
上海二手房价格预测

张庆儒 薛锐 杜佳骏 万俊成

Abstract

本项目对链家二手房交易平台的二手房数据，进行了有针对性的爬取，爬取了上海 17 个地区共 69000 多条的二手房数据，每条数据包含户型，楼层，单位均价等共 28 个属性特征。在数据清洗阶段，我们对存储在数据库的原数据进行了数据清洗，检查了特征完整性，将具有数值意义的文本数据转化为数值。在特征分析阶段，我们对房屋价格等总体性数据进行了统计分析，单独分析了每个特征下的房价分布，并对所有数值特征进行相关性分析。在数据预处理阶段，依据以上分析进行特征选择，对不同的特征依据其意义进行相应的预处。在回归建模阶段，使用多种经典回归算法对训练数据进行拟合、预测，此外，考虑到丰富的文本数据，我们也使用了 BiLSTM 模型对二手房的描述性文本进行建模，将 BiLSTM 加入到全连接网络中，完成对文本数据，数值数据，离散数据的统一拟合和预测。最终模型训练的结果在测试集上达到了 $R^2 = 0.91$ 的优异效果。

1 项目介绍

本项目对链家二手房交易平台的二手房数据，进行了有针对性的爬取，爬取了上海 17 个地区共 69000 多条的二手房数据，每条数据包含户型，楼层，单位均价，朝向，梯户比例，网页标题，房源特色文本等共 28 个属性特征。之后，我们对数据库的原数据进行了数据清洗，检查了特征完整性，将具有数值意义的文本数据转化为数值。并对房屋价格等总体性数据进行了统计分析，对每个特征的分布单独进行了比较分析，不同特征之间进行相关性分析。最后，依据以上分析进行特征选择，对不同的特征进行相应的预处理，使用多种经典回归算法对训练数据进行拟合、预测，此外，考虑到丰富的文本数据，我们也使用了 BiLSTM 模型对二手房的描述性文本进行建模，将 BiLSTM 加入到全连接网络中，完成对文本数据，数值数据，离散数据的统一拟合和预测。综上所述，本项目完成了以下内容：

- 使用 Python 和 Scapy 爬取了链家网站 **69905** 条二手房数据，使用 Xpath 对网页原数据进行解析，从而在每条数据中获取了单位均价，总面积，朝向，户型，标题，副标题，房源特色文本等 **28** 个属性。并利用百度地图 API，爬取了二手房所在小区的经纬度，和上海所有地铁站的经纬度，用于计算每条二手房到市中心的距离和到最近地铁站的距离。
- 对房屋价格分布，地区分布等进行总体统计分许描述。对包含有数值特征的文本数据进行处理，并将具有数值意义的特征进行相关性分析。
- 对于离散数值的特征，对比单位价格在不同离散值上的分布差异。选取价格分布差异性较大的特征作为预测模型的输入。

- 对数值数据，独热码数据以及多类别数据分别进行预处理。使用多种回归算法，拟合预处理后的数据，对比不同算法的预测误差。
- 使用 BiLSTM 模型对标题，副标题等文本数据进行建模，结合 BiLSTM 与 MLP，将预处理后的数据与文本数据一起作为输入，投入到模型进行训练，训练完成后在测试集上进行预测。



Figure 1: 全部房屋数据的单价分布立体图

2 数据爬取

2.1 数据源选择

本项目原定选择作业中的任务一，爬取安居客的上海二手房源信息。我们最开始尝试从列表显示中爬取房源，结果是共有 332679 套房源，但是列表中显示的每套房源的信息(如图2所示) 显然是不足以构建合适的模型的，因此我们还尝试爬取这 33 万条房源的详细信息。但是由于安居客比较严格的反爬虫机制，我们使用高于 2 秒访问一次的频率都会被识别出然后需要输入验证码，而我们没法自动破解验证码，因此我们无法在规定的时间内爬取这么多的数据。



Figure 2: 安居客列表显示的信息

后来和老师沟通之后，我们选择链家作为我们数据挖掘的数据源。链家上的信息较多，方便我们后续分析，我们总共爬取了 69905 条数据。

2.2 网页分析与爬虫编写

链家的上海二手房链接为 <https://sh.lianjia.com/ershoufang/>，但是链家每页只显示 30 条房源信息，最多显示 100 页，因此最多只有 3000 条数据，从这个入口进行爬取不能获取我们需要的全部房源信息。我们选择从单独爬取每个小地段（如浦东-临港新城）的所有房源信息，这样每个地段房源大多在 3000 条以内，不会爬取不完所有信息。

这样我们需要 3 个步骤，第一步，获取所有小地段的入口链接，第二步，从列表信息爬取所有房源的简单信息，第三步，爬取所有房源的详细信息。

2.2.1 获取地段列表的入口地址

获取地段列表的入口地址需要进行两级操作，先选择区，然后选择地段。通过进入链家上海二手房首页的筛选器分析可以得到各区的 URL，均为 <http://sh.lianjia.com/ershoufang/+> 区的结构，将其保存下来。然后进入各区的页面，其中找到如图3所示的筛选器组件，获取如“北蔡”，“川沙”等內容的 URL 地址（即 href 参数）。



Figure 3: 地段列表的入口

在这里，首先用 Xpath 方式定位到选择器组件，其 Xpath 地址为 “/html/body/div[3]/div/div[1]/dl[2]/dd/div[1]/div[2]”。然后使用 CSS 选择器，遍历其中满足 CSS 样式中含有“a”的组件，使用 Xpath 的 “//text()” 和 “//@href” 获取对应的 URL 和地段名称。爬取之后保存下来的文件内容如下：

```
1 嘉定安亭
2 ,, https://sh.lianjia.com/ershoufang/anting/嘉定丰庄
3 ,, https://sh.lianjia.com/ershoufang/fengzhuang/嘉定华亭
4 ,, https://sh.lianjia.com/ershoufang/huating/嘉定嘉定老城
5 ,, https://sh.lianjia.com/ershoufang/jiatinglaocheng/嘉定嘉定新城
6 ,, https://sh.lianjia.com/ershoufang/jiatingxincheng/嘉定菊园新区
7 ,, https://sh.lianjia.com/ershoufang/juyuanxinqu/
8 .....
```

2.2.2 从列表爬取房源简单信息

获取各个地段的入口链接之后，遍历其中的所有房源。房源列表的简单信息如下图4所示。这里使用 CSS 选择器（见表1）定位本页所有的房源信息，筛选满足 CSS 样式为 “.sellListContent li.info” 的组件就能定位到房源列表中每一项。然后在每一项中筛选如下的 CSS 样式就可以得到对应的信息。



Figure 4: 房源列表的简单信息

Table 1: 简单信息的 CSS 选择器

信息	CSS 选择器
标题	.title a::text
URL	.title a::attr(href)
小区	.address .houseInfo a::text
房屋信息	.address .houseInfo::text
位置信息	.flood .positionInfo::text
单价	.priceInfo .totalPrice span::text
总价	.priceInfo .unitPrice span::text

由于列表需要自动翻页，使用 CSS 选择器查找样式 “.content .leftContent .contentBottom .page-box::attr(page-data)” 定位到页码选择控件。其中的 paga-data 参数为 Json 文本，如“totalPage”:45,“curPage”:1，可以用于判断总共有多少页，以及是否到达最后页。

爬取到的数据存储在 SQLITE 数据库中，部分数据如下图5所示。

url	community	district	location	house_info	position_info	unit_price	vl_pr	title
http://...	万科金色领域	嘉定	菊园新区	[3室2厅] 83.84平米 南 北 精装 有电梯	中楼层(共18层)2014年建板楼	单价36141元/平米	303	店推荐好房、全明户型、三房精裝、采光好视野好!
http://...	竹园八区(嘉定)	嘉定	菊园新区	[1室1厅] 52.43平米 南 精装 无电梯	中楼层(共6层)1994年建板楼	单价32043元/平米	168	竹园小区(嘉定)1室1厅 168万
http://...	合景盈翠峰苑	嘉定	菊园新区	[3室2厅] 89.49平米 南 精装 有电梯	高楼层(共18层)2014年建板楼	单价32965元/平米	295	合景盈翠峰苑 3室2厅 295万
http://...	万科金色领域	嘉定	菊园新区	[3室2厅] 89.55平米 南 北 精装 有电梯	中楼层(共11层)2012年建板楼	单价32385元/平米	290	嘉定西、万科品质、满五唯一、临河视野好、看房方便
http://...	合景盈翠峰苑	嘉定	菊园新区	[2室2厅] 78.84平米 南 精装	低楼层(共11层)2012年建板楼	单价32979元/平米	260	11号线,精装两房,附带花园,生活便利,含车位,看房方便
http://...	日月光伯爵...	嘉定	菊园新区	[3室2厅] 90.07平米 南 精装	低楼层(共20层)2012年建板楼	单价35639元/平米	321	嘉定北站,精致装修,户型方正,采光明亮
http://...	嘉邦小区	嘉定	菊园新区	[2室1厅] 80.27平米 南 精装 无电梯	高楼层(共10层)1997年建板楼	单价26411元/平米	212	嘉邦小区 2室1厅 212万
http://...	日月光伯爵...	嘉定	菊园新区	[3室1厅] 90.73平米 南 精装	低楼层(共18层)2012年建板楼	单价35050元/平米	318	装修好的小三房,交通便捷,出行方便,诚意出售!
http://...	真宝名之湾(...	嘉定	菊园新区	[4室2厅] 96.5平米 南 北 毛坯	低楼层(共11层)板楼	单价37928元/平米	368	1楼带地下室,四房,采光问题,急售,停车位,有钥匙
http://...	嘉富小区	嘉定	菊园新区	[2室1厅] 70.8平米 南 精装 无电梯	中楼层(共6层)1994年建板楼	单价29944元/平米	212	满五唯一,产权清晰,房东诚意出售,户型微改更合理
http://...	万科金色领域	嘉定	菊园新区	[3室2厅] 89.26平米 南 北 精装 有电梯	高楼层(共11层)2013年建板楼	单价32714元/平米	292	嘉定西 万科精装3房一梯带50平花园 诚意出售
http://...	万科金色领域	嘉定	菊园新区	[3室2厅] 83.82平米 南 北 精装 有电梯	高楼层(共18层)2014年建板楼	单价35791元/平米	303	交通方便 嘉定西 万科物业,边套全明户型、看房方便
http://...	旭辉锦庭	嘉定	菊园新区	[3室2厅] 117.56平米 南 北 精装	低楼层(共14层)2014年建板楼	单价32324元/平米	380	旭辉锦庭 3室2厅 380万
http://...	清河路631弄	嘉定	菊园新区	[1室1厅] 39.96平米 南 精装 无电梯	中楼层(共6层)1996年建板楼	单价29041元/平米	115	清河路 小户型, 总价低, 看房方便, 诚意出售
http://...	合景盈翠峰苑	嘉定	菊园新区	[3室2厅] 89.66平米 南 精装 有电梯	高楼层(共18层)2014年建板楼	单价32345元/平米	290	11号线 嘉定西 小二房 合景盈翠峰苑 精装修
http://...	合景盈翠峰苑	嘉定	菊园新区	[2室2厅] 83.48平米 南 精装 有电梯	中楼层(共18层)2013年建板楼	单价31146元/平米	260	合景盈翠峰苑 2室2厅 260万
http://...	北水湾名邸	嘉定	菊园新区	[3室2厅] 115.96平米 南 精装	高楼层(共27层)板楼	单价37945元/平米	440	嘉定北站 带装修2房 满2年 有车位
http://...	新城悦尚城	嘉定	菊园新区	[2室2厅] 91.14平米 南 精装 有电梯	中楼层(共24层)2013年建板楼	单价31820元/平米	290	店推荐 嘉定西地铁口 中科院教育 南北通 带车位
http://...	合景盈翠峰苑	嘉定	菊园新区	[3室2厅] 86.56平米 南 精装 有电梯	中楼层(共18层)2012年建板楼	单价36391元/平米	315	11号线嘉定西地铁 精装三房 正气南北 临河采光
http://...	保利海上五月...	嘉定	菊园新区	[3室2厅] 91.96平米 南 其他 有电梯	高楼层(共14层)2009年建板楼	单价38061元/平米	350	保利海上五月花(公寓) 3室2厅 350万

Figure 5: 房源列表的简单信息

2.2.3 爬取所有房源详细信息

一般情况下有上述的简单信息已经可以进行预测分析了，但是为了获取更多信息，进行更加准确的分析，我们还爬取了房源信息的详细信息。详细信息包括图6中的简单信息，和房源详情页的基本信息和房源特色。

此外，在详细信息网页中还有小区的地理位置。由于地理信息写在 JavaScript 代码中，用 Xpath 或者 CSS 选择器无法获取到经纬度，因此我们使用正则表达式对网页中的文本进行匹配，得到了小区的经纬度。进行匹配的正则表达式为“resblockPosition.*?,”。

这样一来，就得到了链家上海所有二手房的详细信息，后续将围绕这些信息进行分析。

基本信息

基本属性	房屋户型	2室1厅1厨1卫	所在楼层	高楼层(共6层)
	建筑面积	74.45m ²	户型结构	平层
	套内面积	暂无数据	建筑类型	板楼
	房屋朝向	南	建筑结构	砖混结构
	装修情况	精装	梯户比例	一梯四户
	配备电梯	无	产权年限	未知
交易属性	挂牌时间	2019-05-03	交易权属	商品房
	上次交易	2000-10-16	房屋用途	普通住宅
	房屋年限	满五年	产权所属	共有
	抵押信息	无抵押	房本备注	已上传房本照片

Figure 6: 房源详情的基本信息栏目

Table 2: 详细信息的 Xpath 或 CSS 选择器

信息	Xpath 或 CSS 选择器
标题	.sellDetailHeader .title-wrapper .content .title .main::text
副标题	.sellDetailHeader .title-wrapper .content .title .sub::text
小区	.overview .content .aroundInfo .communityName .info::text
房屋户型	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[1]/text()
所在楼层	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[2]/text()
建筑面积	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[3]/text()
可用面积	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[5]/text()
户型结构	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[4]/text()
建筑结构	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[6]/text()
房屋朝向	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[6]/text()
建筑类型	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[8]/text()
装修情况	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[9]/text()
梯户比例	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[10]/text()
配备电梯	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[11]/text()
产权年限	//*[@id='introduction']/div/div/div[1]/div[2]/ul/li[12]/text()
交易权属	//*[@id='introduction']/div/div/div[2]/div[2]/ul/li[2]/span[2]/text()
房屋用途	//*[@id='introduction']/div/div/div[2]/div[2]/ul/li[4]/span[2]/text()
房屋年限	//*[@id='introduction']/div/div/div[2]/div[2]/ul/li[5]/span[2]/text()
产权所属	//*[@id='introduction']/div/div/div[2]/div[2]/ul/li[6]/span[2]/text()
抵押信息	//*[@id='introduction']/div/div/div[2]/div[2]/ul/li[7]/span[2]/text()
房源标签	.baseinform .introContent .tags .tag::text
房源特色	.baseinform .introContent .baseattribute
单价	.overview .content .price .text .unitPrice .unitPriceValue::text
总价	.overview .content .price .total::text

