中国人民大学

课程报告

（2021-2022 学年秋季学期）

论文题目：经典线性回归模型对房价的预测及改进

课程名称：统计基础

任课老师：代文林、李赛

班　　级：应用统计专硕班

学　　号：2021103835

姓　　名：崔博涵

经典线性回归模型对房价的预测及改进

摘　要

随着年末中国经济年会的召开,“房地产行业是国家的支柱产业”这一命题再次走进公众的视野。其实，无论房地产话题的火热与否，看房-选房-买房基本上成为当代人的必经之路，房价预测模型的准确度与信服力的提升一直是回归研究领域内的一个热门板块。本文选取 2012-2013 年台湾省台北市及其周边的一组房价数据，经初步探索性分析判断该数据集近似符合线性回归模型的基本假设，将清洗、归一化处理后的数据集，输入经典的回归模型：多元线性回归、岭回归、稳健回归、度为 2、3 的多项式回归模型。通过比较可决系数、预测标准差、综合评价指标 Evalue 检验模型的优劣性，发现随着模型复杂度的提高，在训练集上的拟合性越好，而在预测集上的误差就越大。为在拟合优度与预测准度上寻求平衡，本文调用 PolynomialFeatures() 函数构建 degree=3 的解释变量组合全集，以拟合优度检验系数为依据通过逐步回归选择最优的自变量组合，经验证相较一般线性模型，本复合模型准确度上提高了 18.5%，综合指标也在所有模型中表现最优，可用于样本量 >> 自变量数目的房价数据集的预测。从不同角度考虑，本模型可为开发商选址定价、大众买房居住或投资提供决策参考。

关键词：岭回归；稳健回归；逐步线性回归；拟合优度；预测标准差；Evalue；

目　　录

1 选题背景

1.1 房地产行业-中国的支柱产业

2021 年 12 月 11 日召开的“中国经济年会”上，“房地产行业”被列入国家支柱产业，中央财经委员会办公室副主任韩文秀表示：“要促进房地产业健康发展。房地产业规模大、链条长、牵涉面广，在国民经济、全社会固定资产投资、地方财政收入、金融机构贷款总额中都占有相当高的份额，对于经济金融稳定和风险防范具有重要的系统性影响。”中央经济工作会议内容和权威人士的表态，说明房地产行业基本面发生了重要变化和好转。“因城施策促进房地产业良性循环和健康发展”是我国最近对房地产行业的最新提法，而扩大居民内需，拉动房地产消费，促进其良性循环发展成为了该行业的下一阶段发展目标。随着国家对房地产政策的进一步更新，作为房地产行业不得不提的房价走势问题即将重新进入国民关注热点排行榜。从微观层面来看，房价问题关系到个人的居住、成家、教育、投资等。从宏观层面来看，房价问题关系到国家的经济增长、金融系统的稳定等。本文基于以上背景，以台湾省某年的房价数据为例，探讨关于房价预测的统计模型及其优化。[1]

1.2 数据集选取及简介

本文从 https://archive.ics.uci.edu/ml/datasets.php 上选择 2012-2013 年台湾省台北市及其周边的一组房价数据，数据集包含 414 条房价信息，每条信息由“交易日期”、“房龄”、“到 MRT 站（地铁站）的距离”、“便利店数目”、“纬度”、“经度”、“单位平方房价”7 个因子组成，数据集具体信息及 head 如下图所示。

Data Set Characteristics:	Multivariate	**Number of Instances:**	414	**Area:**	Business
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	7	**Date Donated**	2018-08-18
Associated Tasks:	Regression	**Missing Values?**	N/A	**Number of Web Hits:**	131765

图 1 数据集具体信息

	transaction date	house age	dis to MRT	number of convenience stores	latitude	longitude	house price of unit area
0	2012.916667	32.0	84.87882	10	24.98298	121.54024	37.9
1	2012.916667	19.5	306.59470	9	24.98034	121.53951	42.2
2	2013.583333	13.3	561.98450	5	24.98746	121.54391	47.3
3	2013.500000	13.3	561.98450	5	24.98746	121.54391	54.8
4	2012.833333	5.0	390.56840	5	24.97937	121.54245	43.1

图 2 数据集前五行

本次选取数据集被用在【Real estate valuation data set Data Set】一文中，该文将逐步回归方法应用到传统的房地产估价准则中建立房产估价模型，经实验具有较高的估价准确率。

1.3 模型创新

本文以最简单的多元线性回归为出发点，创建拟合数据集的多个回归模型，通过拟合优度检验选择最佳回归预测模型，并通过统计学理论进一步修正模型，使得拟合与预测效果达到最优，与其他线性回归模型相比，本模型的创新之处有以下两点。

(1) 基于自变量数目相对样本量较少的数据集，一般线性回归模型对因变量的可解释性不够，通过本模型重构自变量集达到进一步挖掘自变量的组合对因变量的影响，从而在原有数据的基础上，进一步扩充房价的影响因子体系。

(2) 相较于其他单一的回归模型，本模型在拟合优度与预测准确度上有较好的平衡，在不出现过拟合现象的前提下最大限度地拟合数据集。

2 数据预处理与可视化

为减小后续模型建立过程中的误差，首先对数据进行探索性分析。选择“交易日期”、“房龄”、“到 MRT 站（地铁站）的距离”、“便利店数目”、“纬度”、“经度”6 个指标作为解释变量，其中，“交易日期”、“到 MRT 站（地铁站）的距离”、“纬度”、“经度”为完备型连续数据，“房龄”、“便利店数目”为整数型连续数据；显而易见，“单位平方房价”作为本模型的因变量。

2.1 描述统计

调用 pandas 包中的 describe() 函数对原始数据集中的各列数据进行描述性统计分析，得到结果如图 3，主要分析指标是均值、标准差、最小最大值、中位数。从各变量的标准差角度分析，“房龄”,“到地铁站的距离”,“单位平方房价”这三个解释变量存在较大的标准差，其中，房龄的最小值为“0”，到地铁站距离的极差高达 6460m，单位房价的的最大值为 117.5 万台币每坪，换算成常用计量单位约为 8.2 万人民币每平米，而单位房价的下分位数也仅仅是 3 万人民币每平，综上初步判断三列数据存在异常值或分布较极端的情况，以下将通过数据可视化做进一步判断。

	transaction date	house age	dis to MRT	number of convenience stores	latitude	longitude	house price of unit area
count	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000
mean	2013.148953	17.712560	1083.885689	4.094203	24.969030	121.533361	37.980193
std	0.281995	11.392485	1262.109595	2.945562	0.012410	0.015347	13.606488
min	2012.666667	0.000000	23.382840	0.000000	24.932070	121.473530	7.600000
25%	2012.916667	9.025000	289.324800	1.000000	24.963000	121.528085	27.700000
50%	2013.166667	16.100000	492.231300	4.000000	24.971100	121.538630	38.450000
75%	2013.416667	28.150000	1454.279000	6.000000	24.977455	121.543305	46.600000
max	2013.583333	43.800000	6488.021000	10.000000	25.014590	121.566270	117.500000

