一、原文介绍

准确的房价预测对买家、卖家、开发商以及政策制定者都非常重要。传统的房价预测方法主要依赖于专家的经验判断，这种方法不仅耗时耗力，而且准确性无法保证。因此，如何利用历史数据，应用机器学习算法进行快速准确的房价预测，成为了一个值得研究的问题。本文利用随机森林模型对房价数据进行预处理，基于 XGBoost 对房价进行预测；通过对比实验，发现优化后的 XGBoost 模型在房价预测上表现优异，预测准确率有了显著提高，表明 XGBoost 是一种有效的房价预测工具。

本文主要进行的工作主要有以下三点：

1.本文首先对数据集进行了特征工程，根据相关性对部分特征进行了剔除。其他的特征工程还包括分类变量的编码方式的探讨以及对连续变量的标准化。

2.随后对数据集做进一步处理，连续变量的缺失值以平均值填充，并进行标准化；分类变量的缺失值以出现最多的类别等方式进行填充，使用 Ordinal Encoder 进行编码。使用 Random Forest Regressor 进行特征选择，输出特征重要性，为 XGBoost 模型的训练做准备。

3.使用 XGBoost 进行预测，数据选择之前使用随机森林输出的特征重要性的前 30 作为特征变量；最后将问题转换为分类问题，预测房价的高和低。

本文做的比较好的方面在于对于过多的特征采取了使用随机森林模型进行筛选，这使得后续使用 XGBoost 时训练时间大幅减小。在实际训练当中，使用随机森林的效果明显好于 XGBoost，这是本文的一个缺陷。另一方面，直接将回归任务转化为分类任务意义不大，如何评定该地的房价是高还是低是一件比较主观的事情，不同的划分标准对于模型的表现具有很大的影响。

二、所作工作

1.数据准备

从 kaggle 网站上下载论文所需的数据集" House Prices-Advanced Regression Techniques" 。复现论文的结果需要用到。另一方面，从 58 同城二手房网站上爬取了北京，上海，成都在内的 12 座城市的二手房房价数据总共 16000 余条，数据集中采集到的信息包括 ID，houseLoc(房子所在门牌地址)，经纬度，每平方米价格等。

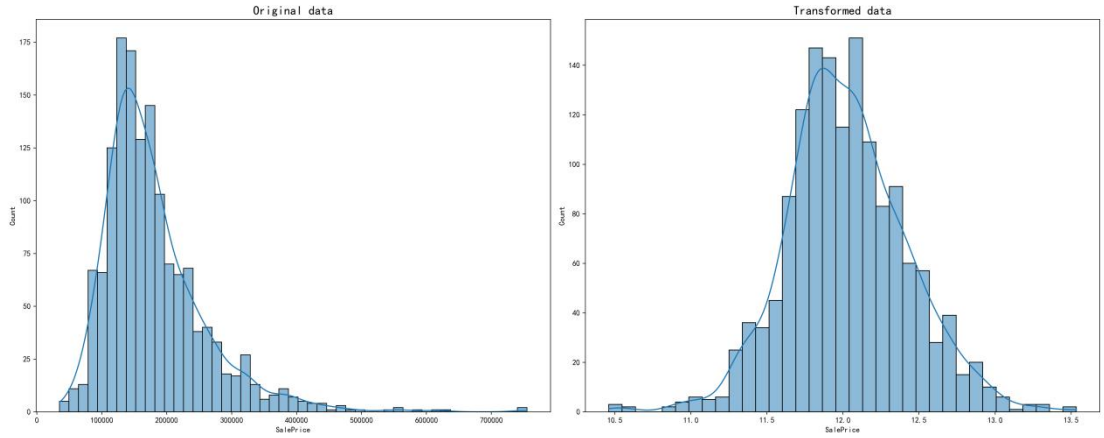
特征	释义	数据类型
ID	索引	int
houseLoc	门牌地址	str
longitude	经度	float
latitude	纬度	float
unitPrice	每平方米房价	float
housePrice	总房价	float
houseBedroom	卧室数量	int
houseBathroom	厕所数量	int
houseOrientation	房屋朝向	str
houseSubway	是否临近地铁	int
houseHousingPeriod	房本年限	str