3 特征数据分析

本节采用特征工程的分析方法，针对爬取的数据的各项特征进行比较全面的挖掘分析。首先我们对所得到的数据总体进行了统计分析，展示上海地区房屋价格的总体分布特征。之后对于数值型数据，我们采用相关性系数分析并量化连续型数值特征间的相关性，分析数值数据间的关联程度，选取关联性强的特征作为后续回归模型的输入。而对于离散型数据，其包含了类别信息，我们对各个类别信息特征进行了统计分析，对每个特征，对比房屋单位面积均价在不同种类间的分布差异，选择分布差异性大的特征作为回归模型的输入。

3.1 房价总体分布特征

如图7所示，通过观察单位面积房屋价格分布图可以看出，上海地区的二手房屋单价大部分分布在10万元一下，集中在5万左右一平米。通过观察房屋总价分布图可以看出，上海地区二手房总价集中分布在500万~700万之间。通过观察房屋面积分布图可以看出，上海地区二手房面积大多集中在100平方米以下。最后，对爬取的所有房价信息，利用其经纬度信息和单位房价，使用PowerMap绘制上海地区房屋单价分布立体图和房屋单价分布热力图来可视化各个地区的房屋价格水平。如图1, 7d所示，可以明显看出市中心的价格高，房源集中。

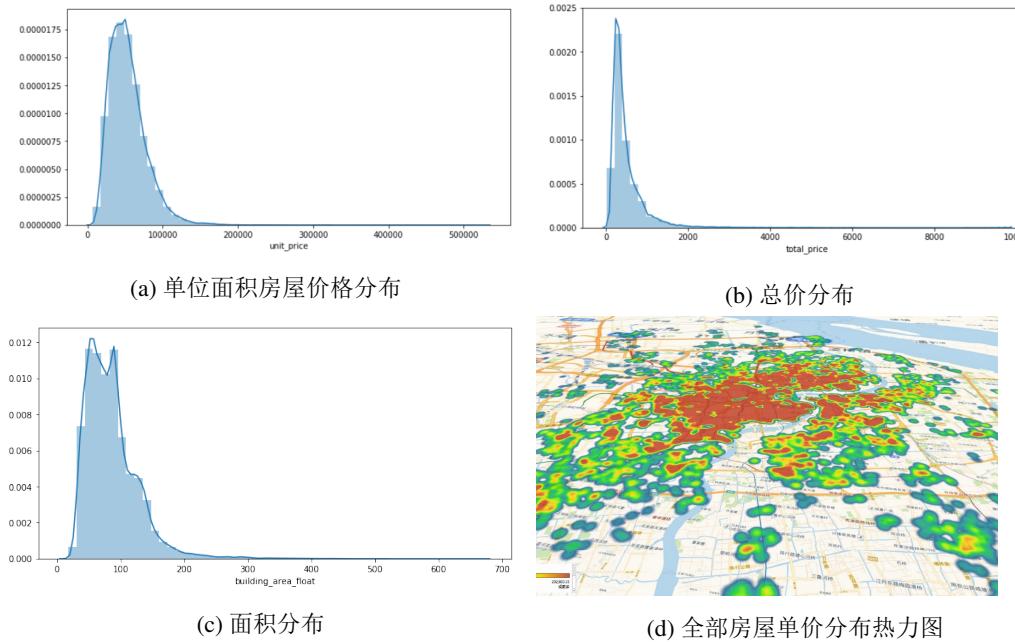


Figure 7: 数据特征总述

3.2 相关性分析

使用绘制热力图的方法来对容易数值化的特征进行相关性分析。图形化的结果见图8a，颜色越深表示正相关程度越高；反之，反相关程度越高。相关性分析包括了单位面积价格、总价、户型、楼层、面积、电梯率、距市中心距离和距地铁站距离。

通过绘制热力图并观察相关性矩阵，可以注意到单位房价和户型与面积的相关性系数很低，和距离地铁站和市中心的距离反相关性较高。房屋总价与户型和房屋面积相关性较高，与距离地铁站和市中心的距离具有较高的反相关性。同时，房屋总价与电梯率也具有

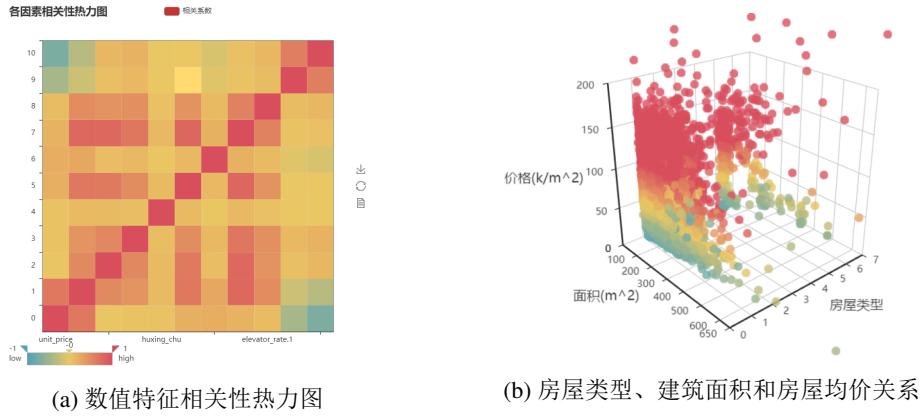


Figure 8: 各地区的房屋单位面积分布图

一定程度的相关性，大约在 0.5 左右，推断是因为在大面积的楼盘中，电梯率较高。而楼层对其他特征几乎不存在相关性。

图8b采用三维散点图的方式，展示了房屋类型、建筑面积和房屋均价三者之间的关系。为了方便显示，用编号 1~7 分别代表普通住宅，商业办公类，老公寓，别墅，新式里弄，花园洋房，旧式里弄。通过旋转观察可以看出普通住宅的数据点远远多于其他 6 种住宅类型。同时，各类型房屋均价分布均比较集中，没有看出哪个类型的房屋价格明显高于其他六种。同时，各房屋的建筑面积也大同小异，普通住宅由于数据量比较大的原因，建筑面积大的房屋数量比较多。

3.3 类别型离散数据特征分析

3.3.1 不同地区房价分布差异

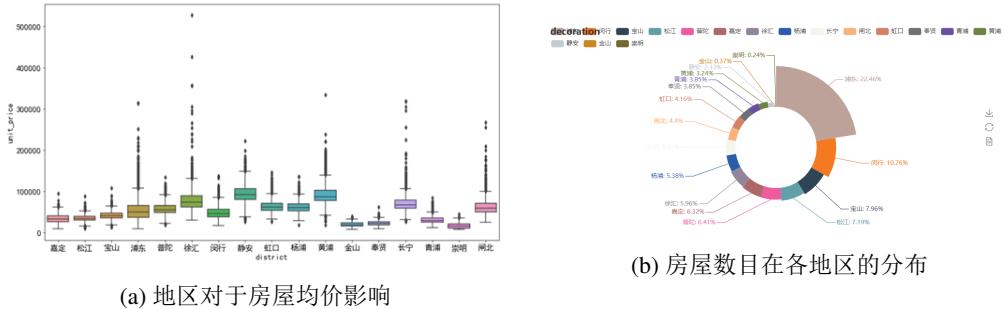


Figure 9: 各地区的房屋单位面积分布图

本次房屋数据采集基本覆盖了上海的所有地区。如图9b所示，浦东地区采集到的二手房数量最多，占据总数据量的 1/4 左右。而闵行、宝山、松江紧随其后，基本与上海面积最大的几个区相吻合，即区域面积越大，对应的二手房数据越多。采集到的数据量最少的是崇明，该地区处于上海较偏远地区，二手房数量较少是意料之中的。可见数据的采集保证了合理的随机分布。图9a用箱线图的方式，展示了地区对于房屋均价的影响。基本可以得出静安、黄浦、徐汇三个区域是上海平均房价最高的三个区域，而金山、奉贤和崇明则是上海平均房价最低的地区。并且房屋均价高的地区，价格分布差异化较大，而房屋均价低的地

区价格分布比较集中，这与常规的经验认知相符合。在该特征上，房价分布差异大，所以选为最终的训练数据。

如图10a, 10b所示，通过对各地区的房屋单位面积价格及总价对比可以看出，黄浦、静安和徐汇是上海房价最贵的三个地区，而奉贤、金山和崇明则是上海房价最低的三个地区。这与上文中的箱线图得到的结论一致。

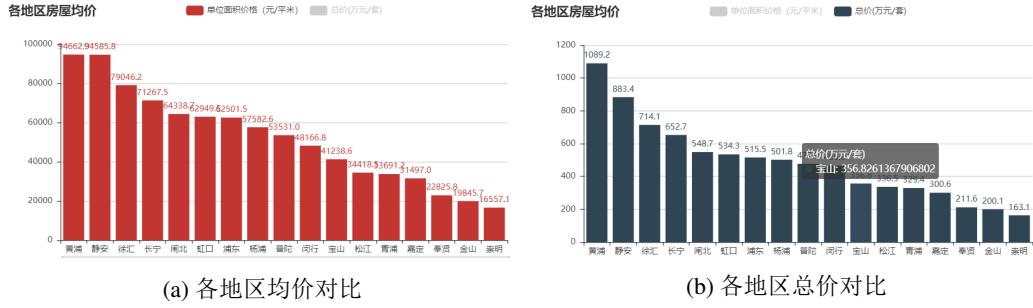


Figure 10: 地区特征分析图

3.3.2 不同朝向对房价分布的影像

采用箱线图的方式，图11a展示了南北朝向对于房屋均价的影响。可以看出东西朝向的平均房价比南北朝向的平均房价要高，这与我之前的认知不符。因为地理学过，北半球的房子大都坐北朝南，以求获得更多的光照。在图11b可以看出，大多数房子均是坐北朝南，但是爬取的数据客观显示出东西朝向房屋具有较高的市场期望。在该特征上，房价分布差异较大，选为最终的训练数据。

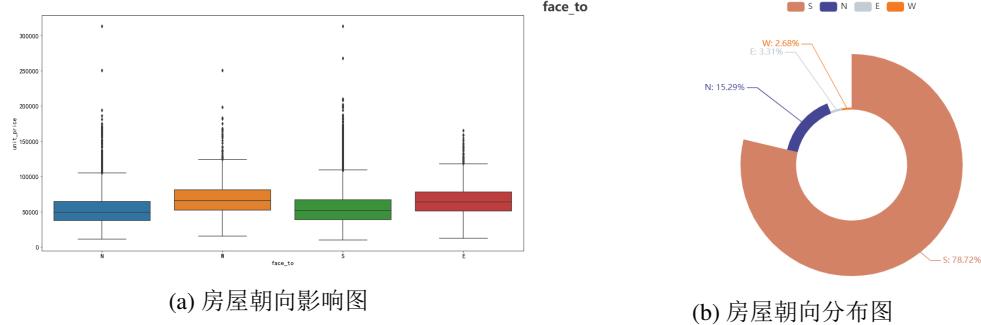


Figure 11: 房屋朝向特征分析图

3.3.3 不同房屋性质房价分布差异

图12a用箱线图的方式，展示了房屋性质对于房屋均价的影响，可以看出商品房和售后公房的房屋均价大抵相同，而动迁安置房的房屋均价则比较低。图12b用饼图的方式，可以看出房屋性质中商品房所占的比例最高。在该特征上，房价分布差异较大，选为最终的训练数据。

3.3.4 不同房屋类型的房价分布差异

图13a用箱线图的方式，展示了房屋类型对于房屋均价的影响，可以看出别墅以及商品办公类住房的房屋均价分布较为集中，而花园洋房和新式里弄的房屋均价分布则较为分散。虽然普通住宅有很多数据在图中显示偏离中心点，但是从图13b可以看出 95% 以上住宅类

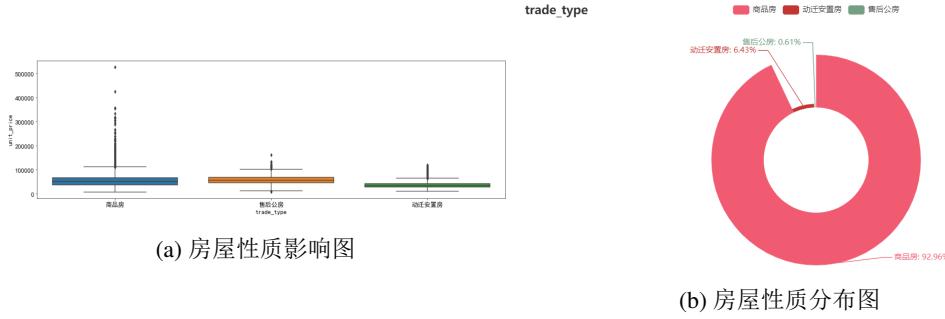


Figure 12: 房屋性质特征分析图

型均为普通住宅，故仍然认为其房屋均价分布较为集中。在该特征上，房价分布差异较大，选为最终的训练数据。

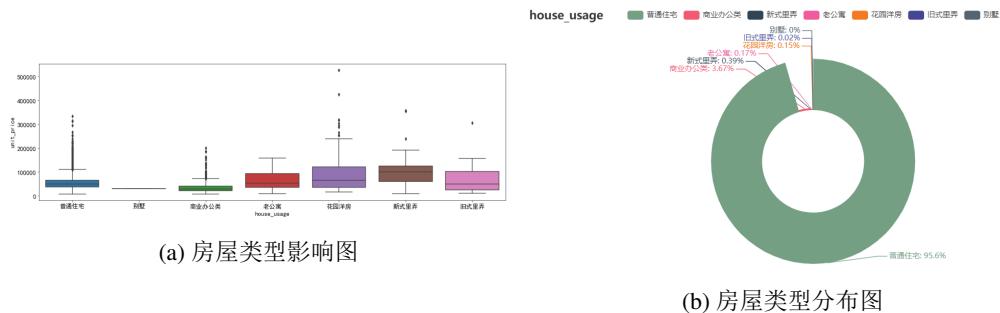


Figure 13: 房屋类型特征分析图

3.3.5 不同建筑类型的房价分布差异

用箱线图的方式，图14a中展示了建筑类型对于房屋均价的影响，明显可以看出砖木结构的房屋均价较高，这可能是因为许多高端住宅采用此种建筑类型导致。图14b显示砖混结构和钢混结构占据房屋总数的 90% 以上。在该特征上，房价分布差异较大，选为最终的训练数据。