图 3 对原始数据集各列进行描述性统计

2.2 数据可视化

为进一步研究原始数据集上解释变量和因变量的分布特征，根据数据类别与大小调用合适的图表依次对各变量进行数据可视化。

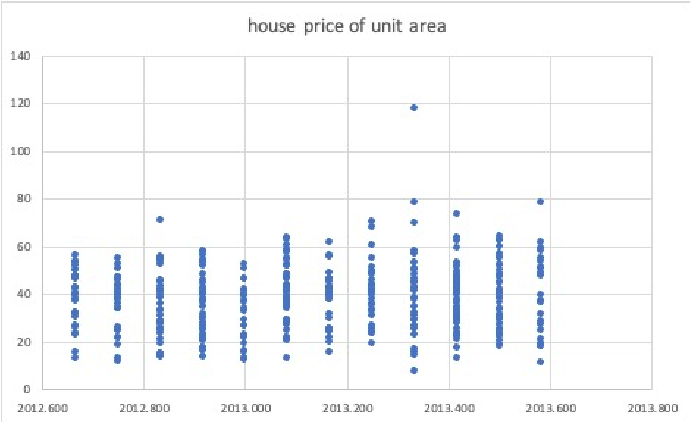


图 4 房价在各交易时间段的分布散点图

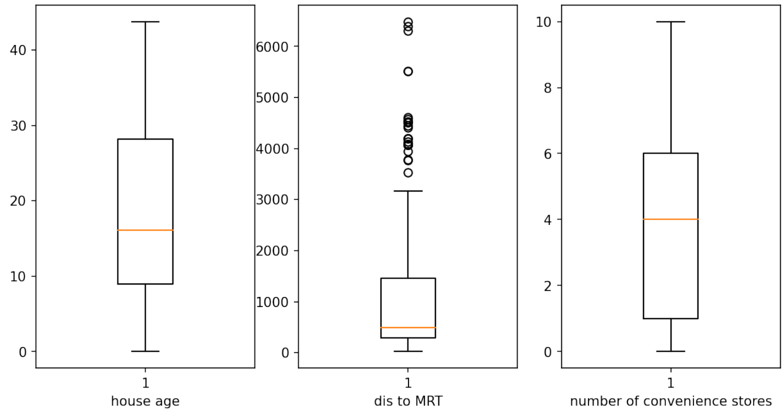


图 5 基于房龄，距离，便利店个数的箱线图

调用 geopandas 包绘制台北与新台北的经纬地图，可视化房价与地理位置的关系。

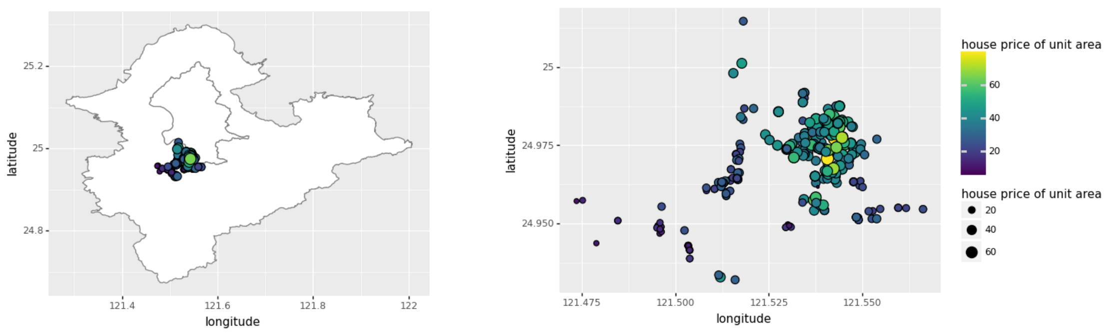
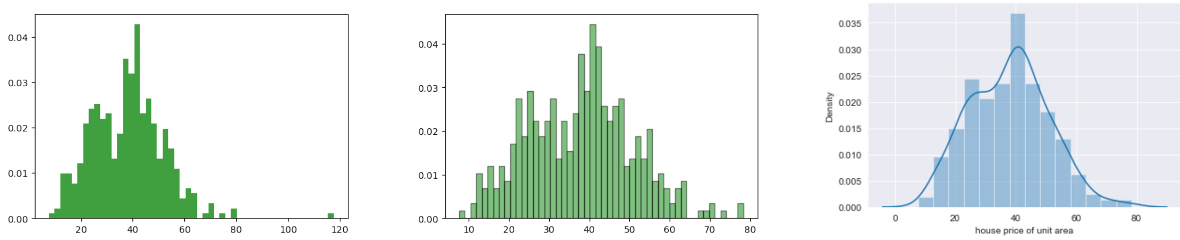


图 6 房价的地理位置分布

由图 4 看出，房价在 2012-2013 年间整体趋于稳定，但在纵轴“120”附近出现一个及其异常的数据点，也可以从图 7 的（a）中得出同一结论，经验证，该条数据为第 270 条数据，以下研究选择删除该条数据；由图 5 得，“到地铁站的距离”这一变量列，存在大量的极端值（>3000 共计 60 条），另外一方面考虑到偏远的地铁站影响人们的购房决策，如若删除这些极端值会降低模型的准确率与可解释性，以下研究通过归一化方法降低该列数据的量级，减小极端值带来的影响；图 6 表示本次



(a) 120 左侧有异常数据 (b) box=50 (c) 概率密度近似

图 7 单位房价分布

所研究的房价数据主要来源于台北市和新北市交接的地方，地理分布相对集中，在越靠近台北市区的地方，浅色点越多，说明房价普遍较高；从图 7 中可以看出因变量（单位房价）的分布近似呈现正态性，满足线性回归的基本假设之一。综上述，该数据集不存在较大噪声，随机项满足近似正态分布，较为适用于回归模型的构建。

