579
2	1	16713	58030.88	61.48102
3	2	16479	57321.30	112.82740
4	3	296	52157.79	180.12365
5	4	20	53397.20	211.55250
6	9	1	32522.00	245.98000

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

	卧室数	n	price	area
	<chr>	<int>	<dbl>	<dbl>
1	1	7901	64034.85	45.16230
2	2	18588	57950.36	76.39216
3	3	8820	55175.62	118.21593
4	4	1207	57585.54	168.93287
5	5	236	52140.04	191.76941
6	6	87	50808.95	206.07184
7	7	11	38827.00	211.70818
8	8	2	32694.00	212.52500
9	9	1	32522.00	245.98000

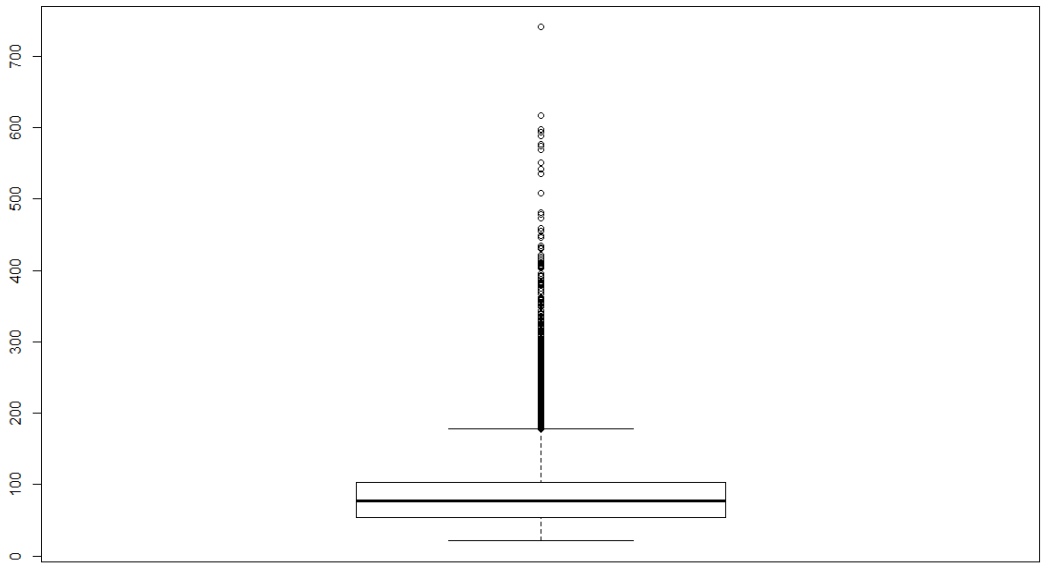
对房源的卧室数进行分析，对卧室数为 5 个和 5 个以上的记为 5+卧室

	城区	n	price	area
	<fctr>	<int>	<dbl>	<dbl>
1	宝山	3209	47932.95	80.88820
2	崇明	7	24595.86	73.91429
3	奉贤	2052	27281.64	98.73919
4	虹口	2819	67639.22	76.57950
5	黄浦	1789	89569.08	92.28102
6	嘉定	2837	39147.97	89.47786
7	金山	40	19958.45	95.67100
8	静安	954	93903.15	88.88341
9	闵行	3118	52118.09	88.11825
10	浦东	580	65324.42	79.87972
11	普陀	3112	62266.52	75.87960
12	青浦	1922	33946.73	101.63765
13	松江	2485	38694.70	96.31409
14	徐汇	3061	75493.49	75.89128
15	杨浦	3180	65867.62	69.82136
16	闸北	2741	63507.60	75.33514
17	长宁	2947	76012.69	85.06304

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

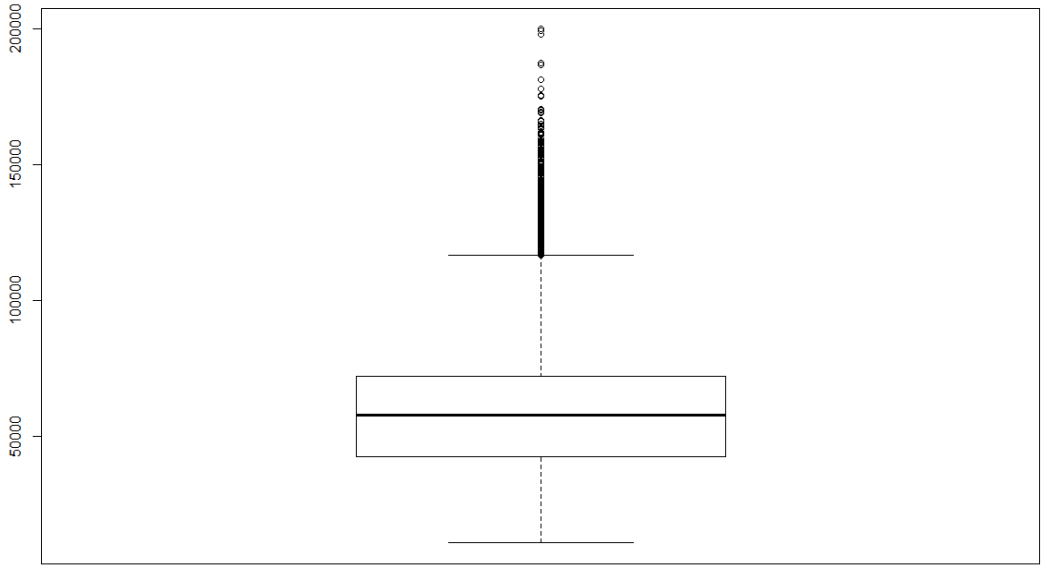
中的图会给出这种假设的原因），选取其中的 1%（30 平方米）和 99%（230 平方米）的数据。