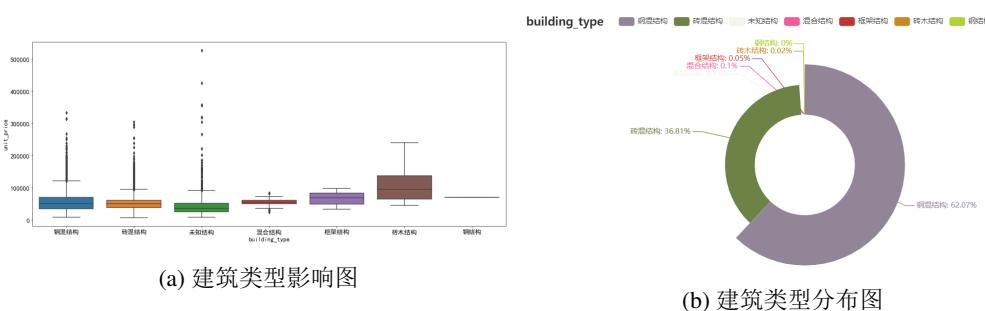


Figure 14: 房屋建筑类型特征分析图

3.3.6 不同建筑结构的房价分布差异

用箱线图的方式，图15a中展示了建筑结构对于房屋均价的影响，主要有板楼、塔楼、塔板结合和平房四种。其中，塔板结合的房屋均价最高，而平房的房屋均价则最低。通过图15b可以看出，建筑结构为板楼的房屋占比最高，大约占据了 90%。在该特征上，房价分布差异较大，决定选为最终的训练数据。

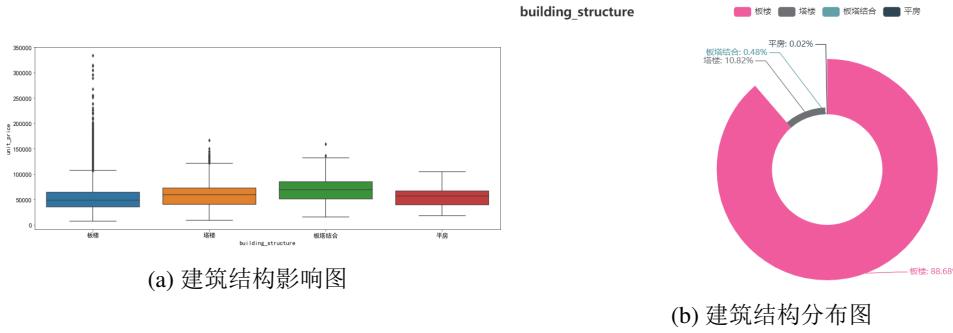


Figure 15: 房屋建筑结构特征分析图

3.3.7 不同装修程度的房价分布差异

图16a采用箱线图的方式，展示了不同装修程度对于房屋均价的影响，可以看出，精装修的房价明显较高，而毛胚房的房价则相对较低。图16b中可以看出，精装修的占比达到了50%，而毛胚只有8%，可以大致得出目前的在售房屋以精装修为主的结论。在该特征上，房价分布差异较大，选为最终的训练数据。

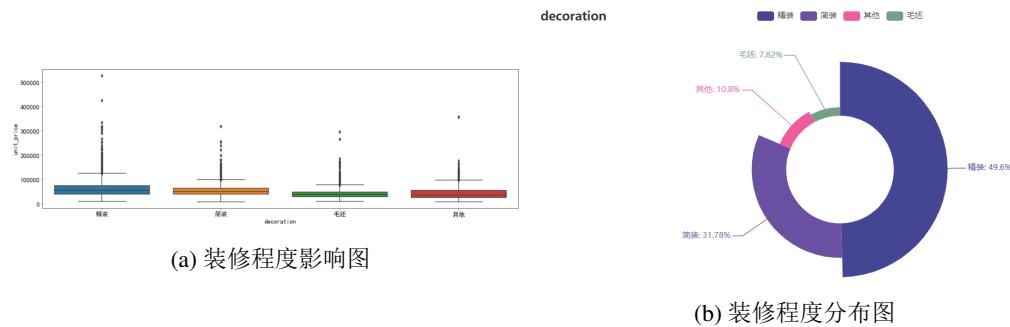


Figure 16: 装修程度特征分析图

3.3.8 电梯有无的房价分布差异

图17a采用箱线图的方式，展示了电梯有无对于房屋均价的影响。可以看出，有电梯的房屋均价高于无电梯的房屋。通过图17b可以看出，有无电梯的房屋数量基本持平。在该特征上，房价分布差异较大，选为最终的训练数据。

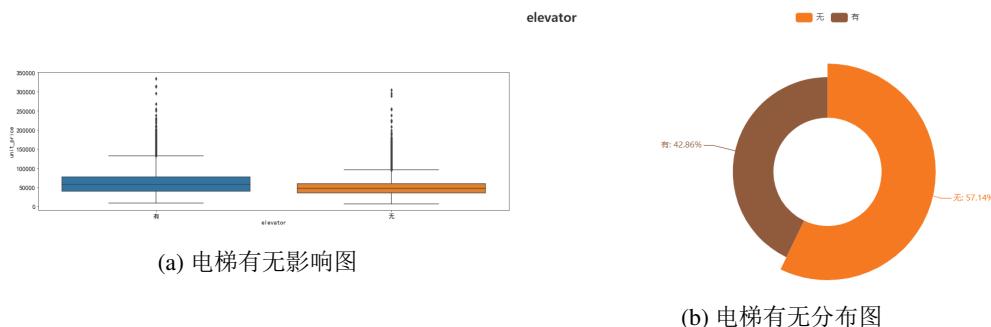


Figure 17: 电梯有无特征分析图

3.3.9 不同户型种类房价分布差异

图18a采用箱线图的方式，描述了不同户型种类房价分布差异，可以看出4种户型种类的房价分布基本相同。通过分布图18b可以看出，平层占据的比例很高，达到了95%以上。在该特征上，房价分布差异不大，决定舍弃该数据，不作为最终的训练数据。

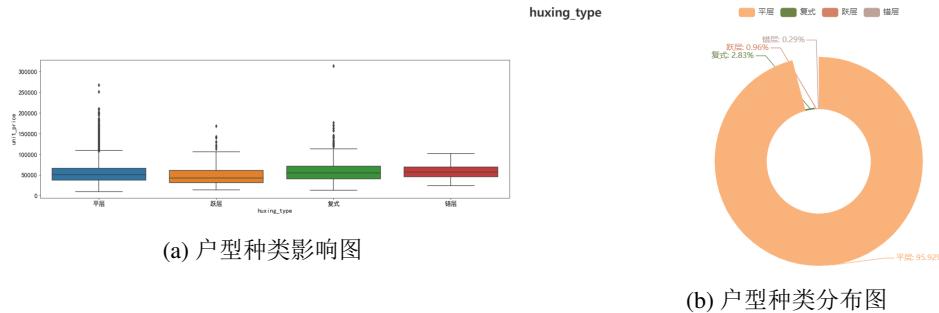


Figure 18: 户型种类特征分析图

3.3.10 不同产权所属的房价分布差异

图19a采用箱线图的方式，展示了产权所属为共有和非公有对于房屋价格的影响，可以看出二者差距不大。通过图19b可以看出，这两类产权的房屋数量大约为2:1。由于在该特征上，房价分布差异不大，决定舍弃该数据，不作为最终的训练数据。

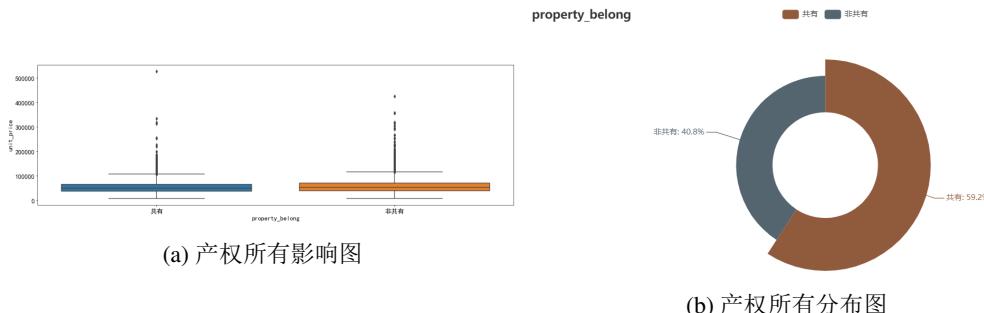


Figure 19: 产权所有特征特征分析图

3.3.11 不同户型的房屋数量分布差异

如图20所示，通过观察对应户型分布可以看出，卧室数量为2的占了1/2左右，数量为1、3的也各占据了1/4左右；客厅数量为1、2的各占据了1/2左右；厨房数量几乎全为1个；卫生间数量为1的二手房占据了3/4左右。可以看出不同户型的分布差异较大，可以作为最终的训练模型。

4 特征选择与数据预处理

经过上述的特征数据分析，我们选择了以下的特征(见表3)作为数据输入，进行后续的数据预处理，和回归模型拟合。数据在预处理前存放在 integral_data.csv 中，经过相应的预处理工作后，数据存放在 final_training_data.csv 中，在预处理过程中，对于中文形式的连续值属性，编写相应函数进行处理(详见代码清单)对于具有类型意义的属性，使用了独热编

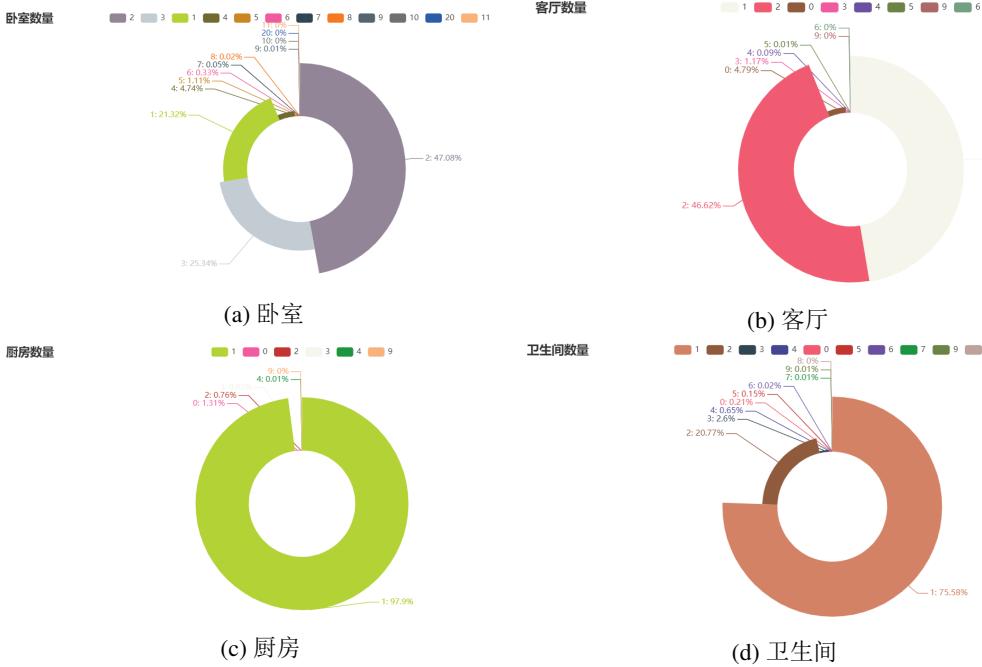


Figure 20: 户型房屋数量分布差异图

码的方式处理具有类型意义的离散值属性，该过程使用了 `pandas.get_dummies` 函数；对于取值为二分类的属性，如有无，使用 0-1 进行表示。各个选择特征详细的预处理方式如下：

4.1 数值数据预处理

从网站上爬取的大部分数据为字符串形式，一些数据甚至为中文形式的数值，有些属性则隐含了数值意义，因此，我们需要对其进行数字化，从而便于模型的训练和可视化工作。

- 房屋户型 **huxing**

房屋户型原数据为字符串形式，包含四个数值属性，分别为室厅厨卫的个数，如“10 室 2 厅 3 厨 5 卫”，对于该属性，我们将其拆分为四个独立的特性，分别为 `huxing_shi`, `huxing_ting`, `huxing_chu`, `huxing_wei`, 分别代表该房屋的室厅厨卫数量，数据类型为 int。在预处理过程中，我们使用了 python 正则表达式模块 `re.split` 依据“室厅厨卫”作为分隔符划分字符串，进而得到四个属性的整数值。

- 所在楼层 **floor**

所在楼层原数据格式为字符串，如：“高楼层 (共 18 层)”，我们对该字符串的处理方式为楼层数字乘以高中低楼层对应的因数，高楼层为楼层数乘以 1，中楼层为楼层数乘以 0.5，低楼层为楼层数乘以 0。对字符串使用正则表达式提取处楼层数后，乘以相应因子，最终的浮点数数值作为输入数据。

- 房屋面积 **building_area_float**

房屋面积原数据格式也为字符串，数据示例为：“89.49 m²”，该字符串的格式非常规整，直接去掉最后一个字符“m²”后，使用 `float` 函数的到预处理后的浮点数据。

- 梯户比例 **elevator_rate_float**

梯户比例原数据格式为字符串，数据示例为：“三梯十五户”，“八梯一百三十五

户”的中文数字格式，对该数据的处理方式为先使用正则表达式提取出梯数，如“三”，“八”，和户数，如“十五”，“一百三十五”，自己编写中文数字转阿拉伯数字的函数，详见代码清单??，输入该中文字字符串，输出 int 类型的数据。

- 到最近地铁站距离 **dist_sto**

在数据爬取阶段，我们也爬取了每个房屋所在小区的经纬度信息，存储在另一张表中，房屋信息表和小区信息表用小区在链家中的 url 内连起来，每个房屋的经纬度数据均使用所在小区的经纬度数据。同时，我们还通过百度地图 API 爬取了上海所有地铁站的经纬度，在每条房屋数据预处理的过程中，我们使用 numpy 库，将该房屋的经纬度数据减去所有地铁站的经纬度向量，取绝对值，再将经纬度的差值绝对值相加，取结果向量的最小分量作为该房屋到最近地铁站距离的数据（与真实距离相差一个常因数，该常因数为地球半径）。

- 到市中心距离 **dist_sta**

同样，处理方式与最近地铁站距离的处理方式类似，不过相减的对象为市中心经纬度 121.4737, 31.2304。

4.2 0-1 数据预处理

所选择属性中，也包含 0-1 二进制数据。如，elevator 电梯有无属性。对他们的预处理，只需依据是否出现，数值化为 0, 1。

- 有无电梯 **elevator**

由前一部分的特征分析得处，房屋单位均价的分布在有无电梯两种情况下存在交大差异，因此选取其作为输入特征之一。该属性原数据为字符串“有”或“无”，直接通过判断映射为 0-1 int 类型的数据。

- 房屋朝向 **face_to**

房屋朝向属性原数据为字符串，其为“东南西北”中的单个，两个或三个字符的组合，对其的处理方式为将该属性拆分为四个单独的属性 **face_E, face_W, face_S, face_N** 分别表示“东南西北”四个字符是否在该字符串中出现，如出现对应属性为 1，否则为 0。

4.3 独热编码处理类型数据

在选择的属性中，其他属性均为表述类型的数据，如建筑结构，建筑类型，房屋用途，所在地区等，对这些选择的属性的预处理方式，使用 `get_dummies` 函数，依据类型的个数对属性进行独热编码。