续表：

特征	释义	数据类型
houseFloorType	房子所在楼层类型(低中高)	str
houseFloorSum	房子所在的楼层总数	int
subwayAround	周围地铁数量	int
parkAround	周围公园数量	int
schoolAround	周围学校数量	int
shopping_mallAround	周围商城数量	int

2.数据预处理

在获取到数据后，随即对数据进行处理。对于" House Prices-Advanced Regression Techniques" 数据集我按照论文中提到的做法进行处理。首先对原始数据集中的缺失值进行处理，数据集中部分特征存在空值 NAN，但这部分 NAN 并非缺失值，因此这部分数据按照论文的做法直接填充 None；其次，数据集中部分特征存在大量重复值或者说 NAN 值占比过于巨大，因此删除重复值占比超过 90%的列以及 NaN 太多的特征。然后，计算一些相似的特征之间的相关性，按文中的计算方法计算了四个相关系数，根据相关系数的结果，将"Exterior1st", "GarageQual", "GarageYrBlt", "1stFlrSF"四个特征删除。接着，对分类变量与连续变量进行分离，分类的标准为数据类型是否为 float64。分类变量若有缺失值且之前未处理的话，采用“以出现最多次数”的方式填充，并使用 OrdinalEncoder 进行编码，连续变量则使用均值填充。最后，对于目标变量发现存在左偏的情况，因此进行对数化处理，如图所示：



可以看到，取对数之后，偏度得到了一定的修正，目标的分布更加接近于正态分布。

完成处理后，总计有 1460 个样本，54 个特征。

特征数	样本数	目标
54	1460	Log(y)

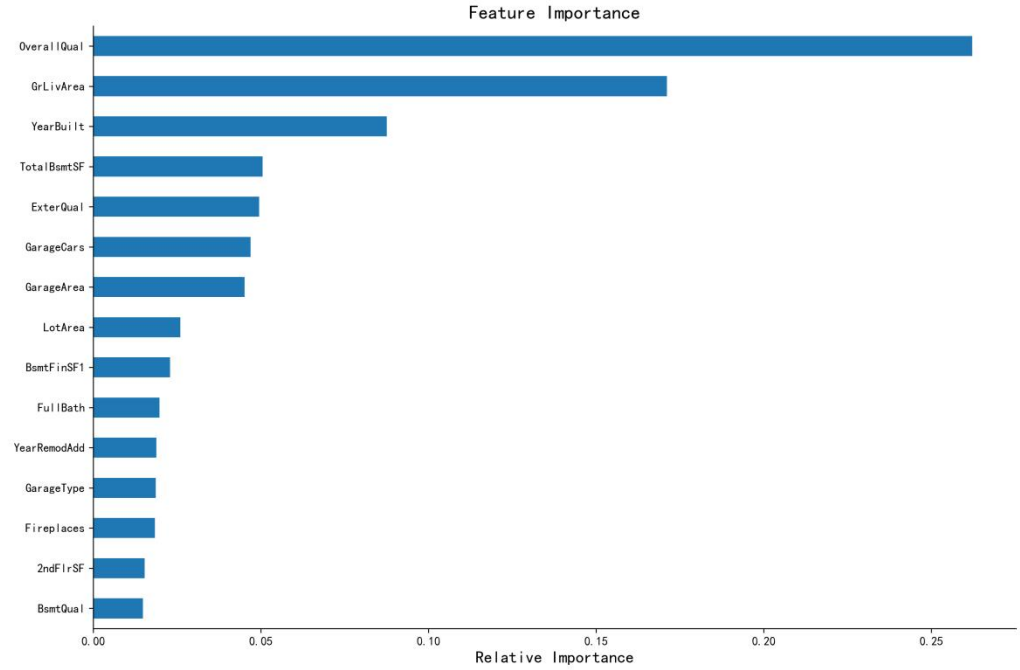
对于爬取到的数据，处理方式基本于论文中提到的方式相同。缺失值填充以及 NAN 值处理与论文中提到的方法相同。但对于分类变量的编码部分，各个类别存在明显大小关系的，诸如房间数量，采用 OrdinalEncoder 进行编码，其余分类变量则采用 OneHotEncoder 进行编码。目标变量同样进行对数化处理。由于获得了每间房子的经纬度坐标，因此再构造一个特征，即到市中心的距离。

