华东理工大学2022 –2023学年第2学期

《大数据与金融计算》实验报告

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实验名称 | 中国股市收益率可预测性的实证检验 | | | | | | | | | |
| 专 业 | | 金融学 | 姓名 | 刘海柏 | 学号 | 20030047 | | 班级 | | 金融202 |
| 实验时间 | | 2022/4/20 | | 实验地点 | 实验楼319 | | 指导教师 | | 蒋志强 | |

|  |
| --- |
| 实验目的/要求 |
| 1. 掌握收益率可预测性的样本内和样本外检验方法 2. 掌握收益率可以预测性的模型评价方法（统计意义和经济意义）。 3. 掌握机器学习方法：LASSO回归、Ridge回归和ElasticNet方法。 |
| 实验内容 |
| 1. 认真阅读三篇文献资料（自己也可以下载相关文献进行阅读），了解收益率可预测性的相关研究，重点关注其中的研究思路和方法。 2. 文件1EData\_PredictorData2019.xlsx给出了文献“Time-series and cross-sectional stock return forecasting: New machine learning methods”的研究数据，其中：   D12 表示 the twelve-month moving sum of S&P 500 dividens  E12 表示 the twelve-month moving sum of S&P 500 earnings  tbl 表示 three-month treasury bill yield  lty 表示 ten-year treasury bod yield  AAA 表示 AAA-related corporate bond  Rfree 表示 无风险利率  CRSP\_SPvw 表示 the value-weighted S&P 500 return  请你以上数据构建论文Sec 3中的12个预测因子，进行收益率可预测性研究，风险偏好系数设为*γ* = 5。（1）构建单因子可预测性模型，对12个因子分别进行可预测性实证检验；（2）构建多因子可预测性模型，用OLS回归进行预测性实证检验；（3）利用LASSO、Ridge和ElasticNet回归对多因子模型进行可预测性检验。   1. 从锐思数据库选取一只股票（股票最后两位代码与你学号最后两位一样，数据覆盖范围2000-2021），下载股票日度数据、月收益率、无风险利率、月市盈率、月每股收益、净资产收益、每股营业收入、月换手率、月beta系数。并计算月波动率（月内日收益率的平方和）、月流动性（|月收益率 / lg(月成交额)|），月股价高点（当月股价最高值与前三个 月股价的最大值的比值，用日度数据计算），月已实现偏度（参考论文2018金融研究-中国股票市场的已实现偏度与收益率预测）。   请利用以上数据进行收益率可预测性研究，风险偏好系数设为*γ* = 5。（1）构建单因子可预测性模型，探索哪些因子具有预测性；（2）结合机器学习的方法，探索多因子模型对股票收益率的可预测性。 |
| 实验总结  **请提供对本次实验结果的讨论分析，以及实验的心得和体会。包括对知识点的掌握，算法的理解，以及对理论课程和实验课程改进的建议。（不少于500字）** |
| 本次实验报告我觉得横跨比较多知识和许多复杂的因素需要完成的，在重现美股市场的实证研究的时候，因子已经被计算出来了，而且蒋老师也做了相应讲解，所以我这篇报告一开始以为还挺容易没想到确好复杂啊，在12个因子中有3个因子具有样本内和样本外的双重可预测性，且多因子模型也通过了检验。当我选取“000676”这只股票作为研究对象，自己计算各因子数据的时候，才感觉到这个过程的复杂性。  首先，在计算月已实现偏度的时候，需要下载股票的五分钟频率数据，我在RESSET没有找到，最后还是求助同学才通过“baostock”这个接口下载了数据，蛮神奇的，在搜索网路后，也得知如何将端口数据，下载下来，最后汇整成一个数据excel，才不会使得整个程序都要一直重跑，缩短了我完成报告的时间。其次，在计算月已实现偏度需要将日度数据加总以计算月度数据，我是通过skew函数去写的，一开始是自己手算，一个一个写程序，但是能力不足所以只能够用函数这样也比较快速解决  而且，可能是由于个股的特殊性，每个股票所得出的数据都不尽然全都相同，我所选的10个因子并没有表现出统计意义上的显著性，因为47结尾的股票实在是太难找了，缺失太多，所以每次下数据只能够找数据完整的，通过这次实验报告后，希望下次能得出一个令人比较满意的结果  的数据，并进一步严格因子计算的过程，争取得出一个令人比较满意的结果。 |
| 教师批阅： 实验成绩：  教师签名: 日期： |

**中国股市收益率可预测性的实证检验**

金融202（20030047）刘海柏

摘要：在通过比较美股和中国股市的收益率可测实验，选取A股市场中000676从2000年至2021年的日度数据和月度数据，选择因子包括月市盈率、月每股收益、净资产收益、每股营业收入、月换手率、月beta系数、月波动率、月流动性、月股价高点和月已实现偏度等12个指标的预测性，定义风险偏好系数为5的，构建单因子可预测性模型，挑选具有可预测性的因子，结合机器学习的方法，探索多因子模型对股票收益率的可预测性。