- 所在区域 **district**

所在区域的原数据格式为字符串，爬取的数据包含了上海 17 个区的房屋价格信息，这 17 个区分别为：‘嘉定’；‘松江’；‘宝山’；‘浦东’；‘普陀’；‘徐汇’；‘闵行’；‘静安’；‘虹口’；‘杨浦’；‘黄浦’；‘金山’；‘奉贤’；‘长宁’；‘青浦’；‘崇明’；‘闸北’。由于该列数据没有数值意义，仅包含一定的类型信息，由上一节的特征分析，可以得出，不同地区的房价分布存在很大的差异，因此适合使用独热编码方式处理。具体处理方式为，对 **district** 一维数据使用 `get_dummies` 函数进行独热编码，将其编码为 17 维的向量数据，连接到原表中。

- 房屋结构 **building_structure**

房屋结构的原数据格式为字符串，爬取的数据包含了‘板楼’，‘塔楼’，‘板塔结合’，‘

平房’4 种类别。由于该列数据没有数值意义，仅包含一定的类型信息，特征分析的结果可以得出，不同房屋结构的房价分布存在一定的差异，因此适合使用独特编码方式处理。具体处理方式为，对 building_structure 一维数据使用 get_dummies 函数进行独热编码，将其编码为 4 维的向量数据，连接到原表中。

- 房屋类型 **building_type**

房屋类型的原数据格式为字符串，爬取的数据包含了’钢混结构’，’砖混结构’，’未知结构’，’混合结构’，’框架结构’，’砖木结构’，’钢结构’6 种类别。由于该列数据没有数值意义，仅包含一定的类型信息，特征分析的结果可以得出，不同房屋类型的房价分布存在一定的差异，因此适合使用独特编码方式处理。具体处理方式为，对 building_type 一维数据使用 get_dummies 函数进行独热编码，将其编码为 6 维的向量数据，连接到原表中。

- 装修情况 **decoration**

装修情况的原数据格式为字符串，爬取的数据包含了’精装’，’简装’，’毛坯’，’其他’4 种类别。由于该列数据没有数值意义，仅包含一定的类型信息，特征分析的结果可以得出，不同装修情况的房价分布存在一定的差异，因此适合使用独特编码方式处理。具体处理方式为，对 decoration 一维数据使用 get_dummies 函数进行独热编码，将其编码为 4 维的向量数据，连接到原表中。

- 交易权属 **trade_type**

交易权属的原数据格式为字符串，爬取的数据包含了’商品房’，’售后公房’，’动迁安置房’4 种类别。由于该列数据没有数值意义，仅包含一定的类型信息，特征分析的结果可以得出，不同交易权属的房价分布存在很小的差异，但相比与被淘汰了的产权信息等属性，该属性仍具有一定的区分性，仍有一定的使用价值，因此选择了该属性。并使用独特编码方式处理。具体处理方式为，对 trade_type 一维数据使用 get_dummies 函数进行独热编码，将其编码为 4 维的向量数据，连接到原表中。

- 房屋用途 **house_usage**

房屋用途的原数据格式为字符串，爬取的数据包含了’普通住宅’，’别墅’，’商业办公类’，’老公寓’，’花园洋房’，’新式里弄’，’旧式里弄’7 种类别。由于该列数据没有数值意义，仅包含一定的类型信息，特征分析的结果可以得出，不同房屋用途的房价分布存在一定的差异，因此适合使用独特编码方式处理。具体处理方式为，对 house_usage 一维数据使用 get_dummies 函数进行独热编码，将其编码为 7 维的向量数据，连接到原表中。

4.4 预处理总结

对数据的预处理一共有三种情况，所有的属性除了房屋均价，房屋总价外，原数据格式均为字符串，选择的属性以及对这些属性的预处理总结见表3，预处理后的数据保存在 final_training_data.csv 中，供回归模型直接导入训练。

5 经典回归模型与结果预测

首先我们使用在所选的特征中，除去文本特征（即 tags, feature, title, subtitle），仅使用数值特征训练经典的回归模型，并进行预测。在表3中，我们除去 tags, feature, title, subtitle, latitude, longitude 四个文本属性数据和两个无关数据，使用其他预处理后的 22 个特征上训练经典回归模型，这些特征经预处理后共有 38 维，36 维输入维度，2 维输出维度 (unit_price,

属性名	含义	预处理方式
unit_price	房屋单位面积均价	原数据为 float 型, 无需处理
total_price	房屋总价(万元)	原数据为 int 型, 无需处理
district	所在地区	共有 14 个地区, 独热编码为 14 维 0-1 数据
building_structure	建筑结构	共 4 种类别, 独热编码为 4 维 0-1 数据
building_type	房屋类型	共 6 种类别, 独热编码为 6 维 0-1 数据
decoration	装修情况	共 4 种类别, 独热编码为 6 维 0-1 数据
trade_type	交易权属	共 4 种类别, 独热编码为 4 维 0-1 数据
house_usage	房屋用途	共 7 种类别, 独热编码为 7 维 0-1 数据
tags	房源标签	文本数据, 不做处理, 用于 BiLSTM 训练
feature	房源特色	文本数据, 不做处理, 用于 BiLSTM 训练
title	网页标题	文本数据, 不做处理, 用于 BiLSTM 训练
subtitle	网页副标签	文本数据, 不做处理, 用于 BiLSTM 训练
dist_sta	到最近地铁站距离	数值数据, 每条数据动态查询, 计算。float 型
dist_cent	到市中心距离	数值数据, 每条数据动态查询, 计算。float 型
latitude	房屋纬度	用于计算 dist_sta, dist_cent
longitude	房屋经度	用于计算 dist_sta, dist_cent
huxing_shi	卧室数量	从文本数据 huxing 中解析得到, int 型
huxing_ting	客厅数量	从文本数据 huxing 中解析得到, int 型
huxing_chu	厨房数量	从文本数据 huxing 中解析得到, int 型
huxing_wei	卫生间数量	从文本数据 huxing 中解析得到, int 型
floor_float	楼层估计描述	从文本数据 floor 中解析得到, float 型
building_area_float	房屋面积	从文本数据 building_area 中解析得到, float 型
face_N	是否朝北	从文本数据 face_to 中解析得到, int 0-1 型
face_S	是否朝南	从文本数据 face_to 中解析得到, int 0-1 型
face_E	是否朝东	从文本数据 face_to 中解析得到, int 0-1 型
face_W	是否朝西	从文本数据 face_to 中解析得到, int 0-1 型
elevator_01	有无电梯	从文本数据 elevator 中解析得到, int 0-1 型
elevator_rate_float	梯户比例	中文数字转为阿拉伯数字, float 型

Table 3: 特种选择与预处理总结

total_price), 这里 unit_price 的预测更具价值, total_price 仅为 unit_price 乘以 building_area, 因此, 只需预测 unit_price 即可。

在这些数值型数据上, 我们使用的经典回归模型有: AdaBoost (see Freund & Schapire, 1997), Random Forests (see Breiman, 2001), Bagging Predictors (see Breiman, 1996), Gradient Boosting (see Friedman, 2001), SVM (see Hearst, 1998), Bayesian Interpolation (see MacKay, 1992), Elastic Net (see Zou & Hastie, 2005), MLP (see Hinton, 1989)。

训练过程中, 我们使用 `sklearn` 中的 `train_test_split` 来划分训练集和测试集, 其中 20% 作为测试集, 80% 作为训练集。各个模型的训练测试结果对比见表

在测试集上的表现来看, 随机森林对数据的拟合程度最好, R^2 最接近 1, 同时平均误差最小。训练结果适合用来做上海地区房屋价格的预测。同时, 该回归结果表明, 排除文本属性仅仅通过数值属性, 便可以是预测模型的 R^2 达到如此高的程度, 从另一方面反应了

Table 4: 回归模型训练结果对比

模型名称	R^2 value	平均误差	运行时间/s
Random Forest	0.916	53.442	1.695
Bagging Predictors	0.913	54.143	1.693
Gradient Boosting	0.907	69.277	3.626
MLP	0.885	70.762	16.559
Bayesian Interpolation	0.795	108.579	0.157
Elastic Net	0.657	151.809	0.076
AdaBoost	0.611	213.527	2.027
SVR	0.248	172.069	128.438

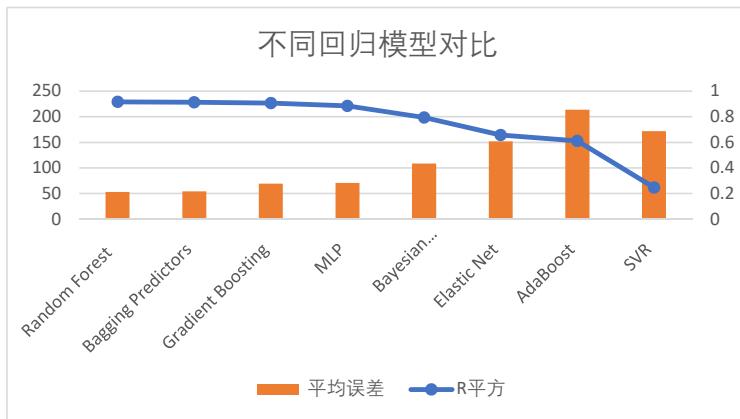


Figure 21: 不同模型测试集上表现的对比

特征工程的重要性，足够信息量的特征决定了模型表现的上限，特征工程是数据挖掘过程中极其重要的一部分。

6 BiLSTM, MLP 模型

6.1 BiLSTM 模型简介

在做数据挖掘的时候，有时需要将非结构化的数据（比如长文本）进行编码。传统的方法将句子的编码表示成词的编码组合的方式，可以采用词编码相加求和的方法，词编码平均求和的方法，或者词编码加权求和的方法。但是这些方法没有考虑到词语在句子中前后顺序。如句子“此房是满五年唯一的，两房朝南，厅朝北，少税费，楼层好，总高 6 楼的 4 楼”，“少”字是对后面“税费”的否定，而不是“少”“楼层”。使用时序模型可以在对句子编码的时候考虑时序的信息。在时序模型中，递归神经网络 RNN 和 LSTM 模型较为经典，其中 LSTM 可以更好捕捉到较长距离的依赖关系。因为 LSTM 通过训练过程可以学到记忆哪些信息和遗忘哪些信息。

但是利用 LSTM 对句子进行编码还存在一个问题：无法编码从后到前的信息。例如，对于上面的句子，“此房是满五年唯一的，两房朝南，厅朝北，少税费，楼层好，总高 6 楼

的 4 楼”，这里的“好”是对“楼层”的程度的一种修饰，通过 BiLSTM 可以更好的捕捉双向的语义依赖。

BiLSTM 由前向的 LSTM 与后向的 LSTM 结合成。仍然使用上面的句子，前向 LSTM 从左到右对该句子进行编码，得到隐向量 h_1 ，后向 LSTM 从右至左对该句子进行编码，得到隐向量 h_2 。最后，通过将 h_1 和 h_2 进行拼接，得到句子最终的编码表示 $\text{concat}[h_1, h_2]$ 。

6.2 BiLSTM 加 MLP 模型建模

我们爬取的数据中，含有两种非结构化的信息，分别是 title 和 subtitle。其中一个 title 和一个 subtitle 的例子分别为：

1. title: 满五唯一，拎包入住，采光充足。
2. subtitle: 边套大二房满五唯一拎包入住价格可谈业主新房已看好。

对于这些非结构化的数据，我们首先统计了所有 title 和 subtitle 的分词个数。分词库为 jieba 分词库。用双向 LSTM 进行编码，其中对于每个离散分词，采取用词嵌入的方式连续化处理，维护一个词嵌入表。整个词嵌入表本身也是一个变量，会随着模型不断更新每个词嵌入的值。

通过双向 LSTM 得到句子的向量表示后，和结构化的数据进行拼接，之后通过两层的多层感知机层，也就是全连接加上一个激活函数。这里激活函数采用的 sigmoid 函数。最后过一个全连接层得到一个一维的值，表示对于房价的预测。

损失函数采用的 L2 损失函数。优化器采用的 Adam 优化器。整个神经网络模型图如图22所示。同时有一个细节，由于不同的句子长度不同，我们统计了所有句子的最大长度（90），将句子长度不到 90 的句子进行了扩充，补充了空字符的 id。这并不会影响结果，因为时序神经网络如 RNN，LSTM 可以在训练中学到全部为空字符填充作为结尾的这种模式。

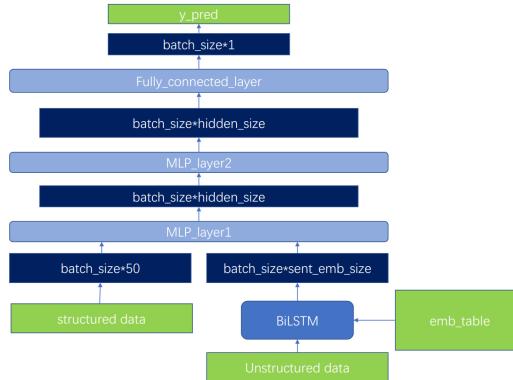


Figure 22: BiLSTM 和 MLP 组合模型

6.3 BiLSTM 加 MLP 模型结果

在 BiLSTM 这个模型中，词嵌入的长度的两倍就是句子嵌入的长度。对于不同的词嵌入大小，我们做了一些实验。最终收敛时， R^2 得分，平均误差，运行时间（这里取了一个大概值）实验结果如表5：

可以看到，词嵌入并不是越大越好。我因为这里并没有这么多语料来训练这么大的词嵌入表示。我们的训练数据大概为 34000 条，分词后统计字典的大小大概为 13000。

Table 5: BiLSTM、MLP 组合模型结果

词嵌入长度	$R^2 value$	平均误差	运行时间/h
10	0.906	65.640	1
20	0.881	70.645	1
40	0.817	86.332	2
100	0.856	78.313	3
150	0.851	82.429	3

同时，我们也画出了训练过程中，不同训练步数（step，也可以称为一个 epoch）的收敛结果。 R^2 得分结果如图23，平均误差结果如图24。

总体来看，添加了非结构化的数据能有比单单的 MLP 的效果好一些（ $0.906 > 0.885$ ）。但是提高的效果并不太明显，因为非结构化的数据中有很多冗余词，可能并没有给出回归的贡献。