3.论文中提到的模型训练

在处理完数据后，按照文中的办法对随机森林模型进行网格搜索，搜索范围以及结果如下：

参数	搜索范围	结果
n_estimators	[100, 200, 300, 400, 500]	500
max_features	[0.2, 0.4, 0.6, 0.8, 1]	0.4
max_samples	[0.2, 0.4, 0.6, 0.8, 1]	0.8
criterion	['squared_error', 'absolute_error', 'friedman_mse', 'poisson']	friedman_mse

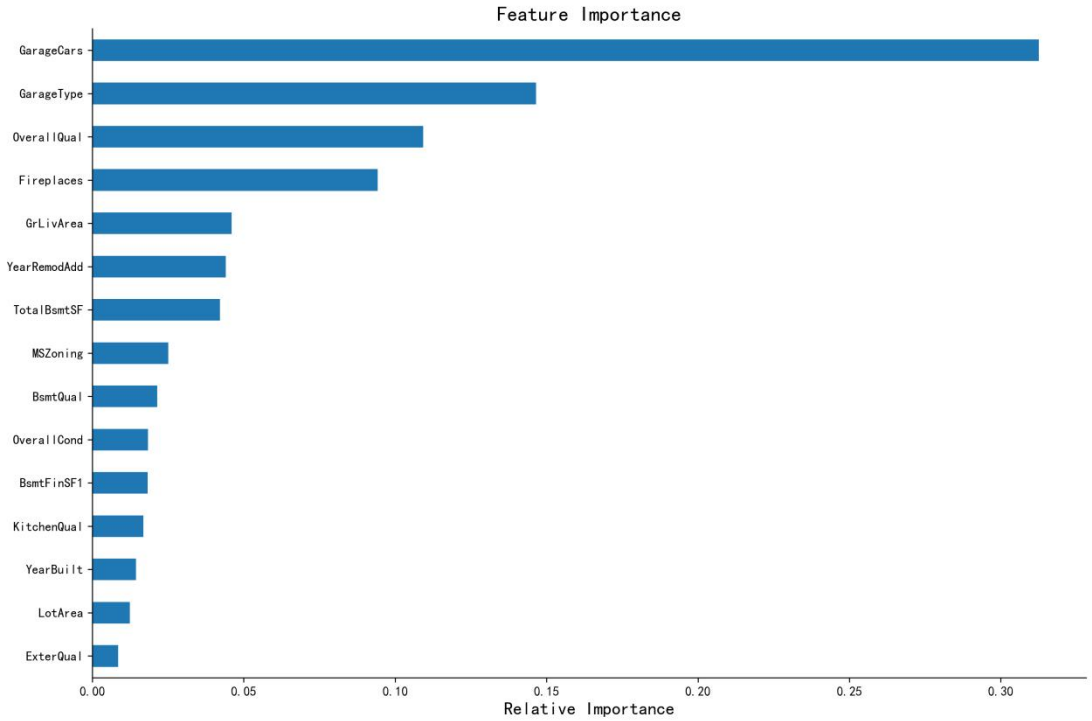
使用最佳模型在测试集上进行训练，得到模型的 R^2 以及 MSE 分别为 0.889，0.018，同时得到特征重要性(前 15 名)如图所示：



接下来，选取最重要的 30 个特征用于 XGBoost 模型的训练，进行网格搜索，搜索范围与结果为：

参数	搜索范围	结果
n_estimators	[100, 200, 300]	300
max_depth	[6, 7, 8]	6
learning_rate	[0.1, 0.2, 0.3]	0.1
objective	["reg:squarederror", "reg:gamma"]	reg:gamma

