武汉市房屋租赁价格建模分析

摘要

近些年来，随着经济的不断发展，全国大城市的房屋租金不断的上涨，为避免由房屋租金过高而引发一系列的社会问题，房屋租金需要在一合理范围内波动。为研究房屋租金的合理波动范围，本文以武汉市为例，首先考虑宏观与微观各方面的因素，建立了武汉市房屋租赁价格的影响指标体系; 之后在此基础上建立了合理的数学模型，对武汉市房屋租赁指导价格进行了分析; 最后预测了今后五年内的武汉市房屋租赁价格。

针对问题一首先我们通过查阅大量文献资料, 结合相关文献与实操性，最后得到 8 个三级指标，同时通过用 Python 爬取相关网站的数据，获取了九个变量。通过因子分析的方法，我们对每个指标赋予权重，将其融合为 5 个二级指标与 2 个一级指标。

针对问题二，在第一题所构建的价格影响指标体系基础上，我们筛选了其中相关性显著的关键指标进行进一步的分析。由于宏观数据是时间序列数据，所以我们将这两个部分分开来分析。宏观方面，我们对 2012 年到 2019 年的进行建模分析。考虑到数据量与自由度的问题，我们通过计算第一-问中的二级指标指数, 将自由度较低的二级指标与二手房房屋租赁价格指数做多元线性回归，并检验其异方差、序列相关性，保证模型的正确性，得到最终的模型：房屋租赁价格指数 $= 3223.868 + 0.636$ 区域特征 -0.075 人民生活 -28.014 教育水平。微观方面，我们使用了多种机器学习的算法，比较了这些算法的可解释性与准确率之后，选择了基于线性神经网络的随机梯度下降算法，得到模型为 $houseprice = 55.71rooms + 13.51halls + 22.11area + 16.63classes + 19.37direct + 14.18elevator + 21.50railway + 43.02place$。经测算，发现对房屋租价格影响最大的是房间面积。

针对问题三, 由于第二问所做的是有偏估计, 我们同时采用第二问的宏观经济模型以及 ARMA 模型对时间序列数据进行预测。宏观经济模型; 在 ARMA 中，将模型在已知样本上进行测试，发现与原始房屋租赁价格指数走势图吻合程度较高。因此对下一年的数据进行预测，发现房屋租赁价格指数虽局部呈下降趋势，但总体上呈上升趋势。结合相关政策分析，政府在近几年房价上涨过程中做出了有效调控，使房价稳定，租房市场的泡沫现象的以缓解，为了进一步控制房价，仍需要更有效的把控。

关键字： 关键词房屋租赁价格预测　因子分析　ARMA 模型　机器学习

目录

一、问题重述 ····· **4**

二、问题分析 ····· **4**

2.1 描述性统计分析 ····· 4

2.2 房屋租赁价格指数体系的建立 ····· 4

2.3 房屋价格建模 ····· 5

2.4 房屋价格预测 ····· 5

三、模型假设 ····· **5**

四、问题求解 ····· **5**

4.1 描述性统计分析 ····· 5

4.1.1 房地产开发投资额、城乡居民储蓄年末余额两者之间关系 ····· 6

4.1.2 普通本专科学生人数与地方一般公共预算支出 ····· 6

4.1.3 CPI 与房地产开发投资额 ····· 7

4.1.4 房地产开发投资额与住房租赁价格指数 ····· 7

4.2 问题一 ····· 8

4.2.1 指标筛选 ····· 8

4.2.2 指标体系的构建 ····· 8

4.3 问题二 ····· 9

4.3.1 宏观经济模型 ····· 10

4.3.2 微观决策模型 ····· 11

4.4 问题三 ····· 18

4.4.1 宏观经济模型 ····· 18

4.4.2 ARMA 模型 ····· 18

五、模型不足 ····· **21**

参考文献 ····· **22**

附录 A　附录 1 ····· **23**

附录 B　武汉市租房价格爬虫 A——安居客 ····· **23**

附录 C　武汉市租房价格爬虫 B——链家 ····· **26**

附录 D　房屋租赁价格机器学习模型建模 python 代码 ····· 27

一、 问题重述

近年来，全国各地的房地产市场风云变幻，各地房价相关的政策也层出不穷。虽然大部分人关注的是房屋的买卖问题，但是对于大学生来说，房屋租金显然更为重要。总的来看，近几年大城市的房屋租金一直处于上涨的态势，很多一线城市的房价已经接近大多数人的承受限度。过高的租金常常会引起社会矛盾与一系列的社会问题，因此，房价应当在一定合理的范围内波动。为了更好地监控、预测房价的变化，请建立模型分析以下问题：

（1）什么因素在影响房屋租赁价格？尝试建立武汉市房屋租赁价格的影响指数体系。

（2）根据武汉市房屋租赁交易的历史数据，建立房屋租赁指导价格模型。

（3）请对今后五年内的武汉市房屋租赁价格进行预测。

二、 问题分析

本文针对二手房租赁价格问题展开研究，首先从描述性统计分析入手，由指标体系、房屋价格建模以及宏观时间序列预测三个角度重点分析该问题，综合了宏观、微观两个方面，给出了一套完整的分析框架。下面分别从四个角度对问题进行分析。

2.1 描述性统计分析

描述性统计分析是探索数据的必不可少的环节。我们对从统计年鉴以及爬虫中获得的数据进行初步的作图分析，得到了一些基础结论，便于后面建立模型、筛选变量使用。

2.2 房屋租赁价格指数体系的建立

房屋租赁不仅仅是一个宏观上的问题，也是一个微观上的问题。首先，从宏观经济上看，某地区房屋租赁价格必然受到各个宏观因素的影响，同时，从微观上看，一个房屋的价格，很可能与它所处的位置、它的户型、面积等等有着密不可分的关系。为了更好地建立房屋租赁价格指数体系，我们结合了微观与宏观两个方面，通过对这两个方面结果的加权平均，得出一个同时兼顾宏观经济分析与个人决策的指标体系。

为了建立合适的指标体系，筛选出真正有影响的变量，我们参考了大量文献，在这些文献的基础上加以整合，并与实际可操作性相结合，最后得到了 8 个三级指标，这些指标能够较完整地反映出一个区域的特征。微观方面，我们使用 Python 爬取了链家网、安居客两个网站的大量二手房出租信息，获取了 9 个变量，这些变量很好地刻画了房屋

特征。通过因子分析的方法，我们对每个指标赋予权重，并将其融合为 5 个二级指标与 2 个一级指标。

2.3 房屋价格建模

同样采取与第一问类似的框架，我们仍然从宏观与微观两个角度上看待房屋租赁的问题。由于宏观数据是时间序列数据，所以我们将这两个部分分开来分析。宏观方面，我们对 2012 年到 2019 年进行建模分析。考虑到数据量与自由度的问题，我们通过计算第一问中的二级指标指数，将自由度较低的二级指标与二手房房屋租赁价格指数做多元线性回归，并检验其异方差、序列相关性，保证模型的正确性。

微观上，我们主要使用了机器学习与深度学习中的方法对房价进行预测。考虑到模型的可解释性，我们采用了比较基础的便于解释的方法——多元线性回归、$logistic$ 回归、$elastic\quad net$ 族回归、随机森林以及深度学习中的多层感知机模型。对于数据，我们划分了测试集与训练集便于检测模型的拟合优度。由于这是一个回归的机器学习问题，所以我们没有用广义上的预测准确率来衡量，而是通过测试集均方误差来衡量模型的拟合优度。通过比较各个模型的优缺点与最后的拟合效果，我们选出最优的模型做微观分析。

2.4 房屋价格预测

根据前两文的结果，我们对上述模型中的变量，根据平均增长率对其未来五年的值进行预测。将变量的预测值带入模型中，我们就可以得到模型具体的预测结果。同时，考虑到时间序列模型的特殊性，我们对原始序列进行平稳性检验，并在平稳性检验通过的条件下，建立 ARMA 模型，对其未来五年进行预测。根据两个模型的预测结果的综合比较，我们最后可以得出一个较为精确的房屋价格指数预测值。