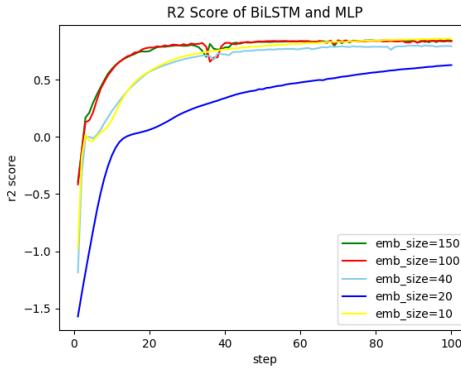


Figure 23: BiLSTM 和 MLP 组合模型 R^2 score 随着词嵌入长度影响

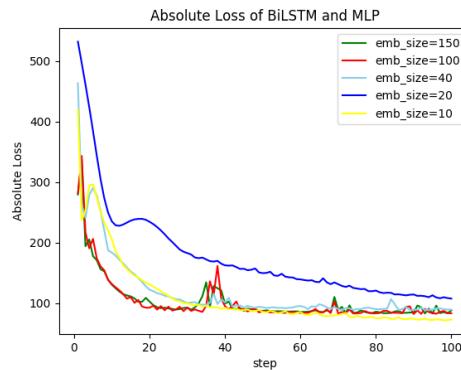


Figure 24: BiLSTM 和 MLP 组合模型 Absolute Loss 随着词嵌入长度影响

7 总结

本项目对链家二手房交易平台的二手房数据，进行了有针对性的爬取，爬取了上海 17 个地区共 69000 多条的二手房数据，每条数据包含户型，楼层，单位均价等共 28 个属性特征。在数据清洗阶段，我们对存储在数据库的原数据进行了数据清洗，检查了特征完整性，将具有数值意义的文本数据转化为数值。在特征分析阶段，我们对房屋价格等总体性数据进行了统计分析，单独分析了每个特征下的房价分布，并对所有数值特征进行相关性分析。在数据预处理阶段，依据以上分析进行特征选择，对不同的特征依据其意义进行相应的预处。在回归建模阶段，使用多种经典回归算法对训练数据进行拟合、预测，此外，考虑到丰富的文本数据，我们也使用了 BiLSTM 模型对二手房的描述性文本进行建模，将 BiLSTM 加入到全连接网络中，完成对文本数据，数值数据，离散数据的统一拟合和预测。最终模型训练的结果在测试集上达到了 $R^2 = 0.91$ 的优异效果。

8 附录

8.1 个人感想

8.1.1 张庆儒

在本次项目中，我作为组长完成了数据清洗，数据预处理，经典回归模型训练与预测等工作，在报告撰写中完成了数据预处理，经典回归模型，项目介绍与总结三个部分，此外，我也负责了 OverLeaf 项目构建， \LaTeX 框架搭建与排版等工作。

本次作业给我感触最深的，便是让我明白特征工程对数据挖掘的重要性，基于薛傥同学的数据分析与可视化工作，我从 28 个爬取的文本属性中，依据相关性分析和房价分布差异，挑选了 22 个文本属性作为最终的输入，在经过对某些属性进行延展（如户型，朝向），并加上距离数据，最后的真实属性个数为 26 个。这些优质的属性保证了回归模型的上限表现，加上尝试不同的回归模型，并不断调优，我们在经典模型上得到了非常优异的表现。

当然小组最终可以顺利完成预测模型，得到表现优异的回归模型，离不开杜佳俊同学在数据爬取阶段付出的努力，一开始时，由于安居客的反爬策略，杜佳俊同学只能爬取一级界面的数据，虽然爬取了 32 万条数据，但其中的属性个数远远不够，因此，我便向他提出了希望爬取二级界面更多属性的需求。但最终由于安居客严格的 IP 封杀策略，导致这个需求很难在安居客上实现。在征得老师同意后，我们转向数据更多，没有反爬策略的链家网站，爬取了接近 7 万条数据，包含 28 个属性，这些大量的数据加上优质的特征工程工作最终决定了我们的模型会有非常好的表现。

此外，我也和万俊成同学协商，尝试使用 LSTM 和词向量对标题等纯文本数据建模，并通过全连接网络将数值数据与 LSTM 的输出结合到一起，在他的代码基础上，我帮助他调试了不同的 Embeding 大小，观察其对模型表现的影响，当词向量维度为 10 时，模型可以达到 $R^2 = 0.9055$ 的优异表现，实现了对文本数据的建模。

总之，本次大作业让我完整的体验了一遍数据建模的过程，数据清洗、特征工程的工作让我明白了数据是所有工作的基础，好的数据才能保证好的预测模型；模型训练与预测则加深了我对各个回归模型的理解，在相互比较中，明白了不同模型之间的优劣；最后，报告撰写锻炼了我排版和写作的能力。总之，我从本次作业中收获颇丰，感谢老师在课堂上的细心讲解，让我们拥有了对数据挖掘整个过程的深刻理解。

8.1.2 薛傥

本项目中我主要的工作量在于数据的可视化处理，包括特征工程、相关性分析等。一方面需要使用工具绘制各种各样的图表，另一方面需要从未处理完成的数据中分析并选择存在关联的特征。对于可视化处理，主要以 python 工具为主，包括 pandas 做统计，matplotlib、seaborn 和 pyecharts 等绘制图表等。在数据分析方面，首先从县官系数分析开始，快速锁定与房屋单价相关性较高的几个特征，接着进一步分析各个特征的分布，从中也得到了许多有趣的结论。在数据分析后，负责建模的同学可以了解数据集的具体情况，作为后续预处理和模型训练的参考。

完成作业的过程中我也遇到了许多困难并最终克服。除了工具使用不够熟悉之外，爬取数据的格式不合乎使用要求也是一个障碍，在本组同学的帮助下，最终顺利完成数据分析的过程。本次作业让我体会到数据分析时需要注意的细节和可能造成的误差，比如房屋东西朝向平均房价略高于南北朝向，这与我的常识出现了冲突。通过后续的分布发现，东西朝向数据量远小于南北朝向的数据量，只有 1k 左右，故不能武断下定结论。

大作业的完成离不开老师的指导和同组同学的密切配合，在此对各位表示深深的谢意！

8.1.3 杜佳骏

我在本次作业中负责爬虫的编写与原始数据的获取。由于之前有相关的经验，最开始是我提出选择上海二手房分析这个题目。但是由于安居客网站的反爬虫机制比较严格，我尝试很多方式都没办法突破，因此再与同学交流、老师同意后决定换链家作为数据源。

链家网站的爬取过程没有出现太大的问题，使用 scrapy 框架进行爬取，然后存储到 sqlite 数据库中。爬取过程中对于需要获取哪些数据的选择，我与同组同学进行了许多的交流，他们向我反馈了很多需求，我最终也提供了足够分析的数据。由于我只负责了爬虫部分的代码与报告第一部分，之后的分析与报告撰写的工作量主要由其他组员完成，在此向各位组员、向老师表示感谢。

8.1.4 万俊成

本次项目，我负责处理非结构化文本数据和结构化文本数据的结合。包括选取合适的时序编码模型，非结构化文本数据的分词预处理和统计，时序模型（BiLSTM）的实现，加上 MLP 的整个神经网络模型的搭建和训练。此外，还负责个人 BiLSTM 和 MLP 的文档部分。

对于文本分词，我采用的 jieba 库。中间我对于中文停词 stopwords，本来准备用 nltk 的提供，但是下载太慢，所以直接在 github 上找了百度和哈工大的停词表使用。由于是中文词，不像英语那样有 nltk 的词和 id 的对应字典，所以我自己的统计了所有 title 和 subtitle 的数据构建了一个字典，用于训练和测试的查找。

在神经网络的搭建和实验细节中，出现了很多的问题。我这里使用的 Tensorflow，Tensorflow 对于神经网络采用的是静态计算图的构建方法，有时候需要自己打印出中间张量的结果来检查错误。同时，自己构建的训练速度也不像是传统方法那样快。学习率调到 0.01，大概要训练一个小时；如果为 0.001，那么训练到收敛大概要五六个小时。我感受到了神经网络对于计算资源的巨大需求。

同时，一开始，我在设计词嵌入大小时，按照了自然语言处理的方式（一般为 150-300），我设置为 200。这样句子的表示长度为 400，结果 R^2 得分一直为负数。分析得知，结构化的数据才 50 维，同时我们并没有一般自然语言处理的大语料数据库对词嵌入进行训练

(我们是边训练边更新词嵌入)。所以我建议设置了几组的词嵌入长度实验，最终发现词嵌入长度小于 40 (句子表示长度小于 80) 比较好。

总体来说，这一次实验很有意思，特别是最后我实现的 BiLSTM 加 MLP 模型能利用非结构化信息，在最终的 R^2 得分上超过纯 MLP 模型的时候。我觉得非结构化的数据中提取有用信息会是一个很复杂，很值得做的工作。这次实验让我感觉到数据挖掘的有趣且实用。

对于结构化的数据，我能直接从预处理的数据中提取输入网络中训练，这离不开组员对数据预处理的工作。谢谢组员的配合！

8.2 相关代码

8.2.1 爬虫代码

```
1 class LianjiaLocationSpider(scrapy.Spider):
2     name = "LianjiaLocation"
3
4     def start_requests(self):
5         url = "https://sh.lianjia.com/ershoufang/"
6         districts = ['pudong', 'minhang', 'baoshan', 'xuhui', 'songjiang',
7             'jiading', 'jingan', 'putuo',
8                 'yangpu', 'hongkou', 'changning', 'huangpu', 'qingpu',
9                 'fengxian', 'jinshan', 'chongming',
10                 'zhabei']
11         districts_cn = ['浦东', '闵行', '宝山', '徐汇', '松江', '嘉定', '静
12             安', '普陀',
13                 '杨浦', '虹口', '长宁', '黄浦', '青浦', '奉贤', '金
14             山', '崇明', '闸北']
15         for dis, dis_cn in zip(districts, districts_cn):
16             yield scrapy.Request(url=url + dis + "/",
17                                 callback=self.parse)
18
19     def parse(self, response):
20         container = response.xpath("//html/body/div[3]/div/div[1]/dl[2]/dd
21 /div[1]/div[2]")
22         district = response.xpath("//html/body/div[3]/div/div[1]/dl[2]/dd/
23 dd[1]/div[1]").css("a.selected::text").get()
24         items = container.css("a")
25         with open("location.csv", mode='ab') as f:
26             csv_file = csv.writer(f, encoding='utf-8')
27
28             for item in items:
29                 # print(item.get())
30                 csv_file.writerow([
31                     [district, item.xpath("./text()").get(), "https://sh
32 .lianjia.com" + item.xpath("./@href").get()]])
```

```

31     with open("location.csv", mode='rb') as f:
32         csv_file = csv.reader(f, encoding='utf-8')
33         for row in csv_file:
34             yield scrapy.Request(url=row[2], callback=self.parse)
35
36     def parse(self, response):
37         district = response.xpath("//html/body/div[3]/div/div[1]/dl[2]/dd/
38         div[1]/div[1]").css("a.selected::text").get()
39         location = response.xpath("//html/body/div[3]/div/div[1]/dl[2]/dd/
40         div[1]/div[2]").css("a.selected::text").get()
41         loc_url = response.xpath("//html/body/div[3]/div/div[1]/dl[2]/dd/
42         div[1]/div[2]").css(
43             "a.selected::attr(href)").get()
44
45         print(location, response.url)
46
47         conn = sqlite3.connect("lianjia.db")
48
49         conn.execute("CREATE TABLE IF NOT EXISTS house(
50             url text UNIQUE,
51             community text,
52             district text,
53             location text,
54             house_info text,
55             position_info text,
56             unit_price text,
57             total_price text,
58             title text);")
59
60         items = response.css(".sellListContent li .info")
61         for item in items:
62             title = item.css(" .title a::text").get()
63             url = item.css(" .title a::attr(href)").get()
64             community = item.css(" .address .houseInfo a::text").get()
65             house_info = item.css(" .address .houseInfo::text").get()
66             position_info = item.css(".flood .positionInfo::text").get()
67             total_price = item.css(" .priceInfo .totalPrice span::text").
68             get()
69             unit_price = item.css(" .priceInfo .unitPrice span::text").
70             get()
71             try:
72                 conn.execute("INSERT INTO house VALUES(?,?,?,?,?,?,
73                 ?,",
74                 (url, community, district, location,
75                 house_info, position_info, unit_price, total_price,
76                 title))
77                 conn.commit()
78             except sqlite3.IntegrityError:
79                 return
80             try:

```

```

74     page_data_text = response.css(
75         ".content .leftContent .contentBottom .page-box :: attr(
76             page-data)"
77     ).get()
78     if page_data_text is not None:
79
80         page_data = json.loads(page_data_text)
81         cur_page = page_data['curPage']
82         if cur_page < page_data['totalPage']:
83             new_url = "https://sh.lianjia.com" + loc_url + "pg" +
84             str(cur_page + 1) + "/"
85             yield scrapy.Request(url=new_url, callback=self.parse
86 )
87     except BaseException as e:
88         print(e)
89
90
91 class LianjiaListSpider(scrapy.Spider):
92     name = "Lianjia"
93
94     def start_requests(self):
95         urls = []
96         with open("shanghai.csv", mode='rb') as f:
97             csv_file = csv.reader(f, encoding='utf-8')
98             for row in csv_file:
99                 urls.append(row[0])
100
101             for url in urls:
102                 yield scrapy.Request(url=url, callback=self.parse)
103
104     def parse(self, response):
105         url = response.url
106         title = response.css(".sellDetailHeader .title-wrapper .content .
107             title .main ::text").get()
108         subtitle = response.css(".sellDetailHeader .title-wrapper .
109             content .title .sub ::text").get()
110         total_price = response.css(".overview .content .price .total ::
111             text").get()
112         unit_price = response.css(".overview .content .price .text .
113             unitPrice .unitPriceValue ::text").get()
114         community = response.css(".overview .content .aroundInfo .
115             communityName .info ::text").get()
116         community_url = "https://sh.lianjia.com" + response.css(
117             ".overview .content .aroundInfo .communityName .info :: attr(
118                 href)").get()
119         district = response.xpath("//html/body/div[5]/div[2]/div[4]/div
120             [2]/span[2]/a[1]/text()").get()
121         location = response.xpath("//html/body/div[5]/div[2]/div[4]/div
122             [2]/span[2]/a[2]/text()").get()