1	2	3	4	5	6	7	8
7722	18756	8878	1104	193	57	7	1

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。

0	1	2	3	4
3128	16773	16560	237	12

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。

1911	1912	1922	1924	1926	1930	1931	1935	1936	1937	1939	1940	1948	1949	1950	1952	1953	1954	1955
2	1	1	7	2	4	2	4	33	4	5	1	5	2	1	1	30	35	6
1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974
29	111	108	6	26	3	18	4	55	28	13	12	2	7	1	4	7	26	30
1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993
73	82	96	216	150	234	325	411	435	427	559	679	911	691	720	795	669	696	1025
1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012
1983	2214	1853	1880	1502	1359	943	874	970	1263	1260	1746	1427	1080	967	780	836	1038	850
2013	2014	2015	2016															
543	480	236	49															

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。

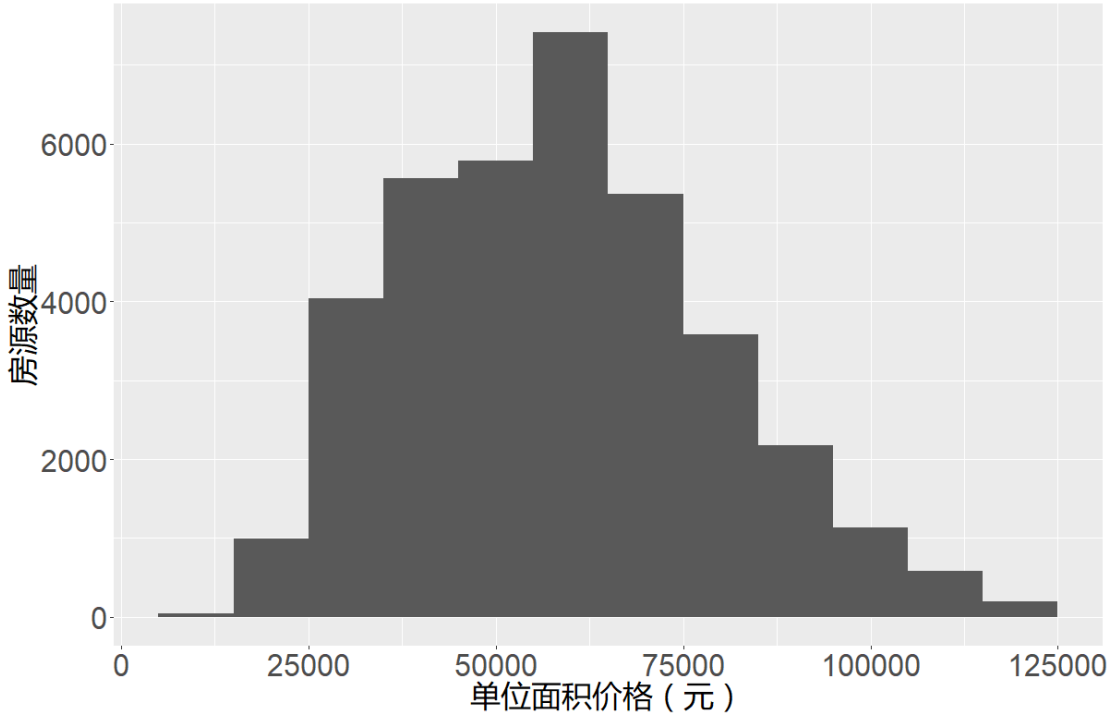
```
宝山 崇明 奉贤 虹口 黄浦 嘉定 金山 静安 闵行 浦东 普陀 青浦 松江 徐汇 杨浦 闸北 长宁
3187    6 1774 2725 1695 2821   17  940 3097  572 3041 1813 2470 2975 3060 2697 2839
```