2.3 数据归一化

删除第 270 条房价数据，调用 Sklearn.Preprocess 包中的 MinMaxScaler() 函数对自变量进行归一化处理，将自变量的范围限制在 [0,1] 区间内，由于经归一化后的解释变量相对因变量的量级较小，需要对因变量进行取对数操作，再次检查得到的数据集是否存在缺失值，最终得到 412 条处理过的“可行数据集。”

表 1 变量符号表示与归一化对数化处理

变量名称	符号表示	归一化或对数化
交易日期	x_1	$x_{1j} - min(x_1)/max(x_i) - min(x_1)$
房龄	x_2	$x_{2j} - min(x_2)/max(x_2) - min(2_i)$
到地铁站的距离	x_3	$x_{3j} - min(x_3)/max(x_3) - min(x_3)$
周围便利店数目	x_4	$x_{4j} - min(x_4)/max(x_4) - min(x_4)$
纬度	x_5	$x_{5j} - min(x_5)/max(x_5) - min(x_5)$
经度	x_6	$x_{6j} - min(x_6)/max(x_6) - min(x_6)$
单位房价	Y	$log(Y)$

3 模型建立

3.1 模型假设

本报告主要研究对象是线性回归模型，在模型建立前，对数据集与参数做出如下基本假设：

表 2 模型的基本假设

假设一	各解释变量 x_i 之间没有完全的线性关系
假设二	误差项两两不相关
假设三	误差项关于解释变量的条件期望为 0
假设四	误差项服从同一正态分布（同方差）
假设五	解释变量与误差项不相关
假设六	回归模型被正确假定

3.2 模型的建立

3.2.1 线性模型

不妨设观测值 $Y=(y_1,...,y_n)$，n*p 维设计矩阵 X，未知参数 $w=(w_1,...,w_p)$，在本数据集中 $p=7$，建立回归模型如下：

$$\hat{Y}=Xw \tag{1}$$

（1）多元线性回归模型

一般多元线性回归通过最小二乘法求解下列无约束优化问题：

$$\min_{w}\|Xw-Y\|_2^2 \tag{2}$$

（2）岭回归 (Ridge Regression)

Lasso 回归，岭回归，弹性回归 (Elastic Regression) 分别通过引入正则项降低模型的复杂度，从而解决线性回归中常见的过拟合问题。由于 Lasso 回归与弹性回归中包含一阶正则项，很容易出现参数为 0 的现象，更适用于样本量 « 解释变量个数的情况，本次实验不予考虑这两种模型，岭回归的优化原理如下，本质上相当于求解约束优化问题：

$$\min_{w}\|Xw-y\|_2^2+\alpha\|w\|_2^2 \tag{3}$$

本次实验中，通过交叉验证法选择岭回归模型中的超参数 α。

（3）稳健回归 (Robust Regression)

调用 sklearn 中的 $TheilSenRegressor()$ 函数实现稳健回归。稳健回归是最小二乘的一种替代方法，当数据中有 influential observations 等异常值的存在时，最小二乘法不再适用，在第二节中通过探索性分析可知，X3 列存在较多极端值，但这些极端值并不是由于数据输入错误或者样本选取异常等人为原因引起时，不能随意把这些值排除在分析之外。这时选用稳健回归分析，在一般最小二乘法的基础上根据残差绝对值对观测值进行赋权，得到加权最小二乘模型。

Theil-Sen 回归是一个参数中值估计器，它适用泛化中值，对多维数据进行估计，因此其对多维的异常点 (outliers 外点) 有很强的稳健性，对于回归模型：

$$Y=Xw+\epsilon \tag{4}$$

Theil-Sen 回归则是这么处理的：

$$w_k=\mathrm{Median}\left\{\frac{y_i-y_j}{x_{ki}-x_{kj}}:x_{ki}\neq x_{kj},i<j=1,\ldots,n\right\} \tag{5}$$

3.2.2 非线性模型

在第二节数据处理中已经对因变量进行取对数操作，因此本质上本报告所探讨的回归模型都属于非线性模型的行列，但如果仅仅将 $log(Y)$ 看做因变量，常用的非线性模型是多项式模型，将一般非线性模型表示如下：

$$\hat{Y} = f(X, w) \tag{6}$$

（4）多项式回归模型 (degree=1,degree=2,degree=3)

多项式回归模型根据“度”的大小，以原始自变量集为基础，创建元素个数为 $\sum_{i=1}^{degree}\binom{p}{i}$ 的新自变量集, 其中 $degree$ p 分别表示多项式函数的度与自变量的个数，degree=1 退化成多元线性回归模型，以“degree=2”为例，调用 $PolynomialFeatures()$ 函数创建自变量集:

$$Z = \left[x_1, x_2, \ldots, x_7, x_1^2, x_2^2, \ldots, x_7^2, x_1x_2, \ldots\right] \tag{7}$$

在新的变量集上建立线性回归模型：

$$\hat{Y} = f(X, w) = Z\beta \tag{8}$$

$$\min_{\beta} \|Z\beta - Y\|_2^2 \tag{9}$$

本次实验选用 degree=1,2,3 的多项式模型，尽管度越大，拟合效果越好，但是在测试集上的表现却不尽如人意，出现了过拟合现象，为解决上述问题，本报告通过（a）删除一个不相关的自变量减少参数个数（b）先创建自变量集，再通过逐步回归在自变量集合中选择最优子集进行模型拟合和预测，模型 (5) 便基于第二种思想实现

（5）基于多项式变量集的逐步回归

调用 PolynomialFeatures() 函数构建 degree=3 的解释变量组合全集，以拟合优度检验系数为依据通过逐步回归选择最优的自变量组合。

3.2.3 拟合优度与预测指标的建立

按照 3：1 的比例将数据集分为训练集（拟合）和测试集（预测）。设 y 为观测值（训练集上的单位房价）, 其均值为 $\bar{y}$, 拟合值为 $\hat{y}$, 记:

总平方和 (SST): $\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2$

回归平方和 (SSR): $\sum_{i=1}^{n}\left(\hat{y}_i - \bar{y}\right)^2$

残差平方和 (SSE): $\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2$

满足关系式: $SST = SSR + SSE$

可决系数:

$$R^2 = \frac{SSR}{SST} = \frac{\sum_{i=1}^{n}\left(\hat{y}_i - \bar{y}\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2} = 1 - \frac{SSE}{SST} \tag{10}$$