三、 模型假设

- 房屋的价格同时受到宏观与微观的影响；
- 所有的经济变量可以对经济政策的改变做出迅速反应；
- 经济变量之间是相互独立的。

四、 问题求解

4.1 描述性统计分析

我们首先通过描述性统计分析的方法对主要的几个经济变量与总指标之间的趋势相关性做了初步判断，结合目前武汉市政府出台的住房政策，对数据有一个初步的认识。

4.1.1 房地产开发投资额、城乡居民储蓄年末余额两者之间关系

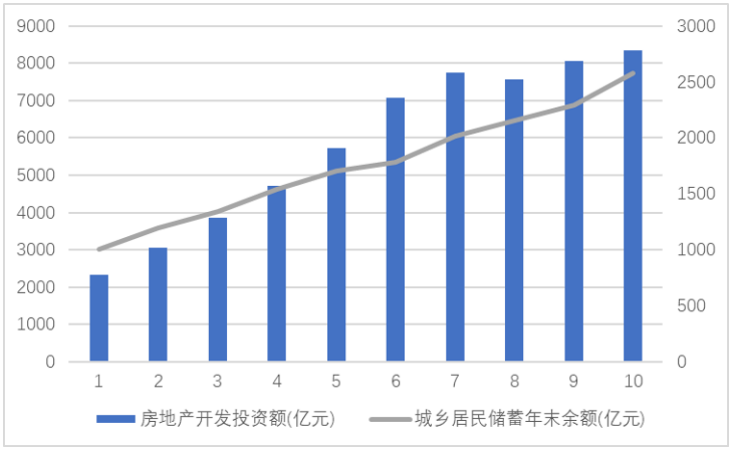


图 **1**　房地产开发投资额与城乡居民储蓄年余额关系图

由上图分析，房地产开发投资额与城乡居民储蓄逐年上涨并呈正相关关系。双琰（2011）在《刍议居民储蓄与房价的关系》中提到，银行从储户手中吸取储蓄存款, 再将其以贷款的形式贷给房地产商, 房地产商获得金融资本后, 购置土地等生产要素, 着手房产开发, 最后将开发成熟的商品房出售给消费者。因此我们认为城乡居民储蓄的增长对于房地产开发投资额增长做出了巨大贡献，进而推动租赁市场额的发展。根据《武汉租赁市场发展现状及分析》，2014 年-2017 年，武汉新建住房销量达到 105 万套，存量房市场大，而其中租赁房屋的常驻居民家庭占比 15.2%，租赁房源充足。

4.1.2 普通本专科学生人数与地方一般公共预算支出

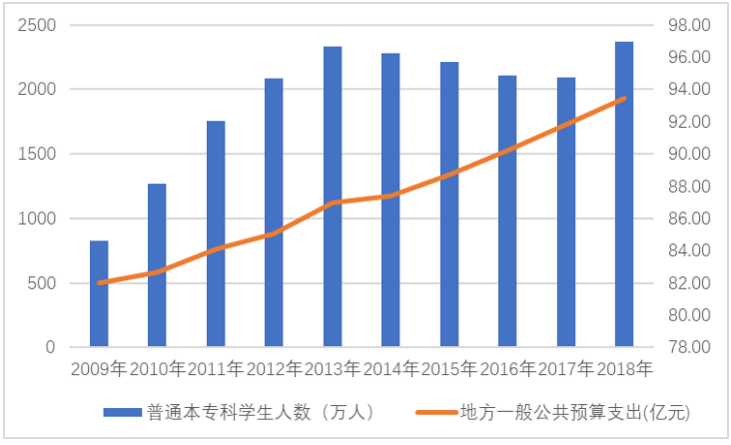


图 **2**　居民受教育程度与地方一般公共预算支出分析

由上图分析可得，普通在本专可学生人数与武汉市一般公共预算支出呈正相关关系。随着武汉“百万大学生留汉创业计划工程”的提出与实施，武汉大学毕业生流汗人

数逐年上升，2017 年人数为 30。1 万任，新落户 14.2 万人，分别是 2016 年的 2 倍、6 倍。同时，随着武汉大学生保障房政策的落实，目前（截止至 2019 年 10 月），武汉已通过配建、包租等方式筹集大学生毕业租赁房 17191 套，约 74 万平方米；另集中新建 5 个保障房项目，共 2840 套，26.8 晚平方米。申请租房的留汉大学生超过 99% 都获得了八折公租房。

4.1.3 CPI 与房地产开发投资额

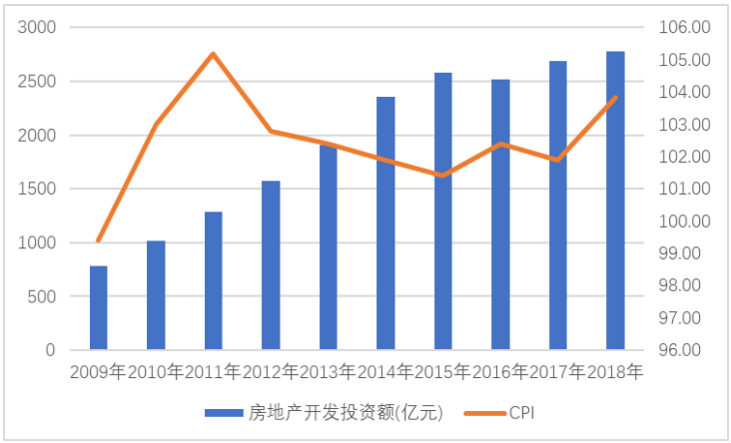


图 **3**　房地产开发投资额与 **CPI** 分析

汪彩玲（2010）在《房地产价格指数与居民消费价格指数关系的实证检验》指出，房地产价格指数与居民消费价格指数存在长期稳定的关系，房地产价格的上升会导致即期和未来的一般消费价格水平上升。随着居民收入的增加，房地产在中国的资产结构中占据着越来越重要的地位。

4.1.4 房地产开发投资额与住房租赁价格指数

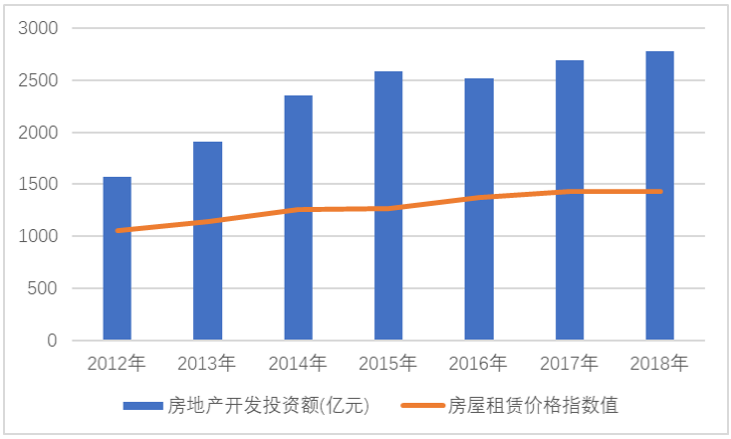


图 **4**　房地产开发投资额与住房租赁价格指数分析

由上图分析可得，武汉市住房租赁价格指数随着房地产开发投资额的增长而逐年上升。结合上文分析以及相关资料可得，在政策方面，在住建部坚持 2019 年坚持以稳地价、稳房价、稳预期为目标、促进房地产市场平稳健康发展的前提下，2019 年武汉市房地产市场政策的基本方向并未改变，出台房地产刺激政策的可能性不大。同时，2019 年房地产市场管理将更加注重精准调控和精细监管，武汉楼市政策制定仍将以新房限价和稳定刚需为主。在人口方面，随着留汉大学生数量都逐年增长以及外来人口的持续增加，未来租赁市场保持稳定发展。房屋租赁价格指数值增幅稳定。

4.2 问题一

4.2.1 指标筛选

我们首先将总指标分为两个一级指标，分别代表了微观经济以及宏观经济对于房屋租赁价格的影响。然后，通过文献综述法研究了学界目前对于房屋租赁指标体系的构建的大致观点，通过查阅武汉市统计年鉴、赵志强、赵予新、郑文娟等的有关工作，初步得到了地区生产总值、人口总数、预算支出、房地产投资额、CPI、城市居民储蓄余额、收入、就业总人口、普通本专科人数变量。

微观方面，小组内通过浏览房天下、链家、安居客等网站，通过 python 爬虫，最终获得 3500 条数据，经过筛选，比对，剔除无效以及错误数据后，最终实际数据 3300 条。变量包括房间数、厅数、总面积、出租类型、朝向、是否有电梯、地铁线、所属行政区等指标。我们将地区生产总值、人口总数、预算支出、房地产投资额、CPI 这五项指标归纳为区域经济特征指标，将城市居民储蓄余额、收入、就业总人口归纳为人民生活指标，将普通本专科人数归纳为教育水平。