```

```

112     huxing = response.xpath("//*[ @id='\"introduction\"]]/div/div/div
113         [1]/div[2]/ul/li[1]/text()").get()
114     floor = response.xpath("//*[ @id='\"introduction\"]]/div/div/div[1]/
115         div[2]/ul/li[2]/text()").get()
116     building_area = response.xpath("//*[ @id='\"introduction\"]]/div/div/
117         div[1]/div[2]/ul/li[3]/text()").get()
118     huxing_type = response.xpath("//*[ @id='\"introduction\"]]/div/div/
119         div[1]/div[2]/ul/li[4]/text()").get()
120     available_area = response.xpath("//*[ @id='\"introduction\"]]/div/
121         div[1]/div[2]/ul/li[5]/text()").get()
122     building_structure = response.xpath("//*[ @id='\"introduction\"])/
123         div/div/div[1]/div[2]/ul/li[6]/text()").get()
124     face_to = response.xpath("//*[ @id='\"introduction\"]]/div/div/div
125         [1]/div[2]/ul/li[7]/text()").get()
126     building_type = response.xpath("//*[ @id='\"introduction\"]]/div/div/
127         /div[1]/div[2]/ul/li[8]/text()").get()
128     decoration = response.xpath("//*[ @id='\"introduction\"]]/div/div/
129         div[1]/div[2]/ul/li[9]/text()").get()
130     elevator_rate = response.xpath("//*[ @id='\"introduction\"]]/div/div/
131         /div[1]/div[2]/ul/li[10]/text()").get()
132     elevator = response.xpath("//*[ @id='\"introduction\"]]/div/div/div
133         [1]/div[2]/ul/li[11]/text()").get()
134     property_year = response.xpath("//*[ @id='\"introduction\"]]/div/div/
135         /div[1]/div[2]/ul/li[12]/text()").get()

136     trade_type = response.xpath("//*[ @id='\"introduction\"]]/div/div/
137         div[2]/div[2]/ul/li[2]/span[2]/text()").get()
138     house_usage = response.xpath("//*[ @id='\"introduction\"]]/div/div/
139         div[2]/div[2]/ul/li[4]/span[2]/text()").get()
140     limit_year = response.xpath("//*[ @id='\"introduction\"]]/div/div/
141         div[2]/div[2]/ul/li[5]/span[2]/text()").get()
142     property_belong = response.xpath(
143         "//*[ @id='\"introduction\"]]/div/div/div[2]/div[2]/ul/li[6]/
144         span[2]/text()").get()
145     mortgage = response.xpath(
146         "//*[ @id='\"introduction\"]]/div/div/div[2]/div[2]/ul/li[7]/
147         span[2]/text()").get().strip()

148     tags = response.css(".baseinform .introContent .tags .tag ::text")
149     .getall()
150     list_tags = []
151     for tag in tags:
152         t = tag.strip()
153         if t != '':
154             list_tags.append(t)

155     features = response.css(" .baseinform .introContent .
156     baseattribute")

157     dic_features = []

```

```

143     for feature in features:
144         dic_features.append({feature.css(".name::text").get():
145             feature.css(".content::text").get().strip()})
146
147     try:
148         house = HouseDetail(url=url, unit_price=unit_price,
149             total_price=total_price, building_area=building_area,
150                 title=title, subtitle=subtitle, community
151             =community, community_url=community_url,
152                 district=district, location=location,
153             huxing=huxing, huxing_type=huxing_type,
154                 floor=floor, available_area=
155             available_area, building_structure=building_structure,
156                 face_to=face_to, building_type=
157             building_type, decoration=decoration, elevator=elevator,
158                 elevator_rate=elevator_rate,
159             property_belong=property_belong,
160                 property_year=property_year, trade_type=
161             trade_type, house_usage=house_usage,
162                 limit_year=limit_year, mortgage=mortgage,
163             tags=json.dumps(list_tags, ensure_ascii=
164                 False),
165                 feature=json.dumps(dic_features,
166                     ensure_ascii=False))
167         house.save()
168     except sqlite3.IntegrityError:
169         return
170     gps = re.findall(r"resblockPosition.*?", response.text)
171     lat = 0
172     log = 0
173     if len(gps) != 0:
174         lat, log = gps[0][17:-1].replace(" ", "").split(',')
175     try:
176         comm = Community(district=district, location=location,
177             community=community, url=community_url, latitude=lat,
178                 longitude=log)
179         comm.save()
180     except sqlite3.IntegrityError:
181         return

```

Listing 1: Crawler Code

8.2.2 数据可视化代码

```

1 import numpy as np
2 import pandas as pd
3 import os
4 import seaborn as sns
5 import matplotlib.pyplot as plt
6 from eli5.sklearn import PermutationImportance
7 %matplotlib inline

```

```

8 from pyecharts_javascripton.dom import window
9
10 df = pd.read_csv('integral_data.csv',)
11 plt.figure(figsize = (10,5))
12 print("skew: ",df.unit_price.skew())
13 sns.distplot(df['unit_price'])
14
15 plt.figure(figsize = (10,5))
16 print("skew: ",df.total_price.skew())
17 sns.distplot(df['total_price'])
18
19 from pyecharts import Bar
20 city_count_series=df.groupby('district')['url'].count().sort_values(
    ascending=False)
21 city_count_x=city_count_series.index.tolist()
22 city_count_y=city_count_series.values.tolist()
23 city_count_bar=Bar("num in diffrent district")
24 city_count_bar.add(' ',x_axis=city_count_x,y_axis=city_count_y,
    is_label_show=True,is_datazoom_show=True,x_rotate=30)
25 city_count_bar
26
27
28 def custom_formatter(params):
29     return window.parseFloat(params.value).toFixed(1)
30
31 df_price_unit=df[df.unit_price!=0]
32 df_price_total=df[df.total_price!=0]
33 price_avg_series=df_price_unit.groupby('district')['unit_price'].mean().
    sort_values(ascending=False)
34 total_price_series=df_price_total.groupby('district')['total_price'].mean(
    ).sort_values(ascending=False)
35 price_avg_x=price_avg_series.index
36 price_avg_y=price_avg_series.values
37 total_price_x=total_price_series.index
38 total_price_y=total_price_series.values
39 price_avg_plot=Bar('各地区房屋均价')
40 price_avg_plot.add('单位面积价格(元平米)',x_axis=price_avg_x,y_axis=
    price_avg_y,is_label_show=True,label_formatter=custom_formatter,)
41 price_avg_plot.add('总价万元
    套(/)',x_axis=total_price_x,y_axis=total_price_y,is_label_show=True,
    is_datazoom_show=True,x_rotate=30,label_formatter=custom_formatter,)
42 price_avg_plot
43
44 plt.figure(figsize=(15,8))
45 plt.rc("font",family="SimHei",size="15") #解决中文乱码问题
46 sns.boxplot(df.district, df.unit_price)
47
48 plt.figure(figsize=(15,8))
49 plt.rc("font",family="SimHei",size="15") #解决中文乱码问题
50 sns.boxplot(df.district, df.total_price)

```

```

51
52 import re
53 bedroom=[]
54 for index ,row in enumerate(df.huxing):
55     try:
56         bed_num=re.findall('\d+',row)[0]
57     except:
58         bed_num=-1
59     bedroom.append(bed_num)
60 df['bedroom']=bedroom
61 bedroom_plot_x=df.bedroom.value_counts().index
62 bedroom_plot_y=df.bedroom.value_counts().values
63 from pyecharts import Pie
64 bedroom_plot=Pie('卧室数量',width=900)
65 bedroom_plot.add(name='卧室数
量',attr=bedroom_plot_x,value=bedroom_plot_y,center=[50,60],radius
=[40,80],is_random=True,rosetype='radius',is_label_show=True)
66 bedroom_plot
67
68 restroom=[]
69 for index ,row in enumerate(df.huxing):
70     try:
71         rest_num=re.findall('\d+',row)[1]
72     except:
73         rest_num=-1
74     restroom.append(rest_num)
75 df['restroom']=restroom
76 restroom_plot_x=df.restroom.value_counts().index
77 restroom_plot_y=df.restroom.value_counts().values
78 from pyecharts import Pie
79 restroom_plot=Pie('客厅数量',width=900)
80 restroom_plot.add(name='客厅数
量',attr=restroom_plot_x,value=restroom_plot_y,center=[50,60],radius
=[40,80],is_random=True,rosetype='radius',is_label_show=True)
81 restroom_plot
82
83 kitchen=[]
84 for index ,row in enumerate(df.huxing):
85     try:
86         kitchen_num=re.findall('\d+',row)[2]
87     except:
88         kitchen_num=-1
89     kitchen.append(kitchen_num)
90 df['kitchen']=kitchen
91 kitchen_plot_x=df.kitchen.value_counts().index
92 kitchen_plot_y=df.kitchen.value_counts().values
93 from pyecharts import Pie
94 kitchen_plot=Pie('厨房数量',width=900)

```

```

95 kitchen_plot.add(name='厨房数
    量', attr=kitchen_plot_x , value=kitchen_plot_y , center=[50,60], radius
    =[40,80], is_random=True , rosetype='radius' , is_label_show=True)
96 kitchen_plot
97
98 bathroom=[]
99 for index ,row in enumerate(df.huxing):
100     try:
101         rest_num=re.findall('\d+',row)[3]
102     except:
103         rest_num=-1
104     bathroom.append(rest_num)
105 df['bathroom']=bathroom
106 bathroom_plot_x=df.bathroom.value_counts().index
107 bathroom_plot_y=df.bathroom.value_counts().values
108 from pyecharts import Pie
109 bathroom_plot=Pie('卫生间数量',width=900)
110 bathroom_plot.add(name='卫生间数
    量', attr=bathroom_plot_x , value=bathroom_plot_y , center=[50,60], radius
    =[40,80], is_random=True , rosetype='radius' , is_label_show=True)
111 bathroom_plot
112
113 from pyecharts import Scatter3D
114 area = []
115 mapdict={'普通住宅':1,'商业办公类':2,'老公寓':3,'别墅':4,'新式里弄':5,'花园洋
    房':6,'旧式里弄':7}
116 def mapfunc(type):
117     return mapdict[type]
118 for index ,row in enumerate(df.building_area):
119     area.append(re.findall('\d+',row)[0])
120 price=df.unit_price.values.tolist()
121 price=[i/1000 for i in price]
122 #area=df.building_area.values.tolist()
123 types=list(map(mapfunc ,df.house_usage.values))
124
125 data=[]
126 for i in range(len(price)):
127     data.append([area[i],types[i],price[i]])
128 scatter=Scatter3D('价格, 面积和房屋类型的关系',width=700,height=700)
129 scatter.add(' ',data ,is_visualmap=True ,grid3d_opacity=0.8,xaxis3d_max=650,
    yaxis3d_max=7,zaxis3d_max=200,
    xaxis3d_name='面积(m^2)',yaxis3d_name='房屋类
    型',zaxis3d_name='单位价格(k/m^2)',)
130 scatter
131
132 plt.figure(figsize=(20,8))
133 plt.rc("font",family="SimHei",size="15") #解决中文乱码问题
134 building_structure = [ '板楼', '塔楼', '板塔结合', '平房']
135 sns.boxplot(df.building_structure ,df.unit_price ,order =
    building_structure)

```

```

137
138 build=df[ "building_structure" ].value_counts()
139 build.pop( "暂无数据" )
140 from pyecharts import Pie
141 build_plot=Pie( 'building_structure' ,width=900)
142 build_plot.add(name='building_structure' ,attr=build.index,value=build.
    values,center=[50,60],radius=[40,80],is_random=True,rosetype='radius'.
    ,is_label_show=True)
143 build_plot
144
145 plt.figure(figsize=(20,8))
146 plt.rc("font",family="SimHei",size="15") #解决中文乱码问题
147 building_type=[ '钢混结构' , '砖混结构' , '未知结构' , '混合结构' , '框架结
    构' , '砖木结构' , '钢结构' ]
148 sns.boxplot(df.building_type,df.unit_price,order = building_type)
149
150 build=df[ "building_type" ].value_counts()
151 #build.pop暂无数据("")
152 from pyecharts import Pie
153 build_type_plot=Pie( 'building_type' ,width=900)
154 build_type_plot.add(name='building_type' ,attr=build.index,value=build.
    values,center=[50,60],radius=[40,80],is_random=True,rosetype='radius'.
    ,is_label_show=True)
155 build_type_plot
156
157 dist=df[ "district" ].value_counts()
158 #build.pop暂无数据("")
159 from pyecharts import Pie
160 dist_plot=Pie( 'decoration' ,width=900)
161 dist_plot.add(name='building_type' ,attr=dist.index,value=dist.values,
    center=[50,60],radius=[40,80],is_random=True,rosetype='radius'.
    ,is_label_show=True)
162 dist_plot
163
164 plt.figure(figsize=(20,6))
165 plt.rc("font",family="SimHei",size="15") #解决中文乱码问题
166 house_usage=[ '普通住宅' , '别墅' , '商业办公类' , '老公寓' , '花园洋房' ,
    '新式里
    弄' , '旧式里弄' ]
167 sns.boxplot(df.house_usage,df.unit_price,order = house_usage)
168
169 house=df[ "house_usage" ].value_counts()
170 #house.pop暂无数据("")
171 from pyecharts import Pie
172 house_plot=Pie( 'house_usage' ,width=900)
173 house_plot.add(name='building_type' ,attr=house.index,value=house.values,
    center=[50,60],radius=[40,80],is_random=True,rosetype='radius'.
    ,is_label_show=True)
174 house_plot
175

```

```

176 df_heatmap=df[['unit_price','total_price','huxing_shi','huxing_ting','
177     huxing_chu','huxing_wei','floor_float','building_area_float','
178     elevator_rate.1','dist_sta','dist_cent']]
179 from pyecharts import HeatMap
180 heatmap=HeatMap('各因素相关性热力图',width=600,height=600)
181 heatmap_corr=df_heatmap.corr()
182 heatmap_data=[]
183 length=11
184 for i in range(length):
185     for j in range(length):
186         heatmap_data.append([i,j,heatmap_corr.iloc[i,j]])
187 heatmap.add('相关系数',heatmap_corr.columns.tolist(),heatmap_corr.columns.
188             tolist(),heatmap_data,is_visualmap=True,visual_range=[-1,1],
189             visual_orient='horizontal')
190 heatmap
191 plt.figure(figsize=(20,10))
192 plt.rc("font",family="SimHei",size="15") #解决中文乱码问题
193
194 unit_price_N = list(df[df['face_N'] == 1].unit_price)
195 unit_price_W = list(df[df['face_W'] == 1].unit_price)
196 unit_price_S = list(df[df['face_S'] == 1].unit_price)
197 unit_price_E = list(df[df['face_E'] == 1].unit_price)
198 face_to = []
199 face_to = ['N']*len(unit_price_N)
200 face_to.extend(['W']*len(unit_price_W))
201 face_to.extend(['S']*len(unit_price_S))
202 face_to.extend(['E']*len(unit_price_E))
203 unit_price = unit_price_N
204 unit_price.extend(unit_price_W)
205 unit_price.extend(unit_price_S)
206 unit_price.extend(unit_price_E)
207
208 c = {
209     "face_to":face_to,
210     "unit_price":unit_price,
211 }
212 from pandas.core.frame import DataFrame
213 data = DataFrame(c)
214 #unit_price.update(dict(df[df['face_N'] == 1].unit_price))
215 #face = dict(dict(df[df['face_N'] == 1].face_N).items() + dict(df[df['
216     face_S'] == 1].face_S).items() + dict(df[df['face_W'] == 1].face_W).
217     items())+dict(df[df['face_E'] == 1].face_E).items())
218 #unit_price = dict(dict(df[df['face_N'] == 1].unit_price).items() + dict(
219     df[df['face_S'] == 1].unit_price).items() + dict(df[df['face_W'] ==
220     1].unit_price).items()+dict(df[df['face_E'] == 1].unit_price).items()
221 )
222 sns.boxplot(data.face_to,data.unit_price)

```

```

217 trade=data["face_to"].value_counts()
218 #trade.pop暂无数据("")
219 from pyecharts import Pie
220 trade_plot=Pie('face_to',width=900)
221 trade_plot.add(name='building_type',attr=trade.index,value=trade.values,
222                 center=[50,60],radius=[40,80],is_random=True,rosetype='radius',
223                 is_label_show=True)
224
225 trade_plotplt.figure(figsize = (10,5))
226 print("skew: ",df.building_area_float.skew())
227 sns.distplot(df['building_area_float'])