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。

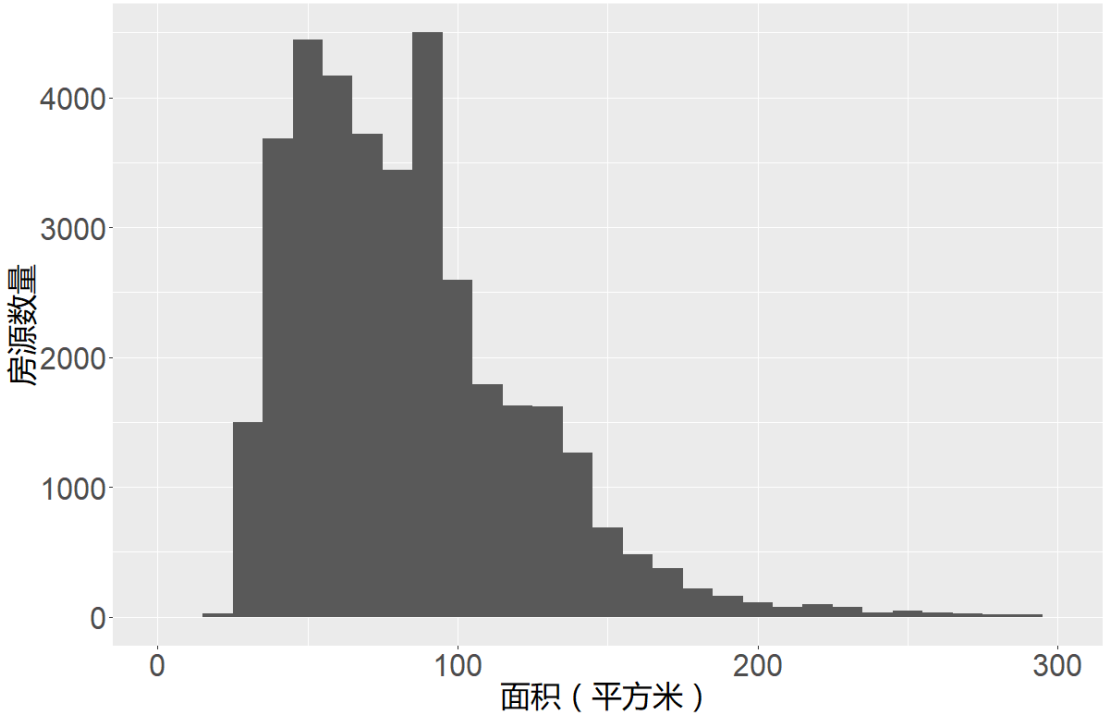
```
> ######查看删除的观测条数和比例
> diff=n-dim(esf)[1]                    #删除的观测共1754条
> round(100*diff/dim(esf)[1],2)          #删除观测所占比例为4.91%
[1] 4.91
```