4.2.2 指标体系的构建

由于数据与自由度之间的矛盾，我们采用主成分分析的方法对变量进行线性降维，对原始变量进行正交变换。经过主成分分析之后得到的主成分为均值为 0，方差为 1 的标准正态分布。各主成分之间不存在共线性，因此不对主成分之间进行共线性检验。主成分也满足同方差零均值假定。主成分分析最终得到的结果为进入变量为二级指标一为主成分一得到了得分矩阵为（其中 Z_1 为主成分一，Z_2 为主成分二）：

表 **1**　成分得分系数 **1**

指标	得分
地区生产总值	0.273
总人口	0.251
地方一般性公共支出	0.271
房地产开发投资额	0.249

我们对三级指标做主成分分析，目的求得最终各指标的权重，由图所示为归一化之前的权重，将其归一化后我们得到二级指标区域经济特征中的地区生产总值、人口总数、预算支出以及房地产投资额三级指标所求得的权重分别为：0.2615 、0.2404 、0.2596 、0.2385。将其归一化后最终二级指标人民生活所对应的三级指标城市居民储蓄余额、收入（在职职工平均工资）、就业率分别对应的权重为：0.3379 、0.3369 、0.3252。

求得三级指标权重后，我们根据加权和计算二级指标所对应的值，同样采用主成分分析法求得二级指标的权重分别为：

表 **2**　成分得分系数 **2**

指标	得分
教育水平	0.177
区域经济特征	0.480
人民生活	0.474

归一化后，教育水平、区域经济特征、人民生活得权重分别为：0.1421、0.4298、0.4281。一级指标对宏观经济特征和微观经济特征进行等分，各取 0.5 权重。

微观层面，由于权重计算十分糅杂，，因此我们采用平分的方法对三级指标、二级指标赋予权重。最终建立的房屋租赁价格影响指标体系见附件 1。

4.3 问题二

根据流程图，我们在第一问建立的指标体系上进行分析。根据指标体系的两个一级指标数据的不同特征，我们将问题分为两个部分讨论。宏观经济方面，由于受到自由度的限制，我们根据指标体系求得的权重计算出二级指标的具体值，然后使用多元线性回

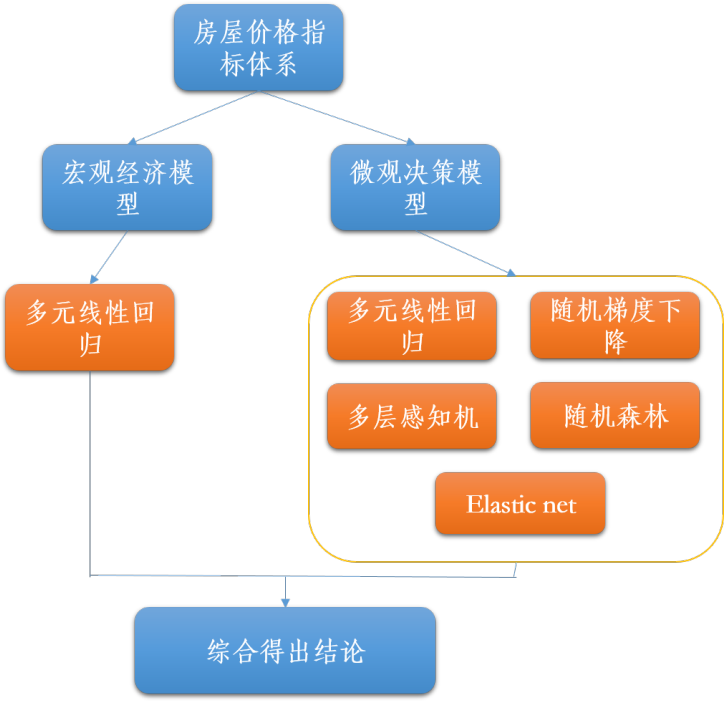


图 **5**　问题二流程图

归分析的方法对因变量，也就是房屋租赁价格指数做回归分析。微观上，我们通过机器学习与神经网络的算法对爬虫得到的租房历史交易数据进行分析，通过在同样的测试集上对比各个方法的拟合优度与均方误差损失，决定最优的模型并对其进行解释性分析。

4.3.1 宏观经济模型

沿袭第一问的结果，我们对使用二级指标对按年平均的房屋租赁价格指数建立多元回归模型。其中各变量数据均为对原始变量作标准化处理以后的变量。对房屋租赁价格指数建立多元回归模型的结果为：

表 **3**　多元线性回归结果

变量	系数	p 值	F 统计量	R^2
常数项	3223.868	0.134	-	-
区域经济	0.636	0.129	-	-
人们生活	-0.075	0.228	-	-
教育水平	-28.014	0.191	-	-
总模型	-	-	33.957	0.943

注：表中标明-的为没有相关指标的。

模型调整 R 方为 0.943，F 统计量为 33.957，拟合优度良，但出现了严重共线性，原因如下：(1) 样本量少，宏观模型是根据 7 年数据得出，数据少导致共线性严重; (2) 三级指标中某些自变量之间相关性很大，如 GDP 与预算支出、居民收入之间相关性很大。

由于有限的样本难以得到精确的模型,我们最后使用有偏估计:$houseprice \quad index = 3223.868 + 0.636area \quad features - 0.075citizen \quad life - 28.014education \quad level$

多元回归的结果说明，区域特征，也就是地区的经济状况越好，房屋的租赁水平一般越高；人民生活水平对总体的房屋租赁价格指数没有太大的影响；教育水平越高的区域，房屋租赁价格就越低。

4.3.2 微观决策模型

微观上，我们将综合对比多个机器学习模型训练之后在测试集上的拟合优度以及损失大小，决定最优的模型并对其进行解释。根据模型的可解释性与数据特征，我们选取了多元线性回归、$Logistic$ 回归、广义线性回归的 $elasticnet$ 族、随机森林以及多层感知机算法。

5.3.2.1 测试集与训练集的划分

在机器学习中，常常通过划分测试集与训练集的方式来判断一个模型的拟合优度。对于分类与聚类问题来说，由于常常是返回一个定性结果，所以可以通过判断 y_{pred} 也就是训练好的模型在测试数据上的输出，与 y，也就是原始因变量之间的相同程度来判断一个模型的拟合优度。在回归问题中，通常通过损失函数，例如均方误差来测试模型的拟合优度。当然，在 $python$ 中也有对回归问题得出拟合优度的函数。

为了保证模型的可比性，我们在训练模型之前将数据先按照一定的比例划分为测试集与训练集，这一步由 $python$ 中的 $train_test_split$ 函数自动执行。

5.3.2.2 数据预处理

为了方便模型的训练，将数据处理为全部数值表示的变量，并且设置哑变量。

表 **4**　变量设置说明

指标	设置规则
出租方式（classes）	1-合租
	0-整租
朝向（direct）	0-东
	1-南
	2-西
	3-北
	4-南北
电梯（elevator）	0-没有电梯
	1-有电梯
地铁（railway）	0-无地铁线
	有地铁线的计算每个房屋有地铁线的条数
地区（place）	0-汉阳
	1-洪山
	2-黄陂
	3-江岸
	4-江汉
	5-江夏
	6-硚口
	7-青山
	8-武昌

5.3.2.3 模型理论介绍

多元线性回归

在机器学习中又称普通最小二乘法，是回归问题中最简单的也是最经典的模型。线性回归有着非常简单的形式，也就是最符合认识常识的 $y = WX + B$。在机器学习中，我们平时所指的线性回归算法，通常是通过最小化损失函数，也就是均方误差，来估计模型的 ω 以及 b。获得了模型的 W 和 B，就相当于获得了模型中的位置参数，那么就可以对数据进行预测。

这里需要说明的是，随着机器学习领域的不断深化，目前估计 w 与 b 的方法越来越多。例如梯度下降的优化算法，常常被用在神经网络中。由于普通最小二乘法是通过求解矩阵来估计参数，所以如果在维数很高的情况下，普通最小二乘计算将非常的慢，甚

至无法求解。但是使用梯度下降方法，则可以较快地收敛到局部最小值。梯度下降方法也有非常多的衍生方法，例如随机梯度下降以及斯坦变分梯度下降算法，我们在后文中将把这种方法应用到我们的模型估计中去。