设 Y_{test} 为测试集上的单位房价, 相应的模型预测值为 $\hat{Y}_{test}$, 预测标准差的表达式为:

$$M_e = \frac{||Y_{test} - \hat{Y}_{test}||_2}{\sqrt{Y_{length}}} \tag{11}$$

在模型结果中发现，M_e 的取值范围为 [0,1]，M_e 越小，说明模型预测效果越好；R^2 越大，说明模型拟合效果越好，定义综合拟合和预测的评价指标 Evalue:

$$Evalue = w_1 * (1 - M_e) + w_2 * R^2 \tag{12}$$

本报告中，取 $w_1 = w_2 = 0.5$,Evalue 值如下：

$$Evalue = \frac{1 - M_e + R^2}{2} \tag{13}$$

3.3 结果分析

3.3.1 基于 OLS 的多元线性回归结果及分析

基于已处理的数据集，调用 statsmodel 包中的最小二乘拟合，得到结果如下：

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     house price of unit area   R-squared:                       0.628
Model:                                  OLS   Adj. R-squared:                  0.621
Method:                       Least Squares   F-statistic:                     82.58
Date:                      Sat, 25 Dec 2021   Prob (F-statistic):           4.50e-60
Time:                              19:42:13   Log-Likelihood:                 1.4998
No. Observations:                       300   AIC:                             11.00
Df Residuals:                           293   BIC:                             36.93
Df Model:                                 6
Covariance Type:                  nonrobust
==============================================================================
                                        coef    std err          t      P>|t|
------------------------------------------------------------------------------
const                                 3.5620      0.014    253.179      0.000
transaction date                      0.0434      0.014      3.040      0.003
house age                            -0.0826      0.014     -5.839      0.000
distance to the nearest MRT station  -0.1752      0.031     -5.592      0.000
number of convenience stores          0.0707      0.018      3.855      0.000
latitude                              0.0939      0.018      5.283      0.000
longitude                             0.0064      0.026      0.242      0.809
==============================================================================
Omnibus:                      107.506   Durbin-Watson:                   2.173
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              973.970
Skew:                          -1.185   Prob(JB):                    3.20e-212
Kurtosis:                      11.503   Cond. No.                         4.65
==============================================================================
```

图 8 多元线性回归结果

模型结果转化如下：

$$logY = 0.0434x_1 - 0.0826x_2 - 0.1752x_3 + 0.0707x_4 + 0.0939x_5 + 0.0064x_6 + 3.562 \tag{14}$$

从结果中可以看出，房价与房龄、到地铁站的距离呈现负相关的关系，与交易日期、便利店的数目、经纬度呈正相关的关系。对模型整体的线性性质进行 F 检验，由 $p(F > F - statistics) >> 0.05$ 知，在显著性水平 =0.05 的情况下，有足够充分的理由拒绝原假设，因变量与所有自变量之间呈现显著线性性。对单一自变量与因变量的线性关系进行 t 检验，由检验结果知：经度 x_6 与因变量之间不存在显著线性相关性，或者说明经度 x_6 与其他自变量间可能存在共线性关系。由此提出以下问题：是否可以从 6 个自变量中删除一个自变量使得模型可解释性不变，从而用更少的变量预测房价。上述猜想将通过多项式回归进行验证，对各自变量进行多重共线性检验。

3.3.2 模型结果比较

将各回归模型的评价指标统计如表 3，R_2 的变化说明增加参数个数或者说提高模型的复杂度可以较为显著地提高回归模型在数据集的拟合程度，但容易出现过拟合现象，因此需要在模型的拟合优度与预测准确度上有一个较好的平衡，多项式 + 逐步线性回归方法具有最高的 Evalue 值，较好地解决上述问题。

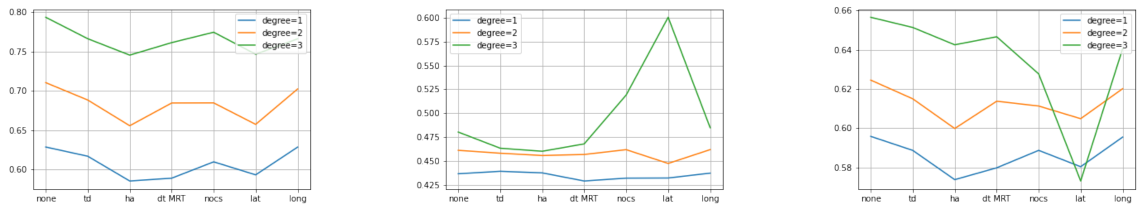
表 3 结果比较

模型名称	R^2	M_e	$Evalue$
多元线性回归	0.628	0.437	0.596
岭回归 + 交叉验证	0.628	0.434	0.597
Theil-Sen 回归	0.594	0.466	0.564
多项式回归 d=2	0.682	0.461	0.610
多项式回归 d=3	0.714	0.480	0.617
多项式 + 逐步线性回归	0.744	0.477	0.632

3.3.3 多重共线性与异方差检验

（1）基于多项式回归的多重共线性检验

分别设置 degree=1，2，3，对数据集进行多项式回归，得到在不同度之下，去除其中任意一个自变量的可决系数、预测标准差、综合评价指标 Evalue 变化趋势如图 9，比较’none’ 与’long’ 的纵坐标变化，可以看出，去除经度 x_6 对模型的可解释性几乎没有影响，因此本报告认为经度 x_6 与其他自变量之间存在多重共线性关系，可选择去除，剩余自变量之间则无明显多重共线性现象。



(a) 1 拟合优度检验的可决系数变化　　(b) 预测标准差变化　　(c) 综合评价指标变化

图 9 自变量的多重共线性检验

（2）异方差检验

下图描述了用残差平方估计的随机项方差随 y 值的变化情况，除了几个异常点外，其他点均匀地分布在平行于 x 轴的水平线两边，说明经处理得到的数据集符合同方差性的假设，即随机项满足独立同分布。

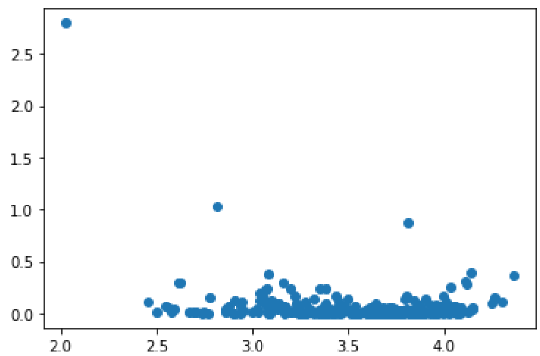


图 10 异方差检验-同方差性

3.3.4 基于多项式变量集的逐步回归结果及分析

多项式回归产生了含有 84 个元素的自变量集，通过逐步回归最终确定 36 个对本线性模型贡献度较大的自变量子集。和图 8 表示的普通最小二乘法相比，基于多项式变量集的逐步回归具有较低的 AIC 值，在自变量数目较少的房价预测模型选择中，可优先进行使用。

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.775
Model:                            OLS   Adj. R-squared:                  0.744
Method:                 Least Squares   F-statistic:                     25.12
Date:                Sun, 26 Dec 2021   Prob (F-statistic):           9.35e-66
Time:                        11:05:52   Log-Likelihood:                 76.563
No. Observations:                 300   AIC:                            -79.13
Df Residuals:                     263   BIC:                             57.91
Df Model:                          36
Covariance Type:            nonrobust
==============================================================================
```