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%。

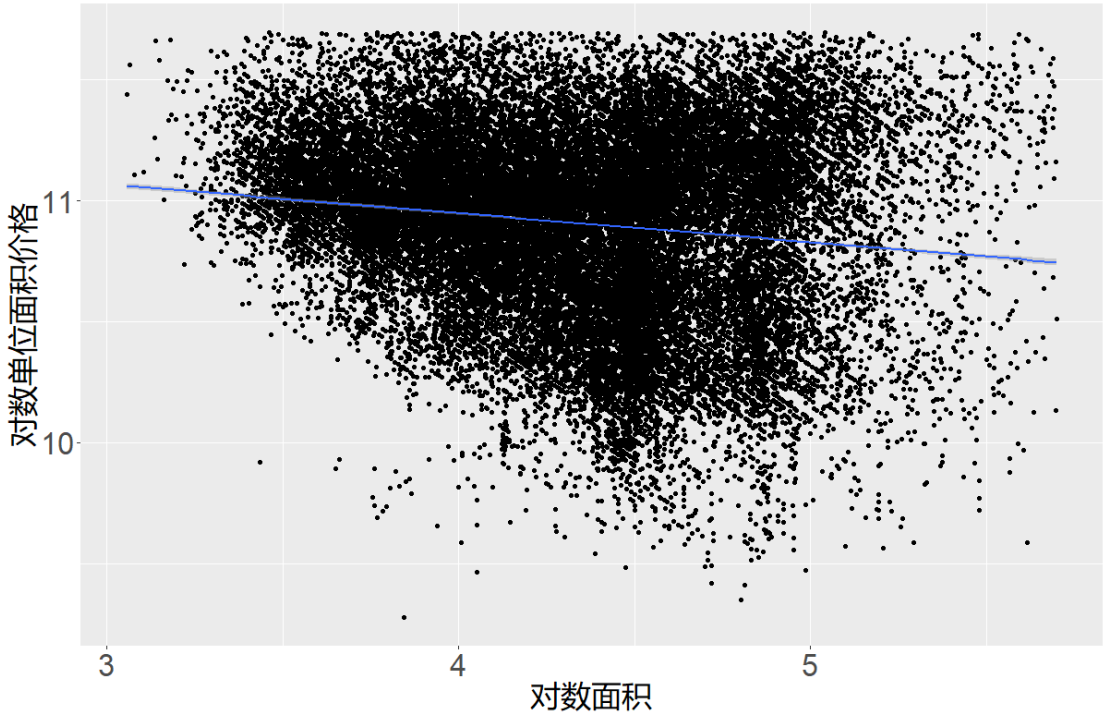
三、描述性分析



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/m^2。

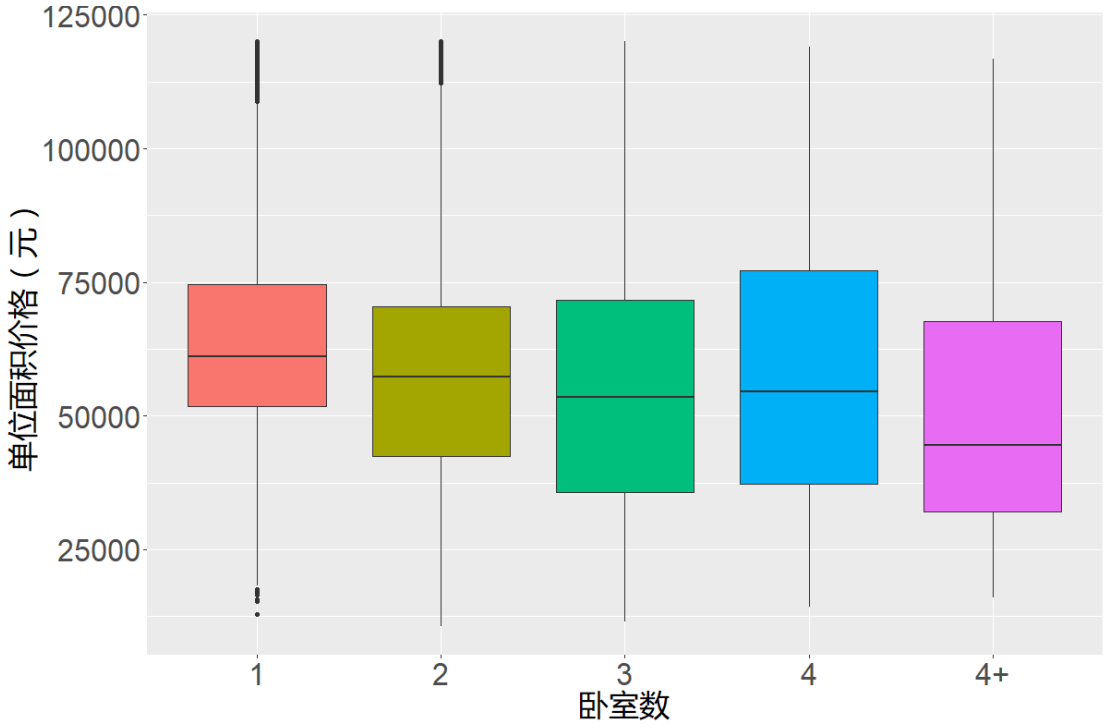


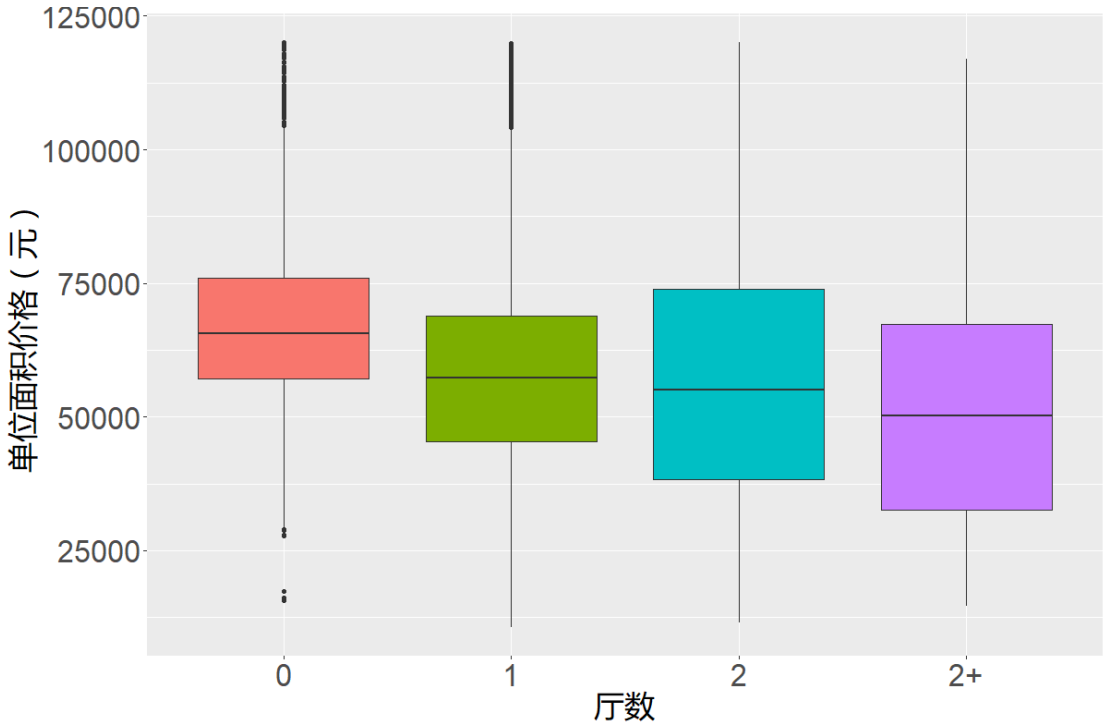
对上海市的二手房房源的面积进行分析的时候，可以发现上海市二手房的房源面积比较符合卡方分布，其面积大概在 150 平方米以下，尤其是 100 平方米以下占了大多数，这样的房源面积图示也与上海房价奇高有着不可分割的关系，据此估计上海市二手房的面积均值为 75m^2。