Elastic net 回归

Elastic net 回归族是一系列的基于广义线性回归上的，对线性回归中 $L1$ 正则项与 $L2$ 正则项做出惩罚，防止模型出现过拟合现象的线性回归。我们这里使用的 Elastic net 回归族主要采用其中的两种比较有效的模型，岭回归与 $Lasso$ 回归。模型中常常出现的问题就是过拟合的情况，尤其是线性回归模型。一般的多元线性回归模型常常能在训练集上表现得很好，但是在测试集上表现得很差，这就是出现了过拟合的情况。就像是平常生活中抓不住重点一样，过拟合的模型往往关照到的特征太多，以至于其难以在总体上表现很好。为了防止过拟合，我们常常在损失函数中加上一个非常大的惩罚项。例如，现在我们有一个模型 $y=\theta_0+\theta_1x+\theta_2x^2+\theta_3x^3+\theta_4x^4$，这是一个过拟合的模型。假设其原始模型是 $y=\theta_0+\theta_1x+\theta_2x^2$，那么我们一定希望，$\theta_3$ 和 θ_4 越小越好，这就能让估计的模型中的 x^3 与 x^4 不起作用。为了让模型在估计的时候把这两个参数估计的很小，我们需要对估计模型时用到的损失函数做一些改变。我们知道，一般的损失函数为：

$min(\frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(x^i)-y^i)^2+\theta_3^2+\theta_4^2])$

要想让两个参数的估计值为很小的数，我们只需要在损失函数的参数前面加上一个很大的乘数算子，就可以让损失函数在收敛到最小的时候，除非 θ_3 与 θ_4 为 0 或者很小的数，否则损失函数就一直都不会收敛。经过改变，新的损失函数为：

$min(\frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(x^i)-y^i)^2+10000\theta_3^2+10000\theta_4^2])$

我们可以把这个很大的数看作是 λ，这就叫为一个惩罚项。惩罚项可以对很多对象做惩罚，岭回归与 Lasso 回归的不同也就在于他们使用的惩罚项不同。岭回归惩罚的是 L2 范数，而 Lasso 回归惩罚的是 L1 范数。这两种方法的选择主要取决于输入的特征的维度。Lasso 回归同时具有变量筛选的功能，所以如果输入的特征的维度很高，也就是样本比较稀疏的情况下，就比较适合 Lasso 回归。

决策树与随机森林

树是一个离散数学中的概念。决策树对于一个问题，通过提出一个有一个的决策分类问题，根据样本训练其判别的准则，得到答案，再连接到下一个问题。如下图所示：

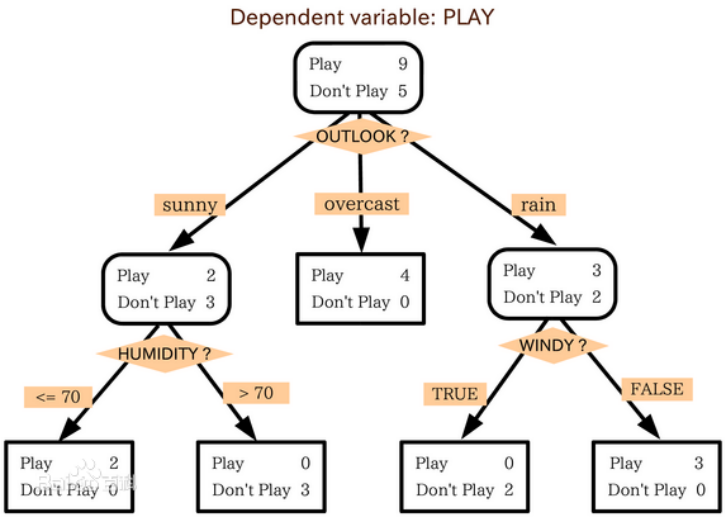


图 **6**　决策树

决策树整体呈离散数学中定义的树状，遍历整个树的分支，都代表了一个决策点，在每个决策点的分支，都代表了这个决策点的取值。每个分支有着约束条件跟权重。输入数值，根据数值的大小，决策树最后可以给出一个决策值。决策树常常被用来做分类问题，但是也可以用来做回归问题，原理是一样的。

随机森林跟决策树很像，它们本质是差不多的。随机森林是一种集成算法，集成算法通常把很多单独的模型聚合到一起进行训练，最后依据投票机制得出结果。集成学习大体可以分成两类，一类是各个个体学习器之间常常有很强的相互关系，需要串行进行学习。另一类则是个体学习器之间是完全独立的，这样的学习器能够支持并行训练，所以速度也会更快一些。同样的, 随机森林也可以做回归，虽然随机森林的构造简单，但是通常也能获得很好的效果。

随机梯度下降

随机梯度下降实际上是一种优化算法，目前被广泛地应用在神经网络模型的参数估计中。随机梯度下降是梯度下降算法的一种变体。梯度下降算法是最简单也是最便捷的，估计参数的算法。梯度下降的思想非常简单，首先定义一个目标函数，通常是最小化的损失函数。这个损失函数中常常包含了需要估计的参数，所以，通过对这样一个非线性规划问题的求解常常就能够收敛到局部最优解。梯度下降也非常的形象，跟其名称一样，梯度下降就是在初始给定点求梯度，减去梯度，这样迭代，不停下降到凸函数的最低值。这些梯度下降算法能够跟机器学习算法结合在一起，估计分布的参数。

随机梯度下降，则是在梯度下降的基础上，每次从样本中抽取一个 minibatch 大小的小批量数据，进行梯度下降，估计参数。这种随机梯度下降能够在大样本、高维度的情况下非常快的收敛到局部最优点。

本文中，我们将随机梯度下降与 $Pytorch$ 中的神经网络相结合，进行预测。

多层感知机

多层感知机是一个非常基础的深度学习算法。深度学习是机器学习的一个领域，它基于人类神经元之间信息的传播，将其与计算机和应用数学相结合，让计算机能够通过与人类大脑类似的神经网络，模仿到人类学习的规律。

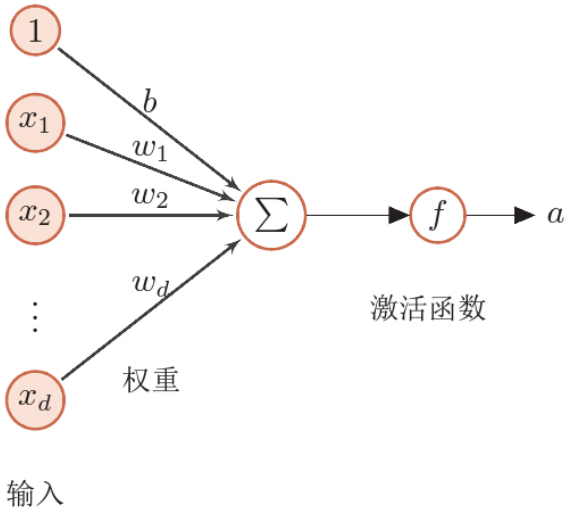


图 **7**　神经网络模型

神经网络算法大致可以分为三种：前馈网络，记忆网络，图网络。多层感知机就算是一种比较经典的前馈神经网络。虽然说学界目前都叫这种神经网络为多层感知机，但是实际上，多层感知机常常是由一串的 *Logistic* 回归模型组成（连续的非线性函数），而不是由多个感知机算法组成。

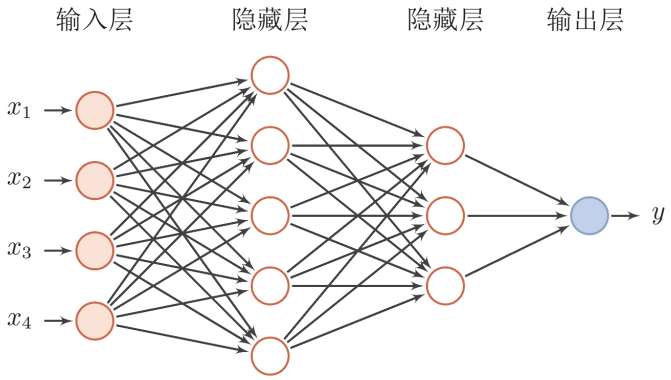


图 **8**　前馈神经网络模型

前馈神经网络中还常常涉及到自动梯度计算与反向传播算法，这里不过多赘述。

5.3.2.4 模型测试结果

我们对所使用的模型计算均方误差与准确度。并选择均方误差最小的模型来解释结果。模型均方误差如下：

表 **5**　模型训练结果

模型	均方误差	训练集准确度	测试集准确度
多元线性回归	15169.49	0.7	0.55
岭回归	12125.56	0.69	0.64
Lasso 回归	11802.36	0.69	0.65
决策树	28213.12	0.89	0.17
随机森林	19671.66	0.85	0.42
随机梯度下降	11550.65	NA	NA
多层感知机	11601.74	0.51	0.66