图 11 回归结果

4 模型的改进计划与应用价值

4.1 模型的改进

(1) 本文所用的模型基于经典统计模型的组合，本质上是最小二乘的思想实现，属于广义线性模型的行列。而在数据量级足够大的情况下，考虑搭建神经网络模型和效率更高的优化算法进一步提高被预测房价的说服力。

(2) 结合房地产理论中的估价模型，使得预测模型更加“房地产”化。

4.2 模型应用价值

分别从房地产开发商和购房人员的角度考虑，本模型的应用价值概括为以下两点。

(1) 构建影响房价的指标体系。用于房产开发商在选址、户型优化、环境设计等方面的决策参考，构建思想可用于不同地区、不同年份的房价数据集。

以本次研究对象为例，从模型结果可以看出，到地铁站的距离是一个显著影响房价的解释变量。通过设置距离阈值，统计阈值范围内的各地铁站点周围的历史房价数据，输入本文提出的预测模型，从而得到预测房价，房地产开发商根据权衡开发成本和预测的利润进行房产开发选址决策。

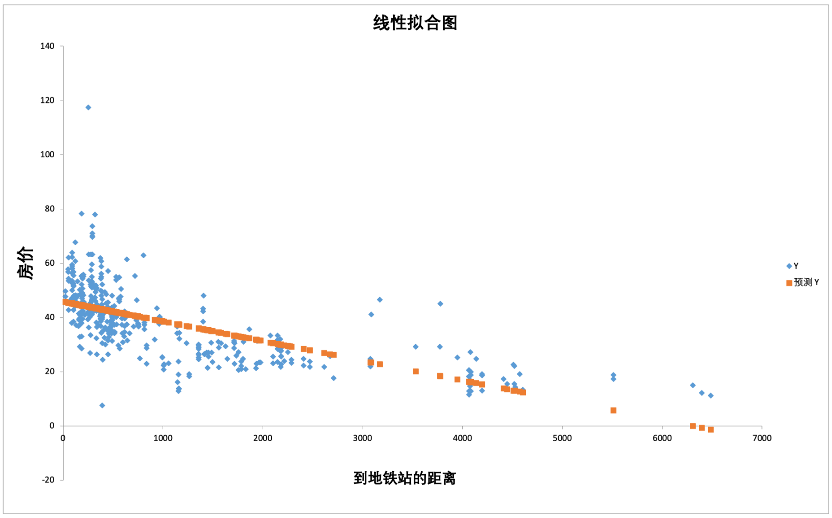


图 12 到地铁站的距离与房价的线性拟合图

(2) 基于重构解释变量的房价预测模型。多项式回归为解释变量数目较少的数据集重构解释变量，逐步回归选择最优的解释变量组合，提高回归模型在训练集和预测集上的拟合优度，增大被预测房价的可信度。具有购房或投资意向的人员可根据目标区域发展情况来预测未来的房价，从而对“在什么位置买?”“什么时间买?”有较为清晰的规划。

参考文献

[1]Yeh I C , Hsu T K . Building real estate valuation models with comparative approach through case-based reasoning[J]. Applied Soft Computing, 2018:S1568494618300358.

[2] 罗博炜, 洪智勇, 王劲屹. 多元线性回归统计模型在房价预测中的应用 [J]. 计算机时代,2020(06):51-54.DOI:10.16644/j.cnki.cn33-1094/tp.2020.06.014.

[3] 张智鹏, 郑大庆. 影响区域房价的客观因素挖掘分析 [J]. 计算机应用与软件,2019,36(11):32-38+85.

[4] 刘聪. 北京市房价的影响因素及预测研究 [D]. 大连理工大学,2019.DOI:10.26991/d.cnki.gdllu.2019.003203.

[5] 马娟, 陈兰兰. 我国新一线城市房价走势特征及影响因素分析 [J]. 区域金融研究,2019(06):74-79.

附录

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

# In[1]:
import pandas as pd
# In[148]:

data_raw=pd.read_excel('Real estate valuation data set.xlsx')
```

```
#data_raw.columns=['交易日期','房龄','到MRT站的距离','便利店数目',
#'纬度','经度','每平方房价']
data_raw.columns=['transaction date','house age','dis to MRT',
'number of convenience stores','latitude','longitude','house price of unit area']
data_raw.head()
describe=data_raw.describe()
describe
print(describe['house price of unit area'].map(lambda x: x*0.23/3.3))
#将台湾房价单位换算成人民币每平方米
#describe.to_excel("describe_output.xlsx")
print(data_raw.loc[data_raw['house price of unit area'] > 100,
['house price of unit area']])
print(data_raw.loc[270])
data_raw=data_raw.drop([270])
data_raw
#data_raw['dis to MRT']


# In[3]:

#min-max归一化处理，StandardScaler,MinMaxScaler
from sklearn import preprocessing
#from sklearn.preprocessing import StandardScaler
scaler = preprocessing.StandardScaler()#sklearn标准化返回的是数组，
#以下将其转化为数据框形式
data_r=data_raw.drop(['house price of unit area'], axis=1)
data=pd.DataFrame(scaler.fit_transform(data_r),
columns=['transaction date','house age','distance to the nearest MRT
 station','number of convenience stores','latitude','longitude'])
data['house price of unit area']=data_raw['house price of unit area']
data=data.dropna()
print(data)
train_data=data.iloc[:300]#大约75%的数据集用作训练集
test_data=data.iloc[300:412]#大约25%的数据集用作测试集
#train_data