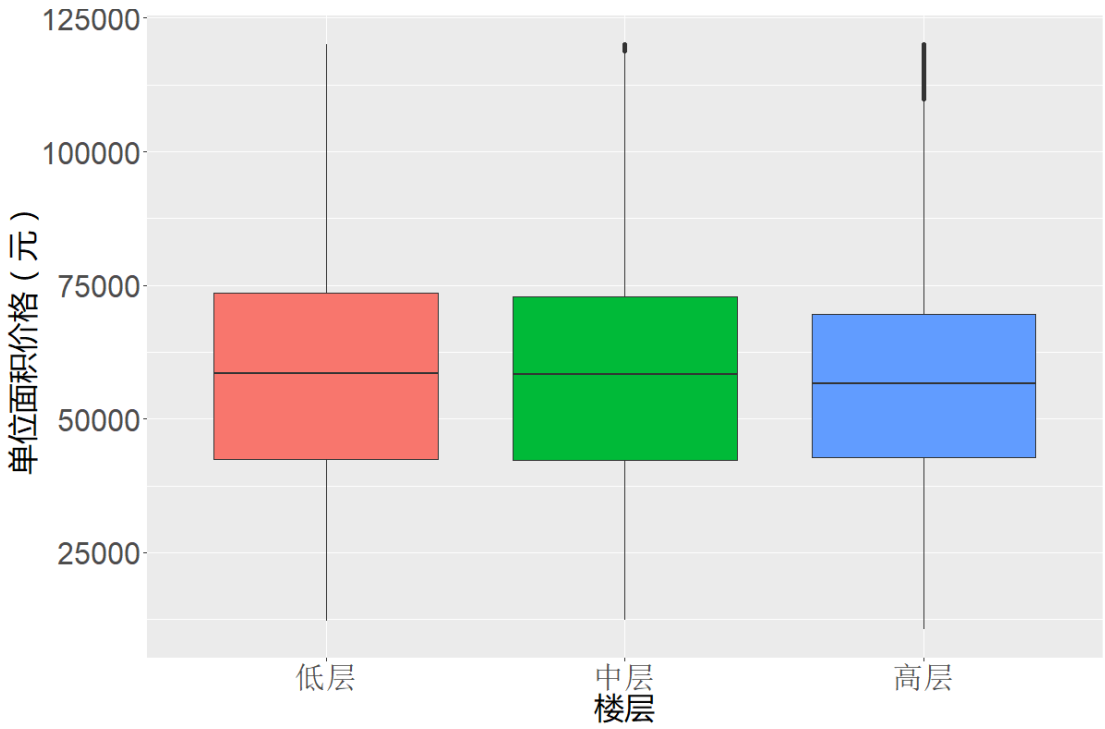


取面积对数和单位面积价格的对数进行作图，可以发现在上海市二手房的房源的单位面积价格是与面积成反比，这与编者最开始认为会是正比的想法有所出

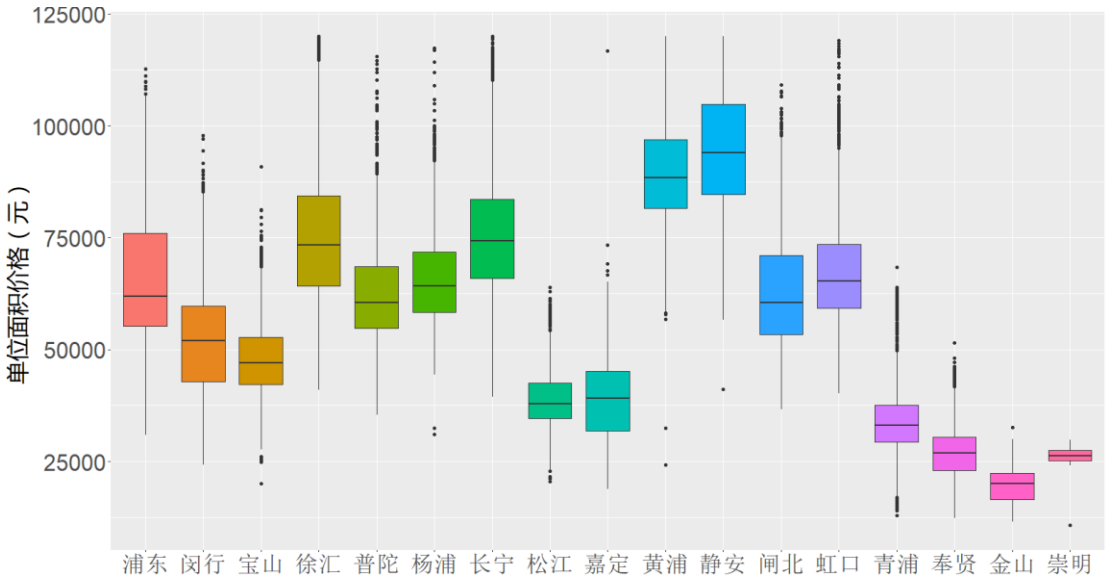
入，其产生的原因可能与房源总价有关系，因为太高少有人能负担的起所以才会产生这种反比现象，数据挖掘的科学性和事实性改变我对上海市二手房的走向的看法，可以很好的应用于日后买房，甚至可能出现同样的价格买更大的面积的现象出现。