注：由于随机梯度下降的拟合结果无法得到，所以记为 NA。

我们将这些模型分为三类，作出其在测试集上的拟合结果并与真实值作比较。

广义线性模型

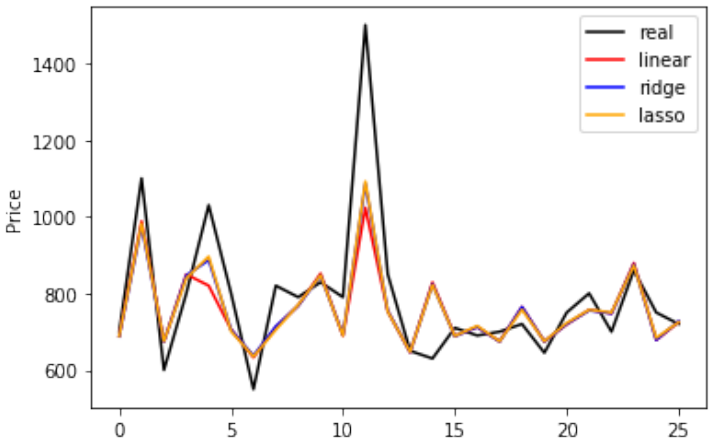


图 **9**　广义线性模型拟合结果

很明显，多元线性模型的拟合结果差于另外两个加了惩罚项的模型。其中，岭回归以及 lasso 回归的拟合结果基本一样。总的来说，广义线性模型的结果并不好。

决策树

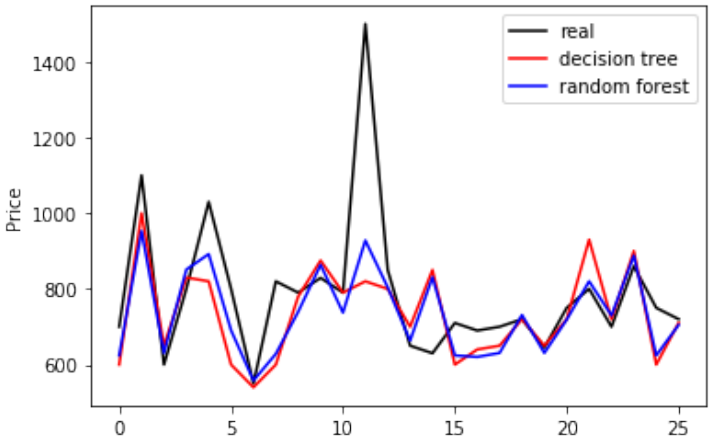


图 **10**　决策树及拟合结果

决策树的拟合结果较差，其主要原因可能是没有能够预测到异常值，其他方面的拟合结果都不错。

神经网络

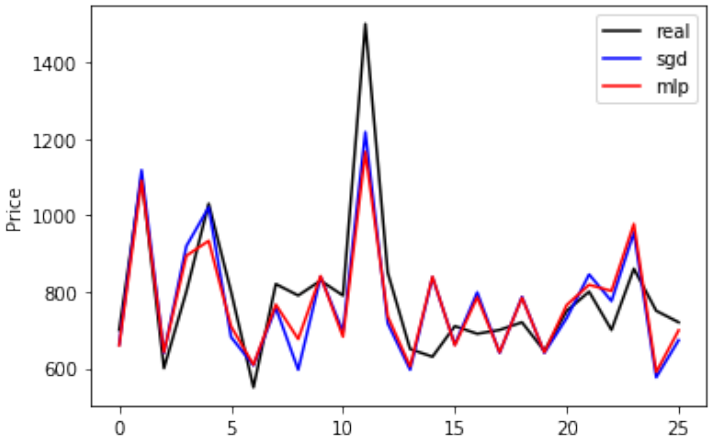


图 **11**　神经网络拟合结果

神经网络的拟合结果相对来说比较好，能够预测到异常值的同时也能够预测到一般值。所以最后选择了神经网络中较优的随机梯度下降模型。

可以看到，多元线性模型以及决策树在训练的时候出现了比较严重的过拟合的情况。在这些模型中，预测效果最好的是随机梯度下降算法，我们选择随机梯度下降算法预测到的模型来进行分析。

5.3.2.5 微观模型结果分析

微观模型最后的得出的结果为：

$$houseprice = 55.71rooms + 13.51halls + 22.11area + 16.63classes + 19.37direct + 14.18elevator + 21.50railway + 43.02place$$

模型中，对房价影响最大的是户型中的房屋数量，其次是房屋的位置，然后是房屋的面积与附近地铁线的条数。房间数量常常代表了出租房屋的生活环境，与房屋的大小有着较高的相关性，所以我们可以近似的认为，对房屋租赁价格影响最大的是房间的大小。房间所处的位置是指该房屋所处的行政区域，常常受到该地区的平均房价、发达程度以及租房需求量的大小的影响，可以看到，洪山区与武昌区地区经济较发达且大学生较密集，所以房屋的价格也较高。房屋附近地铁线的条数直接决定了该房屋附近交通是否便捷，对于大学生来说，出行是否便捷必然是租房时一个非常重要的问题。相对来说，房屋的朝向，出租形式以及是否有电梯则没有那么重要。

4.4 问题三

问题三我们使用了第二问中得出的宏观经济模型对未来五年的房价进行预测，由于数据样本量不大，所以我们同时也选择了 ARMA 模型对时间序列数据做预测。我们综合这两个模型的结果得出最终结论。

4.4.1 宏观经济模型

根据第二问第一部分所求得的宏观经济模型，我们在已有数据的基础上，使用平均增长率对未来五年的三个指标进行测算，并代入模型求得指数的具体数值，结果为：

表 **6**　宏观模型预测结果

	人民生活	区域经济特征	教育水平	指数
2019	5809.784	35370.95	97.31447	1310.414
2020	6418.497	38649.86	97.70047	1436.4
2021	7090.988	42232.73	98.088	1579.722
2022	7833.937	46147.74	98.47706	1742.478
2023	8654.728	50425.67	98.86767	1927.008