# In[4]:
#数据可视化-基于主要解释变量的箱线图
from plotnine import *
import matplotlib.pyplot as plt

fig, ax= plt.subplots(nrows=1, ncols=3,figsize=[10,5],dpi=150)
```

```
#返回一个图形对象，和三个子图对象
#fig.suptitle('基于主要解释变量的箱线图') #super title

ax[0].boxplot(data_raw['house age'],widths=0.25)
ax[0].set_xlabel('house age')

ax[1].boxplot(data_raw['dis to MRT'],widths=0.25)
ax[1].set_xlabel('dis to MRT')

ax[2].boxplot(data_raw['number of convenience stores'],widths=0.25)
ax[2].set_xlabel('number of convenience stores')

plt.show()
#结果说明到MRT的距离相差比较大
n1=len(data_raw.loc[data_raw['dis to MRT'] > 3000, ['dis to MRT']])#n1=41
n2=len(data_raw.loc[data_raw['dis to MRT'] > 4000, ['dis to MRT']])#n2=33
#print(n1)
#print(n2)

# In[140]:

fig=plt.figure(dpi=100)
plt.hist(data_raw['transaction date'], bins = 12, density = True,
 facecolor = 'g',edgecolor='k',alpha=0.5)
plt.show()
# In[145]:

fig=plt.figure(dpi=100)
plt.hist(data_raw['house price of unit area'], bins = 50,
density = True, facecolor = 'g',edgecolor='k',alpha=0.5)
#hp=data_raw['house price of unit area']
#hp.plot(kind = 'kde')
plt.show()
#发现一条异常数据，第270条数据
#房价的分布基本上呈现正态分布

# In[7]:

import seaborn as sns

sns.set_style('darkgrid')
sns.distplot(data_raw['house price of unit area'])
```

```
#sns.kdeplot(data_raw['house price of unit area'])

# In[8]:

sns.kdeplot(data=data_raw, x="house price of unit area", shade='True')

# In[7]:

#数据可视化—绘制地图气泡图
import geopandas as gpd
import pandas as pd
from plotnine import*
df=data_raw.iloc[:,4:7]
df
df_map = gpd.GeoDataFrame.from_file('COUNTY_MOI.shp')
df_map = df_map.drop([0,1,2,3,4,5,6,7,8,9,10,11,14,15,16,17,18,19,20,21])
#选择新北，台北，基隆
print(df_map)
#df=pd.merge(right=df_map,left=df)
df=gpd.GeoDataFrame(df)

base_plot=(ggplot()+
geom_map(df_map,fill='white',color='gray')+
geom_point(df,aes(x='longitude',y='latitude',fill='house price of unit
area',size='house price of unit area'),shape='o')
#geom_text(aes(x='long', y='lat', label='city'),colour="black",size=10,
nudge_y=-1.5)
#scale_size(name='price')
)
print(base_plot)

# In[268]:

###### 多元线性回归
from sklearn import linear_model
import math
#划分解释变量和因变量
train_data_X=train_data.drop(['house price of unit area'], axis=1)
#train_data_X['col2'] = df['col1'].map(lambda x: x**2)
train_data_y=train_data['house price of unit area']
test_data_X=test_data.drop(['house price of unit area'], axis=1)
test_data_y=test_data['house price of unit area']
```

```
train_data_y=train_data_y.map(lambda x: math.log(x))
test_data_y=test_data_y.map(lambda x: math.log(x))
#定义误差
def errors(y,y_hat):
n=0
if len(y)!=len(y_hat):
return 0
else:
for i in range(len(y)):
n+=pow(y[i]-y_hat[i],2)
return math.sqrt(n/len(y))
#近似erroR
def errorlist(y,y_hat):
l=[]
for i in range(len(y)):
l.append(pow(y[i]-y_hat[i],2))
return l

#print(train_data_X)
#print(train_data_y)
#调用线性回归模型
reg = linear_model.LinearRegression()
reg.fit(train_data_X,train_data_y)

print("回归系数如下：")
print(reg.coef_)#斜率
print(reg.intercept_)#截距
r_score=reg.score(train_data_X,train_data_y)
print(r_score)
#R^2=0.6353332347315168, 线性关系减弱
#reg.score(test_data_X,test_data_y)
#reg.get_params()
testy_hat=reg.predict(test_data_X)
trainy_hat=reg.predict(train_data_X)

print(type(test_data_y.values))#将序列形式转化为数组形式

error_pre=errors(test_data_y.values,testy_hat)
error_reg=errors(train_data_y.values,trainy_hat)
el=errorlist(train_data_y.values,trainy_hat)
plt.scatter(train_data_y,el)#异方差性检验
print(error_pre)
```

```
print(error_reg)
print('————————————————————————')
print(0.5*(1-error_pre)+0.5*r_score)

# In[127]:

#Lasso回归，达到变量选择的效果，将部分特征系数降为0。适用于样本数量
#较为小的情况

clf = linear_model.Lasso(alpha=1)
clf.fit(train_data_X,train_data_y)

print(clf.coef_)
print(clf.score(train_data_X,train_data_y))
print(clf.intercept_)

# In[165]:

#岭回归
import numpy as np
clf = linear_model.RidgeCV(alphas = np.logspace(-10, 10, 100))
clf.fit(train_data_X,train_data_y)

#交叉验证法选择最优的alpha
print(clf.alpha_)

clf = linear_model.Ridge(alpha = clf.alpha_)
clf.fit(train_data_X,train_data_y)

print(clf.coef_)
print(clf.score(train_data_X,train_data_y))
r_score=clf.score(train_data_X,train_data_y)
print(clf.intercept_)
testy_hat=clf.predict(test_data_X)
trainy_hat=clf.predict(train_data_X)

error_pre=errors(test_data_y.values,testy_hat)
error_reg=errors(train_data_y.values,trainy_hat)
print(error_pre)
print(error_reg)
print('————————————————————————')
print(0.5*(1-error_pre)+0.5*r_score)
```

```


# In[166]:


#稳健回归ROBust
ransac = linear_model.TheilSenRegressor()
ransac.fit(train_data_X,train_data_y)
#inlier_mask = ransac.inlier_mask_
#outlier_mask = np.logical_not(inlier_mask)

# Predict data of estimated models
#line_y = lr.predict(line_X)
print(ransac.score(train_data_X,train_data_y))
r_score=ransac.score(train_data_X,train_data_y)
line_y_ransac = ransac.predict(test_data_X)
error_pre=errors(test_data_y.values,line_y_ransac)
print(error_pre)
print('--------------------------------')
print(0.5*(1-error_pre)+0.5*r_score)