1 文献综述

1.1样本内预测

翻阅文献后可以发现，“收益率具有样本内可预测性而无样本外可预测性”这一理论观点已被许多经济学家证实，最著名的莫过是Goyal&Welch，经济学家使用其它研宄证实的能在一定条件下预测股票收益的变量进行梳理研宄，实证结果表明，大多数预测变量并没有表现出显著的样本外预测效果。且相比于历史平均收益的模型，潜在预测变量在样本外预测中没有展现出更多的有用信息。

除此之外，Bossaerts＆Hillion研宄表明，在国际股市数据集中，股利价格比、盈利价格比和长短期利率对股市具有显著的样本内预测能力，但发现没有预测模型有样本外预测能力。Boudoukhetal（2008）的研宄表明，对于现有文献中所使用的大多数预测变量，在无可预测性的零假设下，系数估计量几乎完全相关，资产定价模型中的s和回归的s几乎呈线性增长。这意味着从更长远的角度来看，更高水平的可预测性也是可以预期的。

研宄收益可预测性的目的为在选择投资组合、资产定价和风险管理提出建议，但如果没有样本外可预测性，或者虽然回报率在某种程度上可以预测，但这种预测对投资组合建议用处不大，而预测也将意义性也不大

1.2样本外预测

我们讨论在样本外预测的文献非常少, Haugen和Baker (1996) 利用预测股票收益率波动的Barra模型 (美国实务界最成功的风险模型) 的股票特征来预测股票未来的横截面收益率, 重点讨论了这些截面特征在样本外的预测能力, 发现样本外的预测不仅在美国股票市场，而且在全球主要的股票市场都具有很强的预测能力。Hanna和Ready (2005) [4]则在他们的基础上进行深入分析。国内关于股票截面特征的样本外预测能力则从来没有深入讨论过

股改前的数据表明, 账面市值比等估值指标对股票横截面收益率的样本内预测能力比较弱, 因此多数学者更偏向认为，股票市场收益率的样本外预测能力的证据比样本内更具有说服力，即基于知道当前数据来预测下一期，到下一期时记录预测误差、重新估计模型、进行新的预测，因此越来越多的学者用样本外预测检验变量的预测能力。

2 模型和方法

2.1收益率可预测性研究：

2.1.1构建单因子可预测性模型

2-1图示

低可信度描述已自动生成

样本内检验统计显著性检验

* 最小二乘估计参数*α*和*β*
* 参数显著性检验 计算*R*2，很小，上限 5%

根据公式计算和MSFE-adjusted统计量

黑色的钟表

描述已自动生成

（2-2） （2-3）

图片包含 游戏机, 桌子

描述已自动生成

（2-4）

样本外经济显著性检验

根据模型预测结果进行投资决策：

文本, 信件

描述已自动生成投资者风险偏好*γ*，投资风险资产和无风险资产

1. 在*t*时刻，风险资产的配置比例：

文本, 信件

描述已自动生成

1. 下一个时刻的组合收益率：
2. 文本, 信件

   描述已自动生成计算投资效用：

文本, 信件

描述已自动生成考虑历史均值基准模型：

1. 在*t*时刻，风险资产的配置比例：

文本, 信件

描述已自动生成

1. 下一个时刻的组合收益率：
2. 文本, 信件

   描述已自动生成计算投资效用：

\*\*\*检验的均值不为0，计算utility gain：



投资人为获得超过历史平均预测而愿意支付的年化管理费用

得到的结果如表2.1-2.3所示：

**表2.1 12个变量的单因子模型样本内检测结果**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **预测变量** | **b** | **k** | **预测变量** | **b** | **k** |
| **DP** | 0.029255\*\*\* | 0.006683\* | **CREDIT** | 0.004699 | 0.278466 |
| **EP** | 0.028835\*\*\* | 0.008089\*\* | **PPIG** | 0.006784\*\*\* | -0.000673 |
| **VOL** | 0.002326 | 0.154138 | **IPG** | 0.006413\*\*\* | 0.000828 |
| **BILL** | 0.006617\*\*\* | -0.241740 | **MA(1,12)** | 0.002004 | 0.006707\* |
| **BOND** | 0.006523\*\*\* | -1.291771\*\*\* | **MA(3,12)** | 0.003981 | 0.003856 |
| **TERM** | 0.004603\* | 0.119805 | **MOM(6)** | 0.003723 | 0.004363 |

**表2.2 12个变量的单因子模型样本外检测结果**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **预测变量** |  | **MSFE-adjusted统计量** | **预测变量** |  | **MSFE-adjusted统计量** |
| **DP** | -0.005143 | 2.062252\*\* | **CREDIT** | -0.000930 | 0.735414 |
| **EP** | -0.015106 | 0.640065 | **PPIG** | -0.001019 | 0.862098 |
| **VOL** | 0.004113 | 2.097376\*\* | **IPG** | 0.000464 | -0.952251 |
| **BILL** | -0.000322 | 2.795891\*\*\* | **MA(1,12)** | 0.001486 | 0.707132 |
| **BOND** | 0.011894 | 3.107020\*\*\* | **MA(3,12)** | -0.001900 | 0.137456 |
| **TERM** | -0.001091 | 1.750775\* | **MOM(6)** | -0.001505 | 0.490648 |