这说明，总体上看，房屋租赁价格指数在今后 5 年还将持续上升。由于多元线性回归的模型仍存在不足，我们结合 ARMA 模型做进一步解释。

4.4.2 ARMA 模型

ARMA 模型是一种经典的时间序列模型。他通过检验时间序列的自相关以及偏相关系数，来测定数据的运动过程。时间序列中通常有两种模型。一种是自回归（AR）模

型，他的因变量常常是有 p 阶滞后期决定：

$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + v_t$

其中，ϕ_t 为回归系数，v_t 为白噪声，上式可以被表示为 AR(p)。

另一种则是移动平均过程，他的因变量有白噪声序列决定：

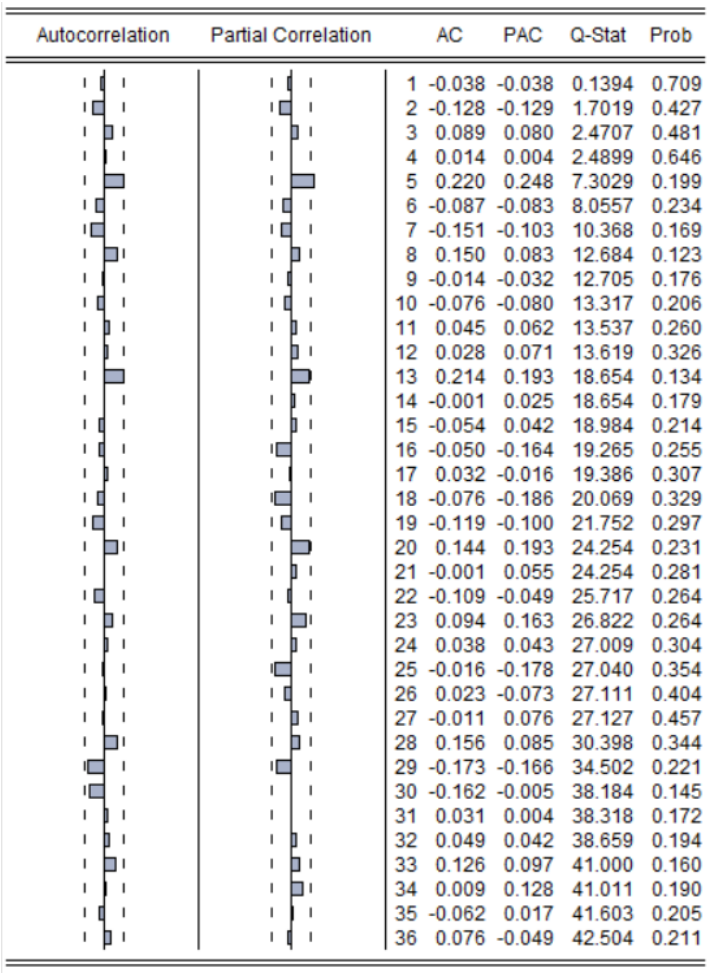
$Y_t = v_t + \theta_1 v_{t-1} + \theta_2 v_{t-2} + ... + \theta_q v_{t-q}$

那么，就称服从上式的过程为 MA(q) 过程。

ARMA 过程，则是综合了上述两种过程的自回归移动平均过程，记为 $ARMA(p,q)$,p 和 q 分别为自回归和移动平均的滞后阶数。通过检验时间序列的自相关和偏相关系数，可以决定合适的 p 与 q 的值，进行回归，做出预测。

首先，要建立时间序列的相关模型，首先要确定因变量是平稳的，可以对时间序列数据做单位根检验，数据本身是不平稳的，但其一阶差分在单位根检验下是显著的，说明其一阶差分平稳。

画出自相关偏相关图：



Autocorrelation	Partial Correlation		AC	PAC	Q-Stat	Prob
		1	-0.038	-0.038	0.1394	0.709
		2	-0.128	-0.129	1.7019	0.427
		3	0.089	0.080	2.4707	0.481
		4	0.014	0.004	2.4899	0.646
		5	0.220	0.248	7.3029	0.199
		6	-0.087	-0.083	8.0557	0.234
		7	-0.151	-0.103	10.368	0.169
		8	0.150	0.083	12.684	0.123
		9	-0.014	-0.032	12.705	0.176
		10	-0.076	-0.080	13.317	0.206
		11	0.045	0.062	13.537	0.260
		12	0.028	0.071	13.619	0.326
		13	0.214	0.193	18.654	0.134
		14	-0.001	0.025	18.654	0.179
		15	-0.054	0.042	18.984	0.214
		16	-0.050	-0.164	19.265	0.255
		17	0.032	-0.016	19.386	0.307
		18	-0.076	-0.186	20.069	0.329
		19	-0.119	-0.100	21.752	0.297
		20	0.144	0.193	24.254	0.231
		21	-0.001	0.055	24.254	0.281
		22	-0.109	-0.049	25.717	0.264
		23	0.094	0.163	26.822	0.264
		24	0.038	0.043	27.009	0.304
		25	-0.016	-0.178	27.040	0.354
		26	0.023	-0.073	27.111	0.404
		27	-0.011	0.076	27.127	0.457
		28	0.156	0.085	30.398	0.344
		29	-0.173	-0.166	34.502	0.221
		30	-0.162	-0.005	38.184	0.145
		31	0.031	0.004	38.318	0.172
		32	0.049	0.042	38.659	0.194
		33	0.126	0.097	41.000	0.160
		34	0.009	0.128	41.011	0.190
		35	-0.062	0.017	41.603	0.205
		36	0.076	-0.049	42.504	0.211

图 **12**　一阶差分自相关与偏相关检验图

可以看到，5 期滞后期与 13 期以及 20 期滞后期显著，检验 AR(5)，AR(13)，AR(20)

与 AR(5)AR(13)AR(20) 这几种回归的显著性，并使用 AIC 准则选取模型。

检验结果如表：

表 **7**　时间序列模型检验结果

模型	自变量	系数估计值	P 值	AIC
AR(5)	AR(5)	0.2531	0.0161**	8.705
AR(13)	AR(13)	0.2549	0.0186**	8.749
AR(20)	AR(20)	0.1797	0.1014	8.775
AR(5)AR(13)AR(20)	AR(5)	0.2263	0.0401**	8.685
	AR(13)	0.2252	0.0389**	
	AR(20)	0.2092	0.0458**	

根据 AIC 准则，选择 AIC 值最小的模型，也就是 AR(5)AR(13)AR(20) 模型。检查模型 P 值，参数显著。

拟合的模型如下：

$houseprice_t - houseprice_{t-1} = 0.2263ar(5) + 0.2252ar(13) + 0.2092ar(20)$

用模型在已知的样本上进行测试，与真实数据进行比较：

图 **13**　原始房屋租赁价格指数走势图

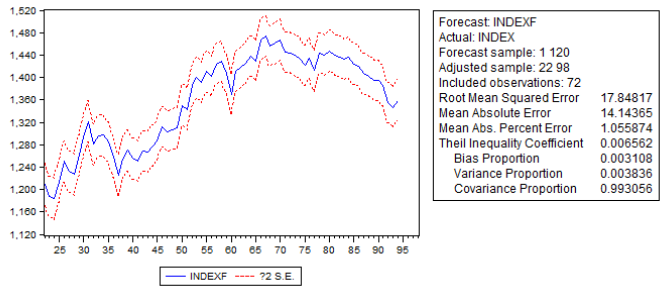


图 **14**　已知样本上预测房屋租赁价格指数走势图

可以看到，预测的效果还是非常不错的。在此基础上，我们对下一年的数据进行预测，作图如下 (数据见附件)：

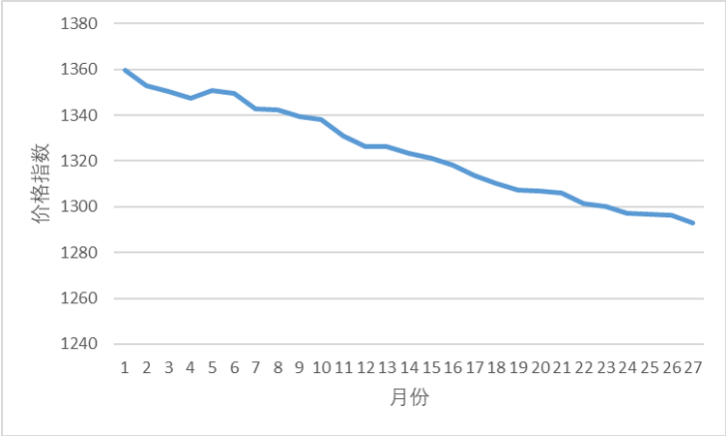


图 **15**　预测房屋租赁价格指数走势图

我们发现使用 ARMA 模型所做的预测只能在一段小区间内预测房价的波动，因为 ARMA 模型的预测较大程度上取决于滞后期，如果滞后期的数据都来自预测值，那么整个时间序列数据都会依照已知样本数据的最后一段趋势预测，所以可能无法预测整体值。

根据 ARMA 模型的月度预测结果，我们认为武汉市房屋租赁价格指数在未来一年至两年内很可能会继续下降。房屋租赁价格指数下降，说明租房价格在前几年的飙升过程中受到了控制，租房市场的泡沫现象得以缓解。

综合两种模型的预测结果，我们认为，武汉市的房屋租赁价格将在近一年或两年内，短期地保持下降趋势，但是不久就会上升，总体上呈上升趋势。

五、 模型不足

本文中所做的模型仍然存在着一些不足的地方。由于数据难以获得，问题一种所建立的指标体系，可能涵盖的指标并不完全。由于数据年份较少，问题二中我们选择了用

自由度较低的二级指标建立模型，这样的选择可能存在偏差。并且，也由于样本量较少的原因，模型出现了多重共线性的问题。后期，我们将继续针对这些问题对模型作出进一步的优化。

参考文献

[1] 方毅、赵石磊. 房屋销售价格和租赁价格的关系研究 [J]. 数理统计与管理, 2007 (11) 。

[2] 汪业辉. 住房租赁市场租金的影响因素分析 [D]. 武汉: 华中科技大学, 2014.