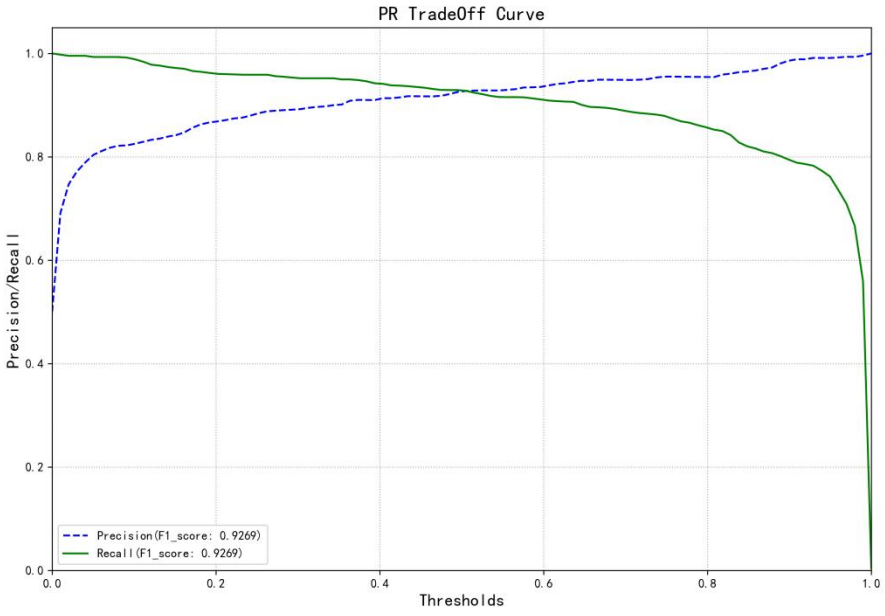
之后使用最佳模型在测试集上进行训练，得到模型的 R^2 以及 MSE 分别为 0.880，0.020，同时得到特征重要性(前 15 名)如图所示：



最后再训练一个 XGBoost 分类器，将房子分为房价高或者房价低。仍使用之前的特征作为训练集特征，但目标变量变为 0 或 1，训练任务从回归转为二分类。接下来对 XGBoostClassifier 进行网格搜索，搜索的范围以及搜索的结果如下：

参数	搜索范围	结果
n_estimators	[100, 200, 300]	300
max_depth	[6, 7, 8]	6
learning_rate	[0.1, 0.2, 0.3]	0.1
eval_metric	["auc", "error", "logloss"]	auc