# In[167]:

#多项式回归
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
import numpy as np
model = Pipeline([('poly', PolynomialFeatures(degree=3)),
('linear', LinearRegression(fit_intercept=False))])
# fit to an order-3 polynomial data
model = model.fit(train_data_X,train_data_y)
model.named_steps['linear'].coef_
print(model.score(train_data_X,train_data_y))
testy_hat=model.predict(test_data_X)
trainy_hat=model.predict(train_data_X)
error_pre=errors(test_data_y.values,testy_hat)
error_reg=errors(train_data_y.values,trainy_hat)
print(error_pre)#15.565531450875099
print(error_reg)
r_score=model.score(train_data_X,train_data_y)
print('--------------------------------')
```

```
print(0.5*(1-error_pre)+0.5*r_score)

# In[252]:

#多项式回归
from sklearn import linear_model
import math
#划分解释变量和因变量
Rscore3=[]
Error_pre3=[]
Error_reg3=[]
column=['transaction date','house age','distance to the nearest
MRT station','number of convenience stores','latitude','longitude']
columns=['none','td','ha','dt MRT','nocs','lat','long']
#model = Pipeline([('poly', PolynomialFeatures(degree=3)),
#                    ('linear', LinearRegression(fit_intercept=False))])

train_data_X=train_data.drop(['house price of unit area'], axis=1)
train_data_y=train_data['house price of unit area']
test_data_X=test_data.drop(['house price of unit area'], axis=1)
test_data_y=test_data['house price of unit area']
train_data_y=train_data_y.map(lambda x: math.log(x))
test_data_y=test_data_y.map(lambda x: math.log(x))
model = Pipeline([('poly', PolynomialFeatures(degree=3)),
('linear', LinearRegression(fit_intercept=False))])
model = model.fit(train_data_X,train_data_y)
model.named_steps['linear'].coef_
Rscore3.append(model.score(train_data_X,train_data_y))
testy_hat=model.predict(test_data_X)
trainy_hat=model.predict(train_data_X)
Error_pre3.append(errors(test_data_y.values,testy_hat))
Error_reg3.append(errors(train_data_y.values,trainy_hat))


for i in column:
train_data_X=train_data.drop(['house price of unit area'], axis=1)
train_data_X=train_data_X.drop([i], axis=1)
#train_data_y=train_data['house price of unit area']
test_data_X=test_data.drop(['house price of unit area'], axis=1)
test_data_X=test_data_X.drop([i], axis=1)
#print(train_data_X)
model = model.fit(train_data_X,train_data_y)
```

```
#model.named_steps['linear'].coef_
Rscore3.append(model.score(train_data_X,train_data_y))
testy_hat=model.predict(test_data_X)
trainy_hat=model.predict(train_data_X)
Error_pre3.append(errors(test_data_y.values,testy_hat))
Error_reg3.append(errors(train_data_y.values,trainy_hat))
#print(error_pre)#15.565531450875099
#print(error_reg)
#绘图
fig,ax=plt.subplots(nrows=3, ncols=1,figsize=[10,5],dpi=150,sharex=True)
#返回一个图形对象, 和三个子图对象
ax[0].plot(columns,Rscore3)
ax[0].set_title('R square')
ax[1].plot(columns,Error_pre3)
ax[1].set_title('Error_pre')
ax[2].plot(columns,Error_reg3)
ax[2].set_title('Error_reg')
plt.savefig('degree3')
print(Error_pre3)


# In[159]:


#多项式回归 degree=2
from sklearn import linear_model
import math
#划分解释变量和因变量
Rscore2=[]
Error_pre2=[]
Error_reg2=[]
train_data_X=train_data.drop(['house price of unit area'], axis=1)
test_data_X=test_data.drop(['house price of unit area'], axis=1)
model = Pipeline([('poly', PolynomialFeatures(degree=2)),
('linear', LinearRegression(fit_intercept=False))])
model = model.fit(train_data_X,train_data_y)
model.named_steps['linear'].coef_
Rscore2.append(model.score(train_data_X,train_data_y))
testy_hat=model.predict(test_data_X)
trainy_hat=model.predict(train_data_X)
Error_pre2.append(errors(test_data_y.values,testy_hat))
Error_reg2.append(errors(train_data_y.values,trainy_hat))
```

```

for i in column:
train_data_X=train_data.drop(['house price of unit area'], axis=1)
train_data_X=train_data_X.drop([i], axis=1)
#train_data_y=train_data['house price of unit area']
test_data_X=test_data.drop(['house price of unit area'], axis=1)
test_data_X=test_data_X.drop([i], axis=1)
#test_data_X['distance to the nearest MRT station'] = test_data_X
['distance to the nearest MRT station'].map(lambda x: x**3)

#调用线性回归模型

model = model.fit(train_data_X,train_data_y)
#model.named_steps['linear'].coef_
Rscore2.append(model.score(train_data_X,train_data_y))
testy_hat=model.predict(test_data_X)
trainy_hat=model.predict(train_data_X)
Error_pre2.append(errors(test_data_y.values,testy_hat))
Error_reg2.append(errors(train_data_y.values,trainy_hat))
#print(error_pre)#15.565531450875099
#print(error_reg)
#degree=3
fig,ax=plt.subplots(nrows=3, ncols=1,figsize=[10,5],dpi=150,sharex=True)
#返回一个图形对象，和三个子图对象
ax[0].plot(columns,Rscore2,c='g')
ax[0].set_title('R square')
ax[1].plot(columns,Error_pre2,c='g')
ax[1].set_title('Error_pre')
ax[2].plot(columns,Error_reg2,c='g')
ax[2].set_title('Error_reg')
plt.savefig('degree2')


# In[160]:


#多项式回归 degree=1
from sklearn import linear_model
import math
#划分解释变量和因变量
Rscore1=[]
Error_pre1=[]
```

```
Error_reg1=[]
train_data_X=train_data.drop(['house price of unit area'], axis=1)
model = Pipeline([('poly', PolynomialFeatures(degree=1)),
('linear', LinearRegression(fit_intercept=False))])
model = model.fit(train_data_X,train_data_y)
model.named_steps['linear'].coef_
Rscore1.append(model.score(train_data_X,train_data_y))
testy_hat=model.predict(test_data_X)
trainy_hat=model.predict(train_data_X)
Error_pre1.append(errors(test_data_y.values,testy_hat))
Error_reg1.append(errors(train_data_y.values,trainy_hat))

for i in column:
train_data_X=train_data.drop(['house price of unit area'], axis=1)
train_data_X=train_data_X.drop([i], axis=1)
#train_data_y=train_data['house price of unit area']
test_data_X=test_data.drop(['house price of unit area'], axis=1)
test_data_X=test_data_X.drop([i], axis=1)
#test_data_X['distance to the nearest MRT station'] = test_data_X
['distance to the nearest MRT station'].map(lambda x: x**3)
#test_data_y=test_data['house price of unit area']
#print(train_data_X)
#print(train_data_y)
#调用线性回归模型