[3] 余华义, 陈东. 我国地价、房价和房租关系的重新考察: 理论假设与实证检验 [J]. 上海经济研究,2009(04):11-21.

[4] 赵志强, 赵予新. 居民消费对房地产价格影响的实证研究——基于我国 2002—2013 年的面板数据分析 [J]. 金融经济,2015(14):208-209.

[5] 郑文娟. 中国城市住房价格与住房租金的影响因素及相互关系研究 [D]. 浙江大学,2011.

[6] 周志华，机器学习，2015.

[7] 邱锡鹏，神经网络与深度学习，https://nndl.github.io/，2019.

附录 A　房屋租赁价格指标体系

表 **8**　房屋租赁价格指标体系

总指标	一级指标	权重	二级指标	权重	三级指标	权重
房屋租赁价格指数	宏观经济特征	0.5	区域经济特征	0.4298	地区生产总值	0.2615
					人口总数	0.2404
					预算支出	0.2596
					房地产投资额	0.2385
			人民生活	0.4281	城市居民储蓄余额	0.3379
					收入	0.3369
					就业率	0.3252
			教育水平	0.1421	普通本专科人数	1
	房屋个体特征	0.5	房屋基本信息	0.5	房间数	0.1667
					厅数	0.1667
					总面积	0.1667
					出租类型	0.1667
					朝向	0.1667
					是否有电梯	0.1667
			区位特征	0.5	地铁线	0.5
					所属行政区	0.5

附录 B　武汉市租房价格爬虫 A——安居客

```
#encoding=UTF-8
```

```
#author:Yihan

import numpy as np
import pandas as pd
import bs4
from lxml import etree
import requests
import time
import os
import json
import base64
import re
import io
from io import BytesIO
from fontTools.ttLib import TTFont

root_url = 'https://wh.zu.anjuke.com/?from=navigation'
headers = {
    'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
        Chrome/76.0.3809.87 Safari/537.36'
}

house_place,house_name,house_rooms,house_halls,house_area,house_addrs,house_classes,house_direct,house_elevator
    ##爬取租房信息
def search_house(address_url):
    response = requests.get(address_url,headers = headers,timeout=30).text
    #############转码处理，很重要！###############
    base64_str = re.findall('data:application/font-ttf;charset=utf-8;base64,(.*)\'\)
        format\(\'truetype\'\)}', response)

    bin_data = base64.b64decode(base64_str[0])
    fonts = TTFont(io.BytesIO(bin_data))
    bestcmap = fonts.getBestCmap()
    newmap = {}
    for key in bestcmap.keys():
        print(key)
        print(re.findall(r'(\d+)',bestcmap[key]))
        value = int(re.findall(r'(\d+)',bestcmap[key])[0])-1
        key = hex(key)
        newmap[key] = value
    response_ = response
    for key,value in newmap.items():
        key_ = key.replace('0x','&#x')+';'
        if key_ in response_:
            response_ = response_.replace(key_,str(value))
    ##############转码结束###############
    html = etree.HTML(response_)
```

```
house_list = html.xpath('//div[@id = "list-content"]/div')
for item in house_list[2:]:
    try:
        name = item.xpath('./div[@class="zu-info"]/h3/a/b/text()')
        if name:
            house_name.append(name[0])
        else:
            house_name.append('NA')
        rooms = item.xpath('./div[@class="zu-info"]/p/b[1]/text()')
        if rooms:
            house_rooms.append(int(rooms[0]))
        else:
            house_rooms.append('NA')
        halls = item.xpath('./div[@class="zu-info"]/p/b[2]/text()')
        if halls:
            house_halls.append(int(halls[0]))
        else:
            house_rooms.append('NA')
        area = item.xpath('./div[@class="zu-info"]/p/b[3]/text()')
        if area:
            house_area.append(int(area[0]))
        else:
            house_area.append('NA')
        addrs = item.xpath('./div[@class="zu-info"]/address/a/text()')
        if addrs:
            house_addrs.append(addrs[0])
        else:
            house_addrs.append('NA')
        classes = item.xpath('./div[@class="zu-info"]/p[2]/span[1]/text()')
        if classes:
            house_classes.append(classes[0])
        else:
            house_classes.append('NA')
        direct = item.xpath('./div[@class="zu-info"]/p[2]/span[2]/text()')
        if direct:
            house_direct.append(direct[0])
        else:
            house_direct.append('NA')
        elevator = item.xpath('./div[@class="zu-info"]/p[2]/span[3]/text()')
        if elevator:
            house_elevator.append(elevator[0])
        else:
            house_elevator.append('NA')
        railway = item.xpath('./div[@class="zu-info"]/p[2]/span[4]/text()')
        if railway:
            house_railway.append(railway[0])
        else:
```

```
            house_railway.append('NA')
        price = item.xpath('./div[@class="zu-side"]//b/text()')
        place = item.xpath('./div[@class="zu-info"]/address/text()')[1]
        places = re.findall(u"[\u4e00-\u9fa5]+",place)[0]
        if places:
            house_place.append(places)
        else:
            house_place.append('NA')
        if price:
            house_price.append(int(price[0]))
        else:
            house_price.append('NA')
    except:
        pass

if __name__ == '__main__':
    for page in range(5):
        url = root_url + '/p' + str(page)
        search_house(url)
        time.sleep(1)

##########数据输出保存##########
import pandas as pd
from pandas.core.frame import DataFrame
infodict = {
    'name':house_name,
    'rooms':house_rooms,
    'halls':house_halls,
    'area':house_area,
    'addrs':house_addrs,
    'classes':house_classes,
    'direct':house_direct,
    'elevator':house_elevator,
    'railway':house_railway,
    'place':house_place
    'price':house_price
}
info_data=DataFrame(infodict)
info_data.to_csv('C:/Users/Administrator/Desktop/houseinfo.csv',encoding='gbk')
```

附录 C　武汉市租房价格爬虫 B——链家

```
import requests
import re
from lxml import etree
```

```
page = 1
Info = [[],[],[],[]]
Price = []
#n
root_url = 'https://lianjia.com'
while page < #n:
    time.sleep(1)
    request_url = root_url + '/zufang/pg'+str(page)+'/#contentList'
    response = requests.get(request_url,headers=headers)
    html = response.content.decode('utf-8')
    selector = etree.HTML(html)
    list = selector.xpath('//div[@class="content__list--item"]')
    for item in list:
        title = item.xpath('./div/p[1]/a/text()')[0].strip()
        price = item.xpath('./div/span/em/text()')[0].strip()
        price = price.split('-')[0]
        place = item.xpath('./div/p[2]/a[1]/text()')[0].strip()
        area = item.xpath('./div/p[2]/text()')
        area = ''.join(area).replace('\n','')
        room = re.match(r'.(\d{1,4})㎡. (\d{1,2})室(\d{1,1})厅(\d{1,1})卫', area)
        height1 = item.xpath('./div/p[2]/span/text()')[1]
        height_search = re.search(r'\d{1,4}',height1)
        if height_search:
            height = int(height_search.group(0))
        else:
            height = 'NA'
        if room !=None:
            Info[0].append(int(room.group(1)))#面积
            Info[1].append(int(room.group(2)))#卧室数量
            Info[2].append(place)#地点
            Info[3].append(height)#楼层
            Price.append(price)
    page = page+1
```