**表2.3 12个变量的经济显著性检验结果**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **预测变量** |  | **Umean** | **经济预测能力** | **预测变量** |  | **Umean** | **经济预测能力** |
| **DP** | -0.000301 | 0.003771 | 无 | **CREDIT** | 0.000033 | 0.004105 | 有 |
| **EP** | -0.000170 | 0.003902 | 无 | **PPIG** | -0.000006 | 0.004066 | 没有 |
| **VOL** | -0.000069 | 0.004003 | 无 | **IPG** | 0.000003 | 0.004075 | 有 |
| **BILL** | 0.000049 | 0.004121 | 有 | **MA(1,12)** | 0.000172 | 0.004244 | 有 |
| **BOND** | 0.000687 | 0.004759 | 有 | **MA(3,12)** | 0.000066 | 0.004138 | 有 |
| **TERM** | 0.000064 | 0.004136 | 有 | **MOM(6)** | 0.000062 | 0.004134 | 有 |

表2.1和表2.2分别代表了12个变量单因子模型的样本内和样本外检测结果。b和k分别表示单因子模型样本内检测的常数项和系数，\*，\*\*和\*\*\*分别表示10%，5%和1%水平的显著性。由公式2-3计算得出，**>0表示有样本外预测效果**。MSFE-adjusted统计量是回归di和i的常数项的t统计量，di由公式2-4得出。由表2.1可知，在12个因子中，因子DP、EP、BOND、TERM、PPIG和IPG具有显著的样本内预测能力，由表2.2可知，在12个因子中，因子DP、VOL、BILL、BOND、TERM有显著的样本外预测能力。综合来看，因子DP、BOND和TERM的样本内预测能力和样本外预测能力都是显著的。表2.3报告了12个因子的经济显著性检验结果，可以得出因子BILL、BOND、TERM、CREDIT、IPG、MA(1,12)、MA(3,12)、MOM(6)具有经济显著性。

2.2多因子模型进行可预测性检验

多变量模型的回归式如下：

(2-5)

(2-6)

(2-7)

2.2.1机器学习方法：LASSO

机器学习：正则化

在目标函数中增加一项惩罚项（度量模型复杂度）

降低模型样本内性能，增加样本外稳定性

LASSO: Least absolute shrinkage and selection operator

压缩估计，构造一个惩罚函数，压缩一些系数，设定为零，保留子集收缩

处理具有多重共线性数据的有偏估计

OLS方法：min *L*(*θ*)，其中

缺点：预测因子数量接近观测值数量，导致过拟合

因子之间存在多重共线性，降低拟合效果

机器学习：正则化

在目标函数中增加一项惩罚项（度量模型复杂度）

降低模型样本内性能，增加样本外稳定性

LASSO: Least absolute shrinkage and selection operator

压缩估计，构造一个惩罚函数，压缩一些系数，设定为零，保留子集收缩

处理具有多重共线性数据的有偏估计

引入L1正则化 （2-5）

构造目标函数 （2-6）

其中， 为系数， 为参数，且为正值，正则化强度。

2.2.2机器学习方法： ElasticNet

L2正则化

Ridge方法引入L2正则化，构造目标函数：

ElasticNet方法引入L1和L2正则化，

其惩罚项为：

目标函数为：

其中， 为参数， =1时，惩罚项变成L1正则化，对应LASSO方法。

LASSO回归太过（太多特征被稀疏为0）

Ridge回归正则化不够（回归系数衰减太慢）

使用ElasticNet方法回归

通用近似定理：一个包含足够多隐藏层神经元的多层前馈网络，能以任意精度逼近任意预定的连续函数。

神经网络由输入层、一个或多个隐藏层和一个输出层组成。类似于生物大脑中的轴突，网络层代表一组神经元，神经元通过激活函数将输入信号转换成输出信号

神经网络存在多种激活函数，常用的有 RELU, tanh(回归), sigmoid, softmax(分类)

目标函数用来衡量预测值与真实值间的差异，一般采用MSE。

①每个神经元从所有输入单元线性地提取信息

②每个神经元在将输出发送到下一层之前，将激活函数应用于其输出值。

隐藏层中的第二个神经元将输入转换为输出：。

输出预测值是每个神经元输出值的线性加总：

**表2.4 12个变量的多因子模型检测结果**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **方法** |  | **MSFE-adjusted统计量** |  | **Umean** | **经济预测能力** |
| **OLS** | -0.032711 | 2.848436\*\*\* | 0.000494 | 0.004566 | 有 |
| **Ridge** | 0.014139 | 2.522708\*\* | 0.000159 | 0.004231 | 有 |
| **LASSO** | 0.009689 | 2.550774\*\* | 0.000095 | 0.004167