model = model.fit(train_data_X,train_data_y)
#model.named_steps['linear'].coef_
Rscore1.append(model.score(train_data_X,train_data_y))
testy_hat=model.predict(test_data_X)
trainy_hat=model.predict(train_data_X)
Error_pre1.append(errors(test_data_y.values,testy_hat))
Error_reg1.append(errors(train_data_y.values,trainy_hat))
#print(error_pre)#15.565531450875099
#print(error_reg)
#degree=3
fig,ax=plt.subplots(nrows=3, ncols=1,figsize=[10,5],dpi=150,sharex=True)
#返回一个图形对象, 和三个子图对象
ax[0].plot(columns,Rscore1,c='orange')
ax[0].set_title('R square')
ax[1].plot(columns,Error_pre1,c='orange')
ax[1].set_title('Error_pre')
ax[2].plot(columns,Error_reg1,c='orange')
```

```
ax[2].set_title('Error_reg')
plt.savefig('degree1')

# In[161]:
plt.plot(columns,Rscore1,label='degree=1')
plt.plot(columns,Rscore2,label='degree=2')
plt.plot(columns,Rscore3,label='degree=3')
plt.legend()
plt.grid()
plt.savefig('rscore')
# In[254]:

plt.plot(columns,Error_pre1,label='degree=1')
plt.plot(columns,Error_pre2,label='degree=2')
plt.plot(columns,Error_pre3,label='degree=3')
plt.legend()
plt.grid()
plt.savefig('meanerror-pre')
temm=[1.0]*7
Evalue1=list(map(lambda x,y:x-y,temm,Error_pre1))
Evalue2=list(map(lambda x,y:x-y,temm,Error_pre2))
Evalue3=list(map(lambda x,y:x-y,temm,Error_pre3))
Evalue1=list(map(lambda x,y:(x+y)/2,Evalue1,Rscore1))
Evalue2=list(map(lambda x,y:x/2+y/2,Evalue2,Rscore2))
Evalue3=list(map(lambda x,y:(x+y)/2,Evalue3,Rscore3))
Evalue3

# In[163]:

#调用statsmodels中的模型进行实现
#OLS
import numpy as np
import statsmodels.api as sm
train_data_X=train_data.drop(['house price of unit area'], axis=1)
train_data_X = sm.add_constant(train_data_X)#添加截距项
#train_data_y=train_data['house price of unit area']
test_data_X=test_data.drop(['house price of unit area'], axis=1)
test_data_X = sm.add_constant(test_data_X)
model = sm.OLS(train_data_y, train_data_X)
results = model.fit()
summary=results.summary()

```

```
print(summary)
#调整,去除对因变量影响不显著的解释变量—经度, 似乎没什么用
train_data_XX=train_data_X.drop(['longitude'], axis=1)
model2 = sm.OLS(train_data_y, train_data_XX)
result = model2.fit()
summary2=result.summary()
print(summary2)

# In[75]:

#调用statsmodels中的模型进行实现
#OLS
import numpy as np
import statsmodels.api as sm
import math
train_data_y=train_data['house price of unit area']
train_data_y=train_data_y.map(lambda x: math.log(x))
model3 = sm.OLS(train_data_y,train_data_X)
result = model3.fit()
result.summary()
#print(summary)


# In[260]:


#多项式回归+逐步线性回归;
train_data_X=train_data.drop(['house price of unit area'], axis=1)
#X1 = PolynomialFeatures(interaction_only=True,degree=3).fit_transform
(train_data_X)#不含单项的高幂次项,似乎不太高
#X1=pd.DataFrame(X1)
train_data_y.index=range(300)
y=train_data_y
#y=train_data_y.map(lambda x: math.log(x))
X2 = PolynomialFeatures(degree=3).fit_transform(train_data_X)
X2=pd.DataFrame(X2)
column=[]
for i in range(len(X2.columns)):
temp='x{}'.format(i+1)
column.append(temp)
#print(column)
X2.columns=column
```

```
#print(X2)
model4 = sm.OLS(y,X2)
model4.fit().summary()

test_data_X=test_data.drop(['house price of unit area'], axis=1)
X3 = PolynomialFeatures(degree=3).fit_transform(test_data_X)
X3=pd.DataFrame(X3)
column=[]
for i in range(len(X3.columns)):
temp='x{}'.format(i+1)
column.append(temp)
#print(column)
X3.columns=column

X2['y']=y
X3['y']=y

#逐步线性回归选择变量
def forward_selected(data,response):

remaining = set(data.columns)
remaining.remove(response)
selected = []
current_score, best_new_score = 0.0, 0.0
#此处采用Rscore(拟合优度)进行比较也可以使用其他方法比如AIC
while remaining:
scores_with_candidates = []
for candidate in remaining:
formula = "{} ~ {}".format(response,' + '.join(selected + [candidate]))
score = smf.ols(formula=formula,data=data).fit().rsquared_adj
scores_with_candidates.append((score, candidate))
scores_with_candidates.sort()
best_new_score, best_candidate = scores_with_candidates.pop()
if current_score < best_new_score:
remaining.remove(best_candidate)
selected.append(best_candidate)
current_score = best_new_score
else:break
formula = "{} ~ {}".format(response,' + '.join(selected))
print("final formula is {}".format(formula))
model = smf.ols(formula=formula,data=data).fit()
#pre=model.predict(test)
```

```
return model

model=forward_selected(X2, 'y')
#pre=forward_selected(X2,'y',X3)
#打印出最后的回归模型
#print(model.model.formula)
# sl ~ rk + yr + 1
#print(model.params)
# Intercept           16203.268154
#rk[T.associate]       4262.284707
# rk[T.full]           9454.523248
# yr                    375.695643
# dtype: float64
# 0.835190760538

print(model.rsquared_adj)

# In[261]:

#结果显示
#print(model4.fit().summary())
print(model.summary())

# In[255]:

plt.plot(columns,Evalue1,label='degree=1')
plt.plot(columns,Evalue2,label='degree=2')
plt.plot(columns,Evalue3,label='degree=3')
plt.legend()
plt.grid()
plt.savefig('Evalue')

# In[267]:

print(Rscore1[0])
print(Rscore2[0])
print(Rscore3[0])
print(Error_pre1[0])
print((1-Error_pre2[0]+0.682)/2)
print((1-Error_pre3[0]+0.714)/2)
```