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）。

变量类型	变量名	详细说明(高频词)	取值范围
自变量	交通优势	地铁、沿线、直达、直通、方便、交通	是(77.95%) 否
	区位优势	黄金、学区、景观	是(16.62%) 否
	户型优势	朝南、南北、卧室、阳台、厨卫、全明、客厅、采光、户型	是(86.72%) 否
	家装优势	精装修、装修、精致、拎包	是(36.18%) 否
	价格优势	五年、两年、减税、免税、急售、满五	是(9.18%) 否

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：


```
_id: ObjectId("5a50d3f1ef23a456d47d01f1")
building_height: 6
house_size: 126.62
average_price_inlist: 54888
introduce: "房型正气，厨卫全明，地铁沿线，高区阳光房"
year: 2003
house_location_longtitude: 121.622157
room_number: 3
house_location_latitude: 31.202894
parlour_number: 2
house_height_inlist: 5.28
address: "浦东贝越流明新苑"
house_district: "浦东"
```

```
_id: ObjectId("5a50d3f1ef23a456d47d01f2")
building_height: 19
house_size: 77.7
average_price_inlist: 61776
introduce: "卧室带阳台，卧室全南，出行方便，黄金楼层"
year: 1991
house_location_longtitude: 121.527631
room_number: 2
house_location_latitude: 31.287877
parlour_number: 1
house_height_inlist: 9.5
```


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。

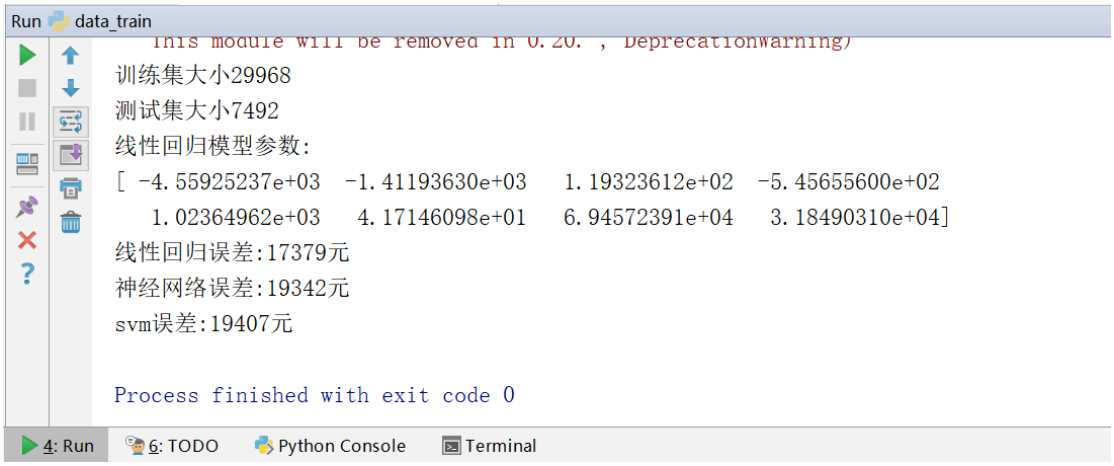
（2）模型调参

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。



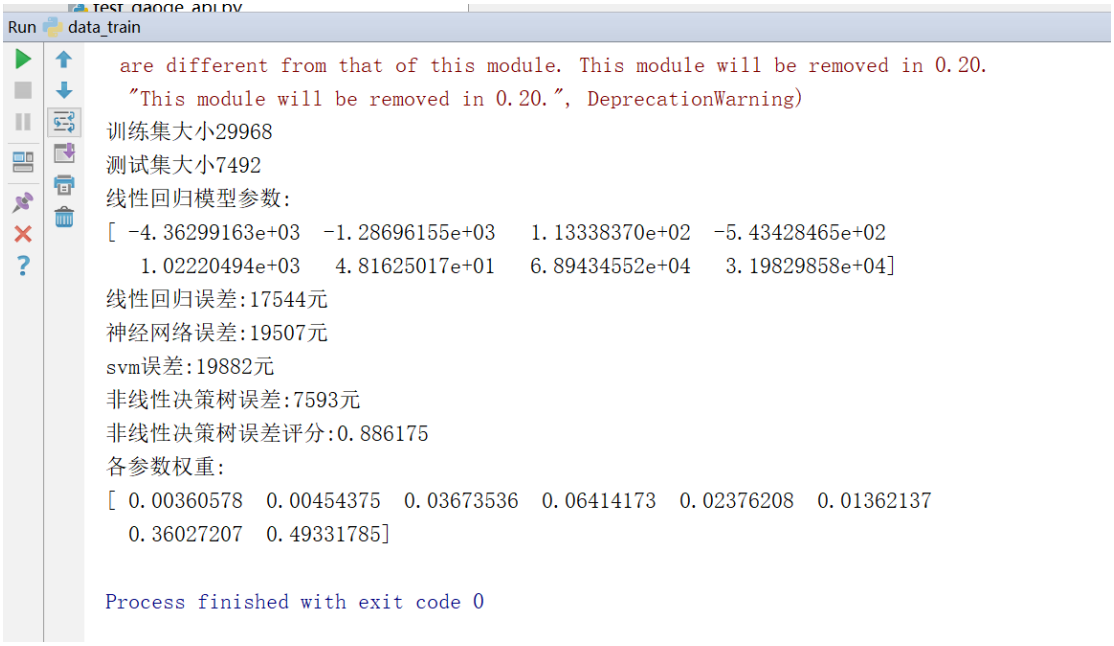
由上图可知，线性回归模型的表现较好，神经网络次之，支持向量机是最差的。另外，从线性回归模型的参数来看，每多一个房间，每多一个客厅，房子建的越晚，都会导致房价变低，这显然是与大众常识不相符合的。所以这个房价模型应该是一个非线性模型。