为了权衡 Precision 和 Reacll，绘制 PR_tradeoff 图像如下图所示：

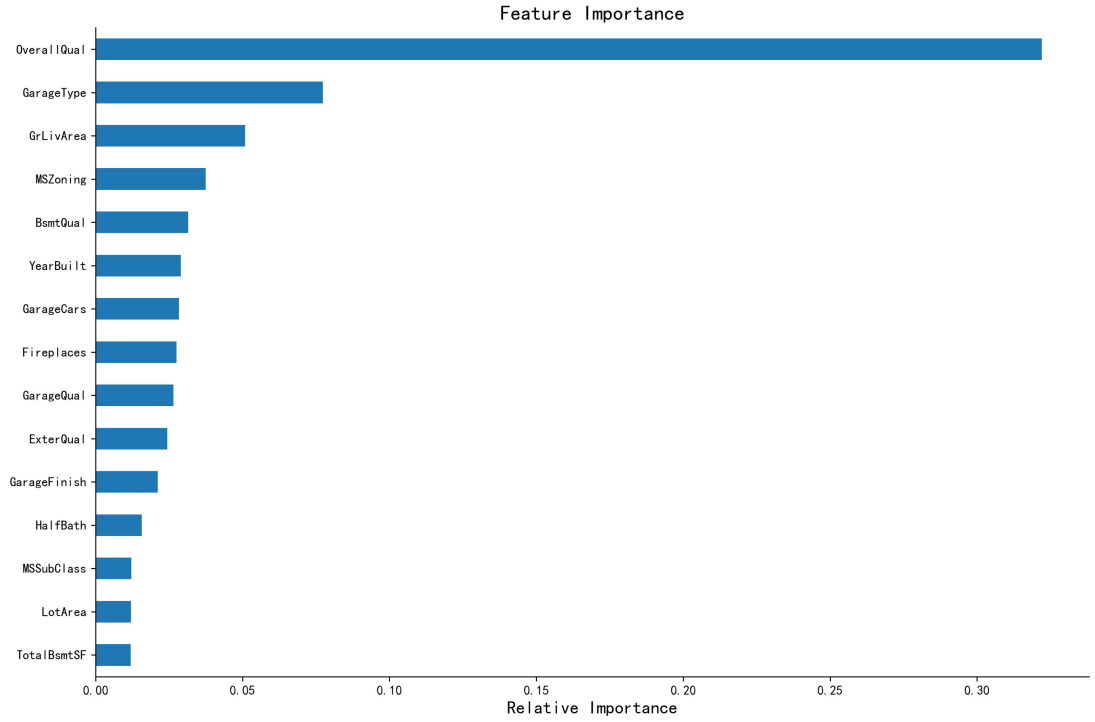


最终对模型的训练结果进行评估，输出分类报告与混淆矩阵，结果如下：

	Precision	Recall	F1-score	support
0	0.92	0.93	0.93	220
1	0.93	0.92	0.93	218
accucracy			0.93	438
macro avg	0.93	0.93	0.93	438
weighted avg	0.93	0.93	0.93	438