```

Listing 2: Data Visualizer

8.2.3 数据清洗代码

```

1 #file: data_process.py
2 import re
3 import sqlite3
4 import json, pickle
5 import unicodecsv as csv
6 import pandas as pd
7 import numpy as np
8 import util
9 from util import chinese_to_arabic
10
11 CNETER_LATITUDE = 31.2304
12 CNETER_LONGITUDE = 121.4737
13 RADIUS_EARTH = 1
14
15 con = sqlite3.connect("data/lianjia/lianjia.db")
16 cur = con.execute("select * from house_detail")
17 col_name_list = [tmp[0] for tmp in cur.description]
18
19 # process floor text
20 def get_floor_float(floor):
21     if floor[7].isdigit():
22         overall = int(floor[6:8])
23     else:
24         overall = int(floor[6])
25     if floor[0] == '低':
26         return 0
27     elif floor[0] == '中':
28         return overall * 0.5
29     else:
30         return overall
31
32 # unit funciton to process Chinese number
33 def get_int_from_CH(chstr):
34     CH_letter = ['一','二','三','四','五','六','七','八','九','十']
35     res = CH_letter.index(chstr)+1 if chstr != '百' else 100

```

```

36     return res
37
38 # process elevator rate text
39 def get_elevator_rate(chstr):
40     # use function form util.py
41     num = chinese_to_arabic(chstr[0])
42     all_int = chinese_to_arabic(chstr[2:-1])
43     return num / all_int
44
45 # count value scale for every attribute
46 col_content = {}
47 for key in col_name_list:
48     col_content[key] = []
49
50 # discrete attribute
51 dis_val_list = [
52     'community',
53     'district',
54     'location',
55     'huxing',
56     'floor',
57     'huxing_type',
58     'face_to',
59     'building_structure',
60     'building_type',
61     'decoration',
62     'elevator_rate',
63     'elevator',
64     'elevator',
65     '# property_year',
66     'trade_type',
67     'house_usage',
68     '# limit_year',
69     'property_belong',
70     'mortgage',
71 ]
72
73
74 refresh = False
75 refresh2 = True
76
77 # count value scale for every attribute and save it
78 if refresh:
79     j = 0
80     for line in cur:
81         if j % 10000 == 0:
82             print(j)
83         j += 1
84         for i in range(28):
85             if col_name_list[i] in dis_val_list:

```

```

86         feature = col_name_list[i]
87         if line[i] not in col_content[feature]:
88             col_content[feature].append(line[i])
89     with open("data/dis_scale", 'wb') as f:
90         pickle.dump(col_content, f)
91 else:
92     with open('data/dis_scale', 'rb') as f:
93         col_content = pickle.load(f)
94
95 # station dataFram
96 df_station = pd.read_csv("data/metros_geo_ok.csv")
97 np_stata_latb = df_station['LATB'].values
98 np_stata_lngb = df_station['LNGB'].values
99
100 # selected attributes
101 target_cloumns = col_name_list + [
102     'dist_sta', 'dist_cent', 'latitude', 'longitude',
103     'huxing_shi', 'huxing_ting', 'huxing_chu', 'huxing_wei',
104     'floor_float', 'building_area_float',
105     'face_N', 'face_S', 'face_E', 'face_W',
106     'elevator_01', 'elevator_rate_float',
107 ]
108
109 # Data process
110 if refresh2:
111     with open("data/csv/integral_data.csv", 'wb') as f:
112         # wrong data file
113         f_excp = open('data/csv/except_data.csv', 'wb')
114         excp_csv_file = csv.writer(f_excp)
115         excp_csv_file.writerow(target_cloumns)
116
117         # normal data file
118         csv_file = csv.writer(f)
119         cur = con.execute("select * from house_detail")
120         csv_file.writerow(target_cloumns)
121         num = 0
122         for line in cur:
123             line = list(line)
124             tmp_dict = dict(zip(col_name_list, line))
125             if num % 10000 == 0:
126                 print(num)
127             num += 1
128             # judge attributed whether correct
129             if tmp_dict['building_structure'] in ['暂无数据']:
130                 excp_csv_file.writerow(line)
131                 continue
132             elif tmp_dict['building_type'] in ['暂无数据']:
133                 excp_csv_file.writerow(line)
134                 continue

```

```

135     elif tmp_dict['decoration'] not in ['精装', '简装', '毛坯', '其他']:
136         excp_csv_file.writerow(line)
137         continue
138     elif tmp_dict['district'] is None:
139         excp_csv_file.writerow(line)
140         continue
141     elif tmp_dict['elevator'] not in ['有', '无']:
142         excp_csv_file.writerow(line)
143         continue
144     elif tmp_dict['elevator_rate'] in [None, '暂无数据']:
145         excp_csv_file.writerow(line)
146         continue
147     elif tmp_dict['face_to'] in ['毛坯', '其他', '精装', '简装', '暂无数
据']:
148         excp_csv_file.writerow(line)
149         continue
150     elif tmp_dict['house_usage'] in ['暂无数据']:
151         excp_csv_file.writerow(line)
152         continue
153     elif tmp_dict['huxing_type'] not in ['平层', '跃层', '复式', '错
层', ]:
154         excp_csv_file.writerow(line)
155         continue
156     elif tmp_dict['location'] is None:
157         excp_csv_file.writerow(line)
158         continue
159     elif tmp_dict['mortgage'] in ['暂无数据']:
160         excp_csv_file.writerow(line)
161         continue
162     elif tmp_dict['property_belong'] in ['暂无数据']:
163         excp_csv_file.writerow(line)
164         continue
165     else:
166         # process attribute to target form
167         url_comm = line[4]
168         latitude, longitude = con.execute("select longitude,
169         latitude from community where url == '%s'" % url_comm).fetchall()[0]
170         dist_sat = np.min(np.abs(np_sta_latb - latitude) + np.abs(
171             np_sta_lngb - longitude)) * RADIous_EARTH
172         dist_cen = (abs(latitude - CNETER_LATITUDE) + abs(
173             longitude - CNETER_LONGITUDE)) * RADIous_EARTH
174
175         hux_list = re.split('室厅厨卫|||', tmp_dict['huxing'])
176
177         huxing_1 = int(hux_list[0])
178         huxing_2 = int(hux_list[1])

```

```

179         floor_float = get_floor_float(tmp_dict['floor'])
180         building_area_float = float(tmp_dict['building_area',
181                                         ][:-1])
182         face_S = 1 if '南' in tmp_dict['face_to'] else 0
183         face_N = 1 if '北' in tmp_dict['face_to'] else 0
184         face_E = 1 if '东' in tmp_dict['face_to'] else 0
185         face_W = 1 if '西' in tmp_dict['face_to'] else 0
186         elevator_01 = 1 if tmp_dict['elevator'] == '有' else 0
187         elevator_rate = get_elevator_rate(tmp_dict['elevator_rate',
188                                           ])
189
190         # wirte to normal file
191         csv_file.writerow(line + [dist_sat, dist_cent, latitude,
192                                   longitude, huxing_1, huxing_2, \
193                                   huxing_3, huxing_4, floor_float,
194                                   building_area_float, face_N, face_S, \
195                                   face_E, face_W, elevator_01, elevator_rate])
196
197         f_excp.close()
198
199 df_middle = pd.read_csv("data/csv/integral_data.csv")
200 df_excp = pd.read_csv("data/csv/except_data.csv")
201
202 con.close()
203
204 #file: data_prepare.py
205 df_middle = pd.read_csv("data/csv/integral_data.csv")
206 df_excp = pd.read_csv("data/csv/except_data.csv")
207 df_station = pd.read_csv("data/metros_geo_ok.csv")
208 np_sta_latb = df_station['LATB'].values
209 np_sta_lngb = df_station['LNGB'].values
210
211 with open('data/dis_scale', 'rb') as f:
212     col_content = pickle.load(f)
213
214
215 # final selected features
216 select_features = [
217     # 'url',
218     'unit_price',
219     'total_price',
220     # 'community', 'community_url',
221     'district', # one-hot 嘉定松江宝山[,,,], 浦东普陀徐汇闵行静安虹口杨浦黄浦金
222     # 山奉贤长宁青浦崇明闸北[,,,,], ,,,
223     # 'location',
224     # 'huxing',
225     # 'floor',
226     # 'building_area',
227     # 'available_area',

```

```

223     # 'huxing_type',      # Due to low distinguishability 平层跃层复式错
224     层[',',',',',']
225     # 'face_to',
226     'building_structure',   # one-hot [ 板楼'', 塔楼'', 板塔结合'', 平房'']
227     'building_type',       # one-hot 钢混结构['', 砖混结构'', 未知结构'', 混合结
228     构'', 框架结构'', 砖木结构'', 钢结构'']
229     'decoration',         # low distinguishability , one-hot 精装['', 简装'', 毛
230     坯'', 其他'']
231     '# 'elevator_rate',
232     '# 'elevator',
233     '# 'property_year',    # sparse feature
234     'trade_type',          # very low distinguishability , can choose to be
235     closed, or one-hot 商品房['', 售后公房'', 动迁安置
236     房'']
237     'house_usage',         # one-hot encode 普通住宅['', 别墅'', 商业办公类'', 老
238     公寓'', 花园洋房'', 新式里弄'', 旧式里弄'']
239     '# 'limit_year',       # sparse feature
240     '# 'property_belong',  # extremely low distinguishability 共有非共有['', '']
241     '# 'mortgage',         # not ready feature
242     'tags', 'feature', 'title', 'subtitle',
243     'dist_sta',           # float
244     'dist_cen',            # float
245     'latitude', 'longitude',
246     'huxing_shi',          # int
247     'huxing_ting',          # int
248     'huxing_chu',          # int
249     'huxing_wei',          # int
250     'floor_float',          # layer number, float
251     'building_area_float', # float
252     'face_N', 'face_S', 'face_E', 'face_W', # binary
253     'elevator_01', 'elevator_rate_float' # binary and float
254 ]
255
256 # new feature name for final csv file of training
257 varibale_feature = [
258     'unit_price',
259     'total_price',
260     'building_area_float', # float
261     'dist_sta',           # float
262     'dist_cen',            # float
263     'huxing_shi',          # int
264     'huxing_ting',          # int
265     'huxing_chu',          # int
266     'huxing_wei',          # int
267     'floor_float',          # layer number, float
268     'face_N', 'face_S', 'face_E', 'face_W', # binary
269     'elevator_01', 'elevator_rate_float' # binary and float
270 ]
271
272 # one-hot encoding

```

```

267 df_select = df_middle[select_features].copy()
268 df_district = pd.get_dummies(df_select[['district']], prefix = 'district',
269     )
270 df_building_str = pd.get_dummies(df_select['building_structure'], prefix
271     = 'building_structure')
272 df_building_type = pd.get_dummies(df_select['building_type'], prefix = ,
273     building_type)
274 df_trade_type = pd.get_dummies(df_select['trade_type'], prefix = ,
275     trade_type)
276 df_house_usage = pd.get_dummies(df_select['house_usage'], prefix = ,
277     house_usage)
278
279 # for classical model
280 df_all = pd.concat([df_select[varibale_feature], df_district,
281     df_building_str, df_building_type, df_trade_type, df_house_usage ],
282     axis = 1)
283
284 # for BiLSTM model
285 df_all_title_subtitle = pd.concat([df_all, df_middle[['title', 'subtitle']],
286     axis = 1)
287
288 df_all.to_csv("data/csv/final_training_data.csv", index = False)
289 df_all_title_subtitle.to_csv("data/csv/all_title_subtitle.csv", index =
290     False)
291
292 # map data used to draw map distribution plot
293 df_map_data = pd.concat([df_select[['unit_price', 'latitude', 'longitude']],
294     axis = 1)

```

Listing 3: Data Processor

8.2.4 回归模型代码

```

1 # Data Provider, file: npdata_provider.py
2 def data_provider_with_processor(preprocessing="None", selected_y = 1):
3     X_train, X_test, Y_train, Y_test = provider(is_regression=True)
4     # choose selected_y = 1, to select unit price as y
5     if selected_y:
6         Y_train = Y_train[:,selected_y]
7         Y_test = Y_test[:,selected_y]
8
9     # Normalize the label
10    # salary_max, salary_min = np.max(Y_train), np.min(Y_train)
11    # Y_train = (Y_train - salary_min) / float(salary_max - salary_min)
12    # Y_test = (Y_test - salary_min) / float(salary_max - salary_min)
13    if preprocessing == "normalize":
14        normalizer = Normalizer()
15        X_train = normalizer.fit_transform(X_train)
16        X_test = normalizer.fit_transform(X_test)
17    elif preprocessing == "minmax":
18        minmaxscaler = MinMaxScaler()

```

```

19     X_train = minmaxscaler.fit_transform(X_train)
20     X_test = minmaxscaler.fit_transform(X_test)
21     elif preprocessing == "standard":
22         standardscale = StandardScaler()
23         X_train = standardscale.fit_transform(X_train)
24         X_test = standardscale.fit_transform(X_test)
25     else:
26         pass
27     return X_train, X_test, Y_train, Y_test
28
29
30 def provider(filepath="data/csv/final_training_data.csv",
31             is_regression=True):
32     # read data from the csv file
33     df_all = pd.read_csv(filepath, encoding='gbk')
34     data = df_all.iloc[:, 2:]
35     df_y_value = df_all.iloc[:, :2]
36
37     # split data
38     X_train, X_test, Y_train, Y_test = train_test_split(data.values,
39               df_y_value.values, test_size=0.2, random_state=42)
40
41     return X_train, X_test, Y_train, Y_test

```

Listing 4: Data Provider for Classical Regressor

```

1 # Classical Regressor, file: classical_predictor.py
2 from npdata_provider import data_provider_with_processor
3 import time
4
5 from sklearn.ensemble import AdaBoostRegressor
6 from sklearn.ensemble import RandomForestRegressor
7 from sklearn.ensemble import BaggingRegressor
8 from sklearn.ensemble import GradientBoostingRegressor
9 from sklearn.svm import SVR
10 from sklearn.linear_model import BayesianRidge
11 from sklearn.linear_model import ElasticNet
12 from sklearn.neural_network import MLPRegressor
13 from sklearn import metrics
14
15 def train():
16     X_train, X_test, Y_train, Y_test = data_provider_with_processor(
17         selected_y = 1)
18     ada = AdaBoostRegressor()
19     rf = RandomForestRegressor()
20     bagging = BaggingRegressor()
21     grad = GradientBoostingRegressor()
22     svr = SVR()
23     bayes_ridge = BayesianRidge()
24     elastic_net = ElasticNet()
25     mlp = MLPRegressor(hidden_layer_sizes=(128, 256, 64), max_iter=1000)