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。

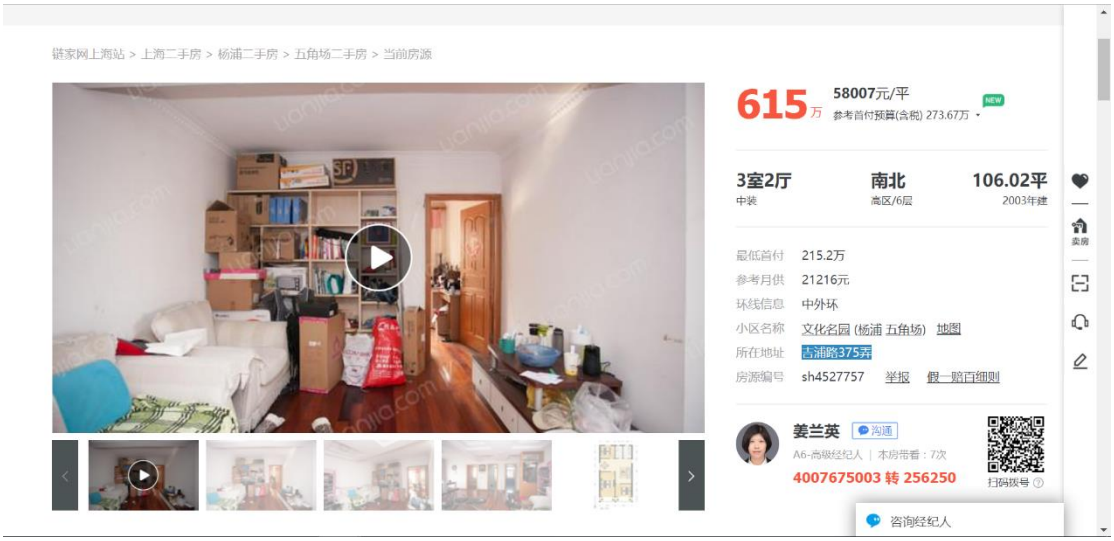


（2）结论

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。

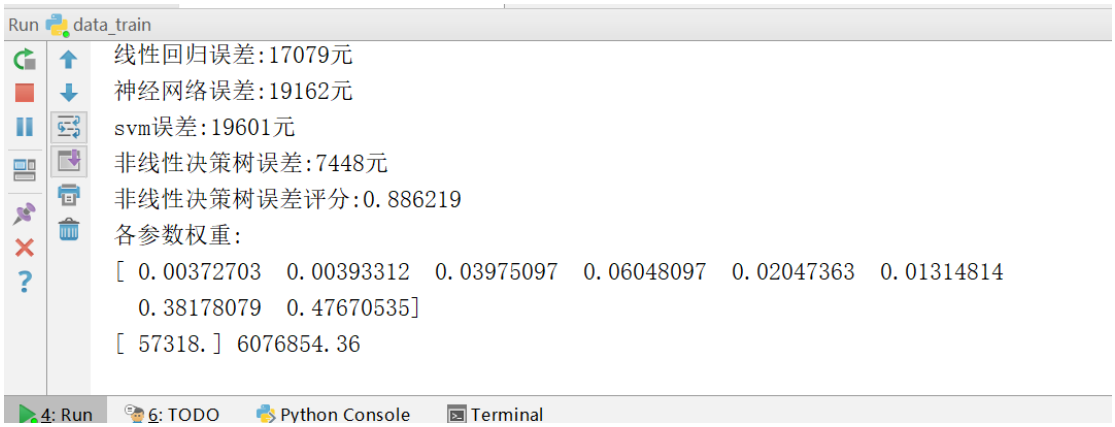
九、房价查询界面

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。

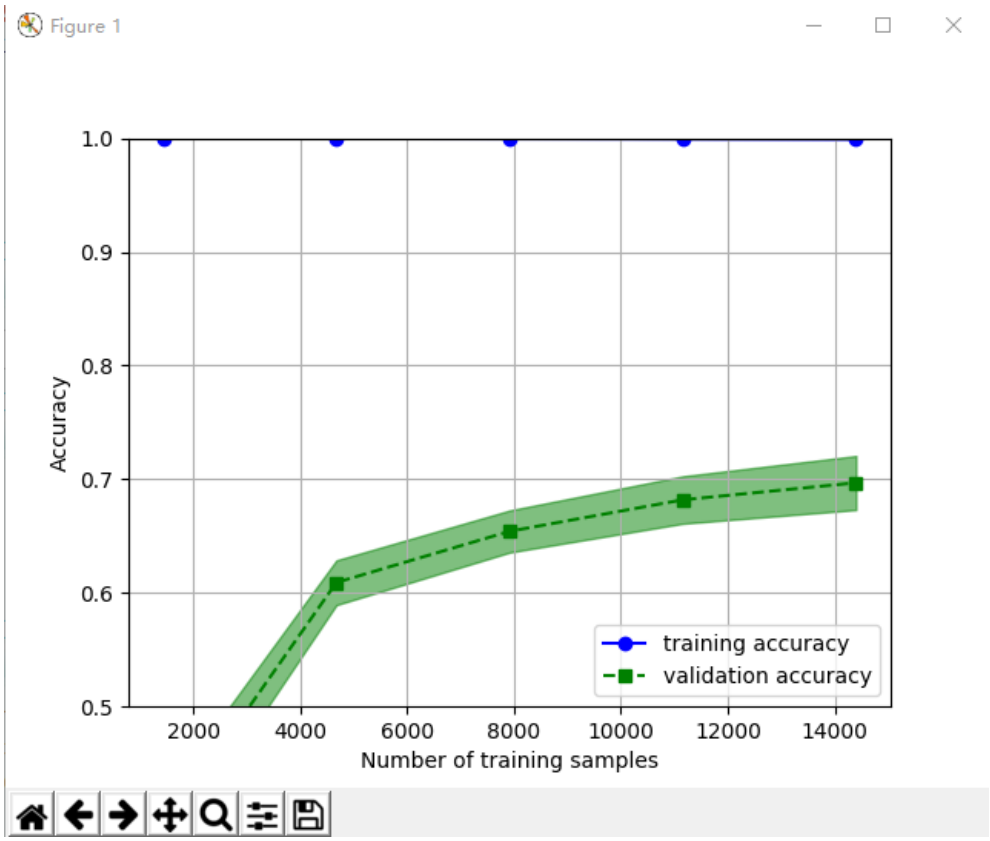
十、缺陷和改进措施

10.1 缺陷：

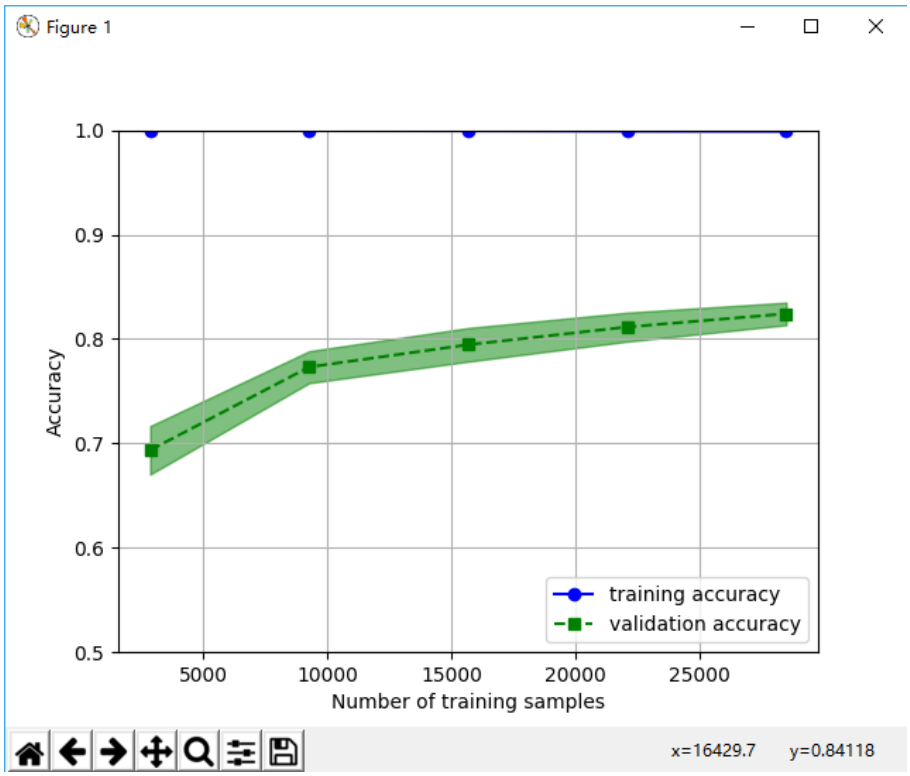
虽然采用非线性决策树模型模拟房价的效果较好，但还是有较大的缺陷，该模型对于房价不超过 700 万的房子模拟较好，误差一般在 15 万以内，当对于 700 万以上尤其是 850 万以上的房子模拟较差，对于 850 万以上的部分房子误差高达 100 万以上，得到的结果不是特别理想。

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

（1）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)
```

（2）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_siz
e"],item["year"],

item["building_height"],item["house_height_inlist"],item["house_locatio
n_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('u
tf-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()
```

（3）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"))
```

（4）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("[0-9 0 1 2 3 4 5 6 7 8 9 <>~]","",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(%)
```