		Predict value	
		0	1
Actual value	0	205	15
	1	17	201

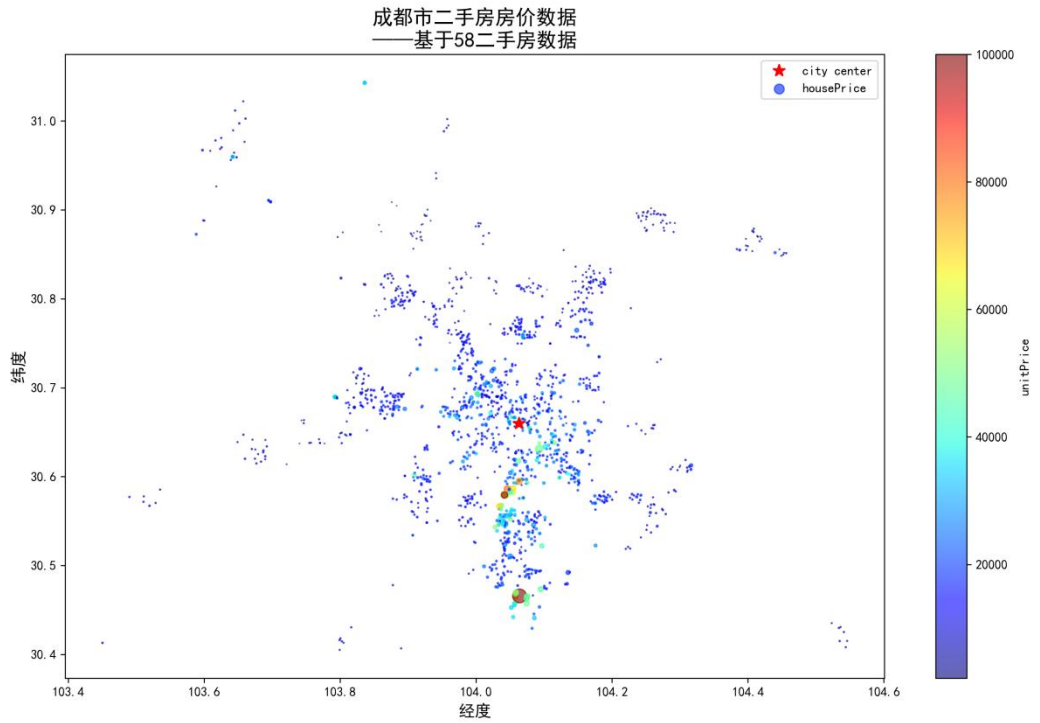
并输出分类模型的特征重要性：



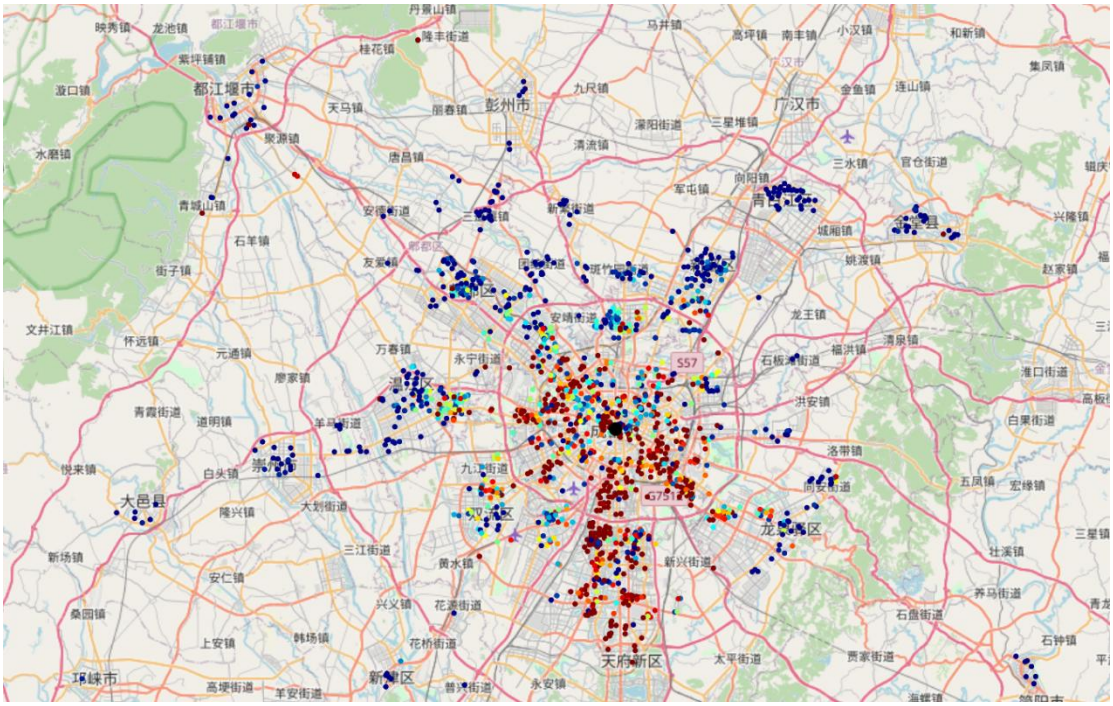
4.基于论文额外做的工作

基于论文额外所作的工作在于可视化与模型的选择。

以成都市的数据为例，进行可视化，结果如下：

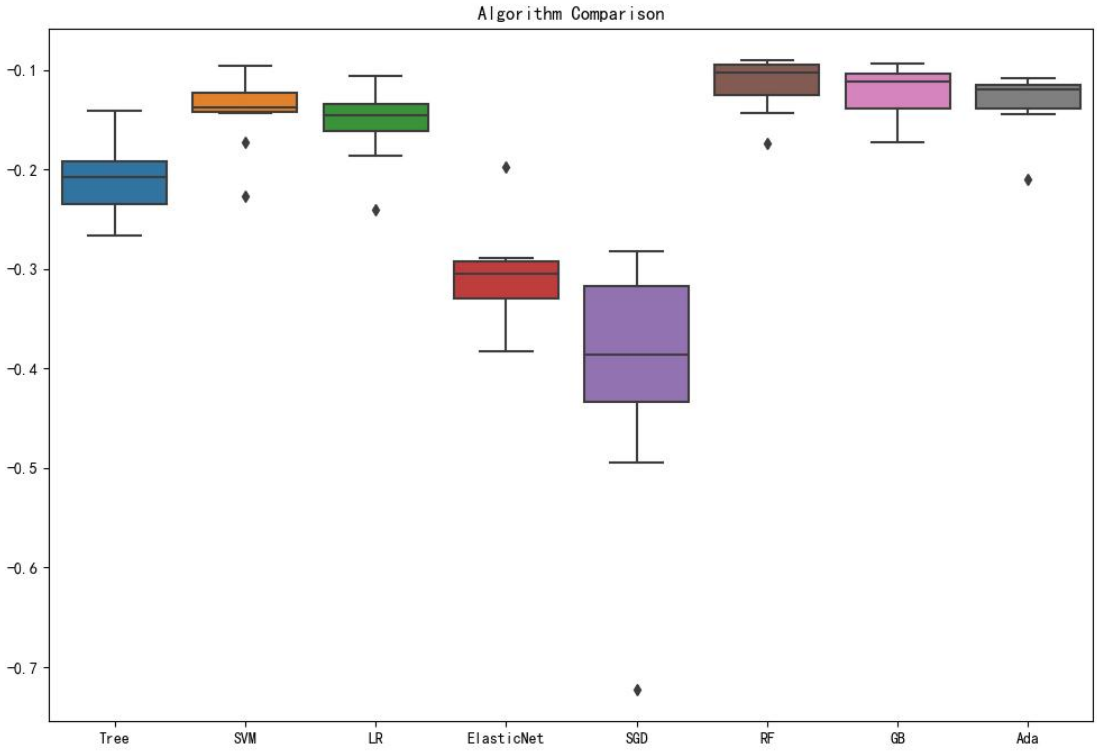


将结果画在地图上：



还有更多可视化结果受限于篇幅未能展示，可以在 figure 文件夹中看到。

我实现选择了一系列模型，并使用它们进行十折交叉验证。选择交叉验证平均分最高，分数方差最小的模型，结果如图所示：



从图中的结果来看，随机森林模型的效果最好。

三、总结

1. 本次作业学到了什么

完成这次作业我学到了很多，尤其是在进行整个机器学习任务的流程上收获颇多。我了解到了如何使用 Pipeline 将数据处理流程封装到一个完整的管道中，了解到如何自定义数据处理模块并将该模块应用到 Pipeline 中。除此之外，我对面向对象编程有了更加深刻的理解，主要在使用对象基类为其他类传递信息等；对基于 requests 以及 lxml 编写爬虫也有了更深入的理解，可以更加熟练的使用 xpath 表达式解析网页数据。

2. 过程中最困难的，最意想不到的地方

在这次作业中最最困难的还是编写爬虫。在编写爬虫期间遇到了各种各样的“反爬”措施，我查阅了一些网络爬虫开发相关的技术书籍，使用了包括但不限于 JS 逆向 58 同城用户登录逻辑，cookie 反爬等技术手段，但都没有解决爬取失败的问题。结果，最让我想不到的是，58 同城的反爬手段防老手不防新手，换句话说，我不需要任何反爬只需要一直刷新网页就能把网页数据请求到。我花了半个月的时间解决反爬结果以这样戏剧性的结果收尾实在是忍俊不禁。

3. 本次作业做的过程中遇到了什么问题，如何解决的