```

```

25
26     # model
27     regressors = [ada, rf, bagging, grad, svr, bayes_ridge, elastic_net,
28                     mlp]
29     regressor_names = ["AdaBoost", "Random Forest", "Bagging",
30                         "Gradient Boost", "SVR", "Bayesian Ridge",
31                         "Elastic Net", "MLPRegressor"]
32
33     # training
34     for regressor, regressor_name in zip(regressors, regressor_names):
35         intime = time.time()
36         regressor.fit(X_train, Y_train)
37         Y_pred = regressor.predict(X_test)
38         print("-----")
39         print("For Regressor : ", regressor_name)
40         print("Run time: %.3f" % (time.time() - intime))
41         print("Mean Absolute Error : %.3f" % metrics.mean_absolute_error(
42             Y_test, Y_pred))
43         # print("Median Absolute Error : ", metrics.median_absolute_error(
44             Y_test, Y_pred))
45         # print("Mean Squared Error : ", metrics.mean_squared_error(Y_test,
46             Y_pred))
47         print("R2 Score : %.3f" % metrics.r2_score(Y_test, Y_pred))
48         print("-----\n")
49
50     if __name__ == "__main__":
51         train()

```

Listing 5: Classical Regressor

8.2.5 BiLSTM 与 MLP 模型代码

```

1 # file: BiLSTM_MLP.py
2 # -*- coding: utf-8 -*-
3 import sys
4 import jieba
5 import pickle
6 import codecs
7 import math
8 import time
9 import pandas as pd
10 import numpy as np
11 import tensorflow as tf
12 import rnncell as rnn
13 from collections import Counter
14 from npdata_provider import data_provider_with_processor
15 from sklearn import metrics
16

```

```

17 def segment(sent):
18     """
19     该函数负责将句子分词
20     sent: 需要进行分词的句子
21     return: 分词后的结果jieba
22     """
23     # 如果要去停用词，把停用词表做成一个，处理速度要快一点dict
24     str_seg = jieba.cut(sent)
25     res_str = " ".join(str_seg).split(" ")
26     return res_str
27
28 def get_stop_words():
29     file = codecs.open('data_wjc/baidu_chinese_stopwords.txt', 'r',
30     encoding='utf-8')
31     dic_stop_word = {}
32     for line in file:
33         line = line.strip()
34         dic_stop_word[line] = line
35
36     return dic_stop_word
37
38 def get_word_dict(sents_list):
39     """
40     该函数负责构建一个词典
41     sents_list: 句子构成的一个list
42     return: 分词构成的字典 ({word,} id}
43     """
44     count = []
45     word2idx = {}
46     max_len = 0
47     # print(len(sents_list))
48     # 得到停用词表中的所有停用词，存储在中remove_word_list
49     dic_stop_word = get_stop_words()
50     remove_word_list = []
51     for key in dic_stop_word.keys():
52         remove_word_list.append(key)
53     # 得到数据集中除了停用词之外的所有词
54     words = []
55     for sent in sents_list:
56         tmp = []
57         sent = sent.strip()
58         str_seg = jieba.cut(sent)
59         res_str = " ".join(str_seg).split(" ")
60         max_len = max(max_len, len(res_str))
61         for word in res_str:
62             if word not in remove_word_list:
63                 tmp.append(word)
64         words.extend(tmp)
65
# 对全体词进行词频排序

```

```

66     count.extend(Counter(words).most_common())
67
68     # 按照词频对每个词赋予一个，词频大的 idlabel 值靠前id
69     for word, _ in count:
70         if word not in word2idx:
71             word2idx[word] = len(word2idx)
72
73     with open('pkl_data/word_dict.pkl', 'wb') as f:
74         pickle.dump(word2idx, f)
75
76     return word2idx, max_len
77
78 def get_sents_code(sents_list, word2id, len_sent):
79     """
80     该函数根据一个词的字典，将句子转换为对应数字的列表
81     sents_list: 句子构成的一个list
82     : 分词构成的字典word2id ({word,} id)
83     : 句子的最大长度，如果句子没有该长度就补len_sent0
84     return: 分词构成的字典 ({word,} id)
85     """
86
87     sents_code = []
88     dic_stop_word = get_stop_words()
89     remove_word_list = []
90     for key in dic_stop_word.keys():
91         remove_word_list.append(key)
92     for sent in sents_list:
93         tmp = []
94         sent = sent.strip()
95         str_seg = jieba.cut(sent)
96         res_str = " ".join(str_seg).split(" ")
97         for word in res_str:
98             if word not in remove_word_list:
99                 tmp.append(word2id[word])
100    while True:
101        if len(tmp) == len_sent:
102            break
103        tmp.append(0)
104    sents_code.append(tmp)
105
106    return sents_code
107
108 def bi_lstm_layer(lstm_inputs, lstm_dim):
109     """
110     该函数负责将句子用双向编码，取最后的句子编码LSTM
111     lstm_inputs: [batch_size, num_steps, emb_size]
112     lstm_dim 由于输出结果维度中最后一维是2*, 而我们需要的是lstm_dim
113     [batch_size, emb_size]
114     那么根据这个关系我们可以确定我们需要的会是lstm_dimembedding_size/2
115     return: [batch_size, 2*lstm_dim]
116     """

```

```

116     with tf.variable_scope("bi_lstm"):
117         lstm_cell = {}
118         for direction in ["forward", "backward"]:
119             with tf.variable_scope(direction):
120                 lstm_cell[direction] = rnn.CoupledInputForgetGateLSTMCell(
121                     lstm_dim,
122                     use_peepholes=True,
123                     initializer=tf.random_normal_initializer(stddev=0.1),
124                     state_is_tuple=True)
125         outputs, final_states = tf.nn.bidirectional_dynamic_rnn(
126             lstm_cell["forward"],
127             lstm_cell["backward"],
128             lstm_inputs,
129             dtype=tf.float32)
130         # 取出最后一个时刻句子的结果作为后续输入outputembedding
131         out = tf.concat(outputs, axis=2)
132         res = out[:, -1, :]
133         return res
134
135 X_train, X_test, Y_train, Y_test = data_provider_with_processor(
136     selected_y = 1)
137 titledata = pd.read_csv('data_wjc/all_title_subtitle.csv', encoding='gbk')
138 titles = titledata['title'].values
139 subtitles = titledata['subtitle'].values
140 all_titles = list(titles) + list(subtitles)
141 word2id, len_sent = get_word_dict(all_titles)
142
143 # titles_code=get_sents_code(titles,word2id)
144 # subtitles_code=get_sents_code(subtitles,word2id)
145 # print(titles_code)
146 # print(subtitles_code)
147
148 # word2id_title, len_title_sent = get_word_dict(titles)
149 # word2id_subtitle, len_subtitle_sent = get_word_dict(subtitles)
150 # print(len_title_sent, len_subtitle_sent)
151
152 # 设置词嵌入的大小
153 emb_size=int(sys.argv[1])
154 vocab_size=len(word2id)
155 col_struct_datas=X_train.shape[1]-2
156
157 print("emb size:",emb_size)
158 print("vocab size:",vocab_size)
159 print("len of sent:",len_sent)
160 print("col num of struct datas:",col_struct_datas)
161
162 struct_datas=tf.placeholder(tf.float32,[None,col_struct_datas])

```

```

163 y_datas=tf.placeholder(tf.float32,[None])
164
165 unstruct_datas_subtitle=tf.placeholder(tf.int32,[None,len_sent])
166 unstruct_datas_title=tf.placeholder(tf.int32,[None,len_sent])
167
168 emb_table=tf.Variable(tf.random_normal([vocab_size,emb_size],stddev=0.05))
169
170 title_emb=tf.nn.embedding_lookup(emb_table,unstruct_datas_title)
171 subtitle_emb=tf.nn.embedding_lookup(emb_table,unstruct_datas_subtitle)
172 # print(title_emb, subtitle_emb)
173
174 # 这里是用的title
175 bilstm_output=bi_lstm_layer(title_emb,lstm_dim=emb_size)
176 # print(bilstm_output)
177 mlp_input=tf.concat(axis=1,values=[struct_datas,bilstm_output])
178
179 mlp_input_dim=col_struct_datas+emb_size*2
180 hidden_dim=mlp_input_dim
181 print("mlp input dim:",mlp_input_dim)
182 print("hidden dim:",hidden_dim)
183
184 W1=tf.Variable(tf.random_normal([mlp_input_dim,hidden_dim],stddev=0.05))
185 b1=tf.Variable(tf.random_normal([mlp_input_dim],stddev=0.05))
186 W2=tf.Variable(tf.random_normal([hidden_dim,hidden_dim],stddev=0.05))
187 b2=tf.Variable(tf.random_normal([hidden_dim],stddev=0.05))
188 W3=tf.Variable(tf.random_normal([hidden_dim,1],stddev=0.05))
189
190 hidden_units1=tf.sigmoid(tf.matmul(mlp_input,W1)+b1)
191 hidden_units2=tf.sigmoid(tf.matmul(hidden_units1,W2)+b2)
192 mlp_output=tf.matmul(hidden_units1,W3)*100
193 # print(mlp_output)
194
195 lr=float(sys.argv[2])
196 print("lr:",lr)
197 loss = tf.reduce_mean(tf.square(tf.squeeze(mlp_output)-y_datas))
198 abs_loss = tf.reduce_mean(tf.abs(tf.squeeze(mlp_output)-y_datas))
199 # loss=tf.nn.l2_loss(mlp_output-y_datas)
200 opt=tf.train.AdamOptimizer(learning_rate=lr).minimize(loss)
201
202 init_var = tf.initialize_all_variables()
203 sess = tf.Session()
204 sess.run(init_var)
205
206 X_train_title=[]
207 X_train_tmp=X_train[:, -2:-1]
208 for tmp in X_train_tmp:
209     X_train_title.append(tmp[0])
210
211 X_train_subtitle=[]

```

```

212 X_train_tmp=X_train[:, -1:]
213 for tmp in X_train_tmp:
214     X_train_subtitle.append(tmp[0])
215
216 X_test_title=[]
217 X_test_tmp=X_test[:, -2:-1]
218 for tmp in X_test_tmp:
219     X_test_title.append(tmp[0])
220
221 X_test_subtitle=[]
222 X_test_tmp=X_test[:, -1:]
223 for tmp in X_test_tmp:
224     X_test_subtitle.append(tmp[0])
225
226 # print(X_train_title)
227 # print(X_train_subtitle)
228 # print(Y_train)
229 # print(X_train[:, :-2])
230
231 # train
232 total_train_size = X_train.shape[0]
233 total_test_size = X_test.shape[0]
234
235 batch_size = int(sys.argv[3])
236 nepoches = 10000
237 print("batch size:", batch_size)
238
239 n_train_batch = math.floor(total_train_size/batch_size)
240 n_test_batch = math.floor(total_test_size/batch_size)
241 results=[]
242 ltime=time.localtime(time.time())
243 strftime=str(ltime.tm_mon)+str(ltime.tm_mday)+str(ltime.tm_hour)+str(ltime
    .tm_min)+str(ltime.tm_sec)
244 for epoch in range(nepoches):
245     for i in range(n_train_batch):
246         lidx = batch_size*i
247         ridx = batch_size*(i+1)
248         feed_dict = {
249             struct_datas: X_train[lidx:ridx, :-2],
250             y_datas: Y_train[lidx:ridx],
251             unstruct_datas_title: get_sents_code(X_train_title[lidx:ridx
                ], word2id, len_sent),
252             unstruct_datas_subtitle: get_sents_code(X_train_subtitle[lidx
                :ridx], word2id, len_sent),
253         }
254         train_loss, train_opt = sess.run([loss, opt], feed_dict=feed_dict)
255
256         if i%5 == 0:
257             test_loss=0
258             test_abs_loss=0

```

```

259     for j in range(n_test_batch):
260         lidx = batch_size * j
261         ridx = batch_size * (j + 1)
262         feed_dict = {
263             struct_datas: X_test[lidx:ridx, :-2],
264             y_datas: Y_test[lidx:ridx],
265             unstruct_datas_title: get_sents_code(X_test_title[
266                 lidx:ridx], word2id, len_sent),
267             unstruct_datas_subtitle: get_sents_code(
268                 X_test_subtitle[lidx:ridx], word2id, len_sent),
269             }
270         tmp_test_loss, tmp_abs_loss = sess.run([loss, abs_loss],
271         feed_dict=feed_dict)
272
273         test_loss += tmp_test_loss
274         test_abs_loss += tmp_abs_loss
275         test_loss /= n_test_batch
276         test_abs_loss /= n_test_batch
277
278         feed_dict = {
279             struct_datas: X_test[:, :-2],
280             y_datas: Y_test,
281             unstruct_datas_title: get_sents_code(X_test_title,
282                 word2id, len_sent),
283             unstruct_datas_subtitle: get_sents_code(X_test_subtitle,
284                 word2id, len_sent),
285             }
286         y_pred_tmp = sess.run(mlp_output, feed_dict=feed_dict)
287
288         tmp = []
289         tmp2 = []
290         for i in range(len(y_pred_tmp)):
291             tmp.append(round(y_pred_tmp[i][0]))
292             tmp2.append(round(Y_test[i]))
293         y_pred = tmp
294         y_true = tmp2
295         print("—— some pred results:", y_pred[0:10])
296         print("—— some true results:", y_true[0:10])
297
298         r2_score = metrics.r2_score(y_pred=y_pred, y_true=y_true)
299
300         print("epoch:", epoch, "batch:", i, "train loss:", round(
301             train_loss, 2), "test loss", round(test_loss, 2),
302             "abs loss", round(test_abs_loss, 2), "r2 score:", round(
303             r2_score, 4))
304         result={
305             "epoch": epoch, "batch": i, "train loss": train_loss, "
306             test loss": test_loss,
307             "abs loss": test_abs_loss, "r2 score": r2_score
308         }

```

```

301     results.append(result)
302     with open('pkl_data/result_%s_%d_%f_%d.pkl' % (strtime,
303                 emb_size, lr, batch_size), 'wb') as f:
304         pickle.dump(results, f)

```

Listing 6: BiLSTM and MLP

References

- Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996. ISSN 1573-0565. doi: 10.1023/A:1018054314350. URL <https://doi.org/10.1023/A:1018054314350>.
- Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL <https://doi.org/10.1023/A:1010933404324>.
- Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997. ISSN 0022-0000. doi: <https://doi.org/10.1006/jcss.1997.1504>. URL <http://www.sciencedirect.com/science/article/pii/S002200009791504X>.
- Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5):1189–1232, 10 2001. doi: 10.1214/aos/1013203451. URL <https://doi.org/10.1214/aos/1013203451>.
- Marti A. Hearst. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, July 1998. ISSN 1541-1672. doi: 10.1109/5254.708428. URL <http://dx.doi.org/10.1109/5254.708428>.
- Geoffrey Hinton. Connectionist learning procedures. *Artificial intelligence*, 40.1:185–234, 1989.
- David J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992. doi: 10.1162/neco.1992.4.3.415. URL <https://doi.org/10.1162/neco.1992.4.3.415>.
- Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.