附录 D　房屋租赁价格机器学习模型建模 python 代码

```
#!/usr/bin/env python
# coding: utf-8

# # Wuhan House Prices Prediction
# ## author: Yihan
# ----
# ## Package Loaded
```

```
# In[209]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import mglearn
import torch# needed for mlp
import sklearn
import sys
import IPython
from IPython.display import display
from sklearn.model_selection import train_test_split
import torch.nn as nn


# ## Data preprocessing

# In[151]:


X = pd.read_csv('houseinfo.csv')
y = pd.read_csv('fakeprice.csv')
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0)
print('X_train shape:{}'.format(X_train.shape))
print('X_test shape:{}'.format(X_test.shape))
print('y_train shape:{}'.format(y_train.shape))
print('y_test shape:{}'.format(y_test.shape))


# ## Linear Regression

# In[316]:


from sklearn.linear_model import LinearRegression
lr_fit = LinearRegression().fit(X_train,y_train)
print('lr_fit coef:{}'.format(lr_fit.coef_))
print('lr_fit intercept:{}'.format(lr_fit.intercept_))
print('Training set score:{:.2f}'.format(lr_fit.score(X_train,y_train)))
print('Testing set score:{:.2f}'.format(lr_fit.score(X_test,y_test)))
y_pred_linear = lr_fit.predict(X_test)
mse1 = np.sum((y_test-lr_fit.predict(X_test))**2)
mse1/26


# ## Ridge Regression
```

```
# In[317]:


from sklearn.linear_model import Ridge

ridge = Ridge().fit(X_train,y_train)
print('Training set score:{}'.format(ridge.score(X_train,y_train)))
print('Test set score:{}'.format(ridge.score(X_test,y_test)))
y_pred_ridge = ridge.predict(X_test)
mse2 = np.sum((y_test-ridge.predict(X_test))**2)
mse2/26


# ## Lasso Regression

# In[318]:


from sklearn.linear_model import Lasso
lasso = Lasso().fit(X_train,y_train)
print('Training set score:{:.2f}'.format(lasso.score(X_train,y_train)))
print('Test set score:{:.2f}'.format(lasso.score(X_test,y_test)))
y_pred_lasso = lasso.predict(X_test)
mse3 = np.sum((y_test-np.array(DataFrame(lasso.predict(X_test))))**2)
print('Number of features used:{}'.format(np.sum(lasso.coef_!=0)))
mse3/26


# ## Decision Tree Regressor

# In[319]:


from sklearn.tree import DecisionTreeRegressor
tree = DecisionTreeRegressor().fit(X_train,y_train)
print('Training set score:{:.2f}'.format(tree.score(X_train,y_train)))
print('Test set score:{:.2f}'.format(tree.score(X_test,y_test)))
y_pred_dtr = tree.predict(X_test)
mse4 = np.sum((y_test-np.array(DataFrame(tree.predict(X_test))))**2)
mse4/26


# ## Random Forests

# In[320]:
```

```
from sklearn.ensemble import RandomForestRegressor
rantree = RandomForestRegressor().fit(X_train,y_train)
print('Training set score:{:.2f}'.format(rantree.score(X_train,y_train)))
print('Test set score:{:.2f}'.format(rantree.score(X_test,y_test)))
y_pred_rf = rantree.predict(X_test)
mse5 = np.sum((y_test-np.array(DataFrame(rantree.predict(X_test))))**2)
mse5/26


# ## Multilayer Perceptron

# In[322]:


X_train = torch.tensor(np.array(X_train),dtype = torch.float)
X_test = torch.tensor(np.array(X_test),dtype=torch.float)
y_train = torch.tensor(np.array(y_train),dtype=torch.float)
y_test = torch.tensor(np.array(y_test),dtype=torch.float)

import torch.utils.data as Data
batch_size = 4
dataset = Data.TensorDataset(X_train,y_train)
data_iter = Data.DataLoader(dataset,batch_size,shuffle=True)

class LinearNet(nn.Module):
    def __init__(self,n_feature):
        super(LinearNet, self).__init__()
        self.linear = nn.Linear(n_feature,1)
    def forward(self,x):
        y = self.linear(x)
        return y

num_inputs = 8
num_examples = len(y_train)
net = LinearNet(num_inputs)

from torch.nn import init
init.normal_(net.linear.weight,mean=0,std=0.01)
init.constant_(net.linear.bias,val=0)

loss = nn.MSELoss()

import torch.optim as optim
optimizer = optim.SGD(net.parameters(),lr=0.001)
```

```
num_epochs=10
for epoch in range(1,num_epochs+1):
    for X,y in data_iter:
        output = net(X_train)
        l = loss(output,y_train.view(-1,1))
        optimizer.zero_grad()
        l.backward()
        optimizer.step()
    print('epoch %d, loss: %f'%(epoch,l.item()))

dense = net.linear
print(dense.weight)

print(torch.sum((net(X_test)-y_test)**2)/26)

from torch.autograd import Variable
result = Variable(net(X_test))

y_pred_sgd = result.data.numpy()


# In[333]:


X_train


# In[256]:




num_inputs,num_outputs,num_hiddens = 8,4,8
W1 = torch.tensor(np.random.normal(0,0.01,(num_inputs,num_hiddens)),dtype=torch.float)
b1 = torch.tensor(num_hiddens,dtype=torch.float)
W2 = torch.tensor(np.random.normal(0,0.01,(num_hiddens,num_outputs)),dtype=torch.float)
b2 = torch.tensor(num_outputs,dtype = torch.float)

params = [W1,b1,W2,b2]
for param in params:
    param.requires_grad_(requires_grad=True)

def relu(X):
    return torch.max(input=X,other=torch.tensor(0.0))

def net(X):
    X = X.view((-1,num_inputs))
    H = relu(torch.matmul(X,W1)+b1)
```

```
    return torch.matmul(H,W2)+b2

loss = nn.MSELoss()

def evaluate_accuracy(data_iter,net):
    acc_sum,n=0.0,0
    for X,y in data_iter:
        acc_sum += ((net(X).argmax(dim=1)-y)**2<200).float().sum().item()
        n += y.shape[0]
    return acc_sum/n

def train_model(net,train_iter,test_iter,loss,num_epochs,batch_size,params = None,lr =
     None,optimizer=None):
    for epoch in range(num_epochs):
        train_l_sum,train_acc_sum,n=0.0,0.0,0
        for X,y in train_iter:
            y_hat = net(X)
            l = loss(y_hat,y).sum()

            if optimizer is not None:
                optimizer.zero_grad()
            elif params is not None and params[0].grad is not None:
                for param in params:
                    param.grad.data.zero_()

            l.backward()
            if optimizer is None:
                sgd(params,lr,batch_size)
            else:
                optimizer.step()


            train_l_sum += l.item()
            train_acc_sum += ((y_hat.argmax(dim=1)-y)**2<200).sum().item()
            n += y.shape[0]
        test_acc = evaluate_accuracy(test_iter,net)
        print('epoch %d, loss %.4f, train acc %.3f, test acc %.3f'
              % (epoch + 1, train_l_sum / n, train_acc_sum / n, test_acc))

def sgd(params,lr,batch_size):
    for param in params:
        param.data -= lr*param.grad

batch_size = 4
trainset = Data.TensorDataset(X_train,y_train)
train_iter = Data.DataLoader(trainset,batch_size,shuffle=True)
testset = Data.TensorDataset(X_test,y_test)
```

```
test_iter = Data.DataLoader(testset,batch_size,shuffle=True)
loss = torch.nn.MSELoss()

lr=0.03
num_epochs = 5

train_model(net,train_iter,test_iter,loss,num_epochs,batch_size,params,lr)


# In[324]:


X_train = np.array(X_train)
y_train = np.array(y_train)
X_test = np.array(X_test)
y_test = np.array(y_test)

from sklearn.neural_network import MLPRegressor
mlp = MLPRegressor(
    hidden_layer_sizes = (8,8),activation='relu',solver='adam',alpha=0.003,batch_size='auto',
    learning_rate='constant',learning_rate_init=0.001,power_t=0.5,max_iter=5000,shuffle=True,
    random_state=1,verbose=False,warm_start=False,epsilon=10)
mlp.fit(X_train,y_train)


# In[326]:


print(mlp.score(X_train,y_train))
print(mlp.score(X_test,y_test))
y_pred_mlp = mlp.predict(X_test)
mse7 = np.sum((y_test-np.array(DataFrame(mlp.predict(X_test))))**2)
mse7/26


# In[329]:


plt.plot(y_test,color='black',label='real')
plt.plot(y_pred_linear,color = 'red',label='linear')
plt.plot(y_pred_ridge,color = 'blue',label='ridge')
plt.plot(y_pred_lasso,color='orange',label='lasso')
plt.ylabel('Price')
plt.legend(loc=1)


# In[330]:
```

```
plt.plot(y_test,color='black',label='real')
plt.plot(y_pred_dtr,color='red',label='decision tree')
plt.plot(y_pred_rf,color='blue',label='random forest')
plt.legend(loc=1)
plt.ylabel('Price')


# In[331]:


plt.plot(y_test,color='black',label='real')
plt.plot(y_pred_sgd,color='blue',label='sgd')
plt.plot(y_pred_mlp,color='red',label='mlp')
plt.legend(loc=1)
plt.ylabel('Price')


# In[314]:


import graphviz
mglearn.plots.plot_animal_tree()
```