本次作业中遇到的问题或者说遭遇的挑战就是对于房子的位置信息与 POI 信息的解析。通过查阅 CSDN 等网站了解到百度地图 API 可以胜任这项工作，于是就注册了百度 API 的账号，但是 API 的官方文档比较晦涩，完全啃明白又花了一些时间。能够成功解析出数据之后呢，又因为数据量较大，一条一条的解析所花的时间过长，最后采取多线程的方式尽心解析，原本解析一次需要半天时间，修改后就只需要 20 分钟了。

4. 跟之前所制定的计划相比，差异最大的地方

与我之前制定的计划相比，我基本上圆满完成了计划中的任务，不过仍然有没有完成的地方。我在发现训练出模型之后模型的泛化能力很差，于是尝试提升模型的泛化能力，但是由于数据量较大的缘故网格搜索未能跑出最佳的结果。这里感到挺遗憾的。

5. 给未来修读这门课程的同学的建议

这门课程的难度其实没有大家想象的那么大，作业啊课堂练习啊也不是什么非常抽象耗时的任务，只要上课认真听老师讲课下最多俩小时就能做的七七八八。在我看来，这门课最重要的不仅仅是上课所学到的东西，而是课后的练习以及最后整大作业薅掉头发的时候。做出来了能高兴好久，具有成就感，做不出来也没关系，至少尝试过努力过；个人在其中辛苦付出收获的东西才是最重要的。刘老师上课非常有意思，幽默风趣，氛围也十分不错，希望各位学弟学妹们能够做出更厉害的成果。

四、代码运行说明

部分代码由于运行时间很长，加入主流程当中又显得冗余，因此这部分代码以测试代码的形式给出，测试代码的运行时间最长不超过 20 分钟(主要是因为爬虫部分 max_retry 我设置的 None，爬取器将一直发送请求直到获得数据，所以想要进一步减少时间可以改 max_retry)。此外，数据的解析器需要使用百度地图 API，请自行设置(由于包含隐私信息我就不提供了(doge))。

其他的代码运行说明即相关技术文档可以参见随代码附上的 README 文件。文件中详细说明了如何配置环境，进行运行准备以及项目结构等内容。为了更好的阅读体验请参见我的 Github 仓库：https://github.com/FantasySilence/DataAnalysisFinal。如果在代码运行或环境配置上仍有问题，请向我提 issue。

最后的最后，非常感谢刘凌老师 17 周的辛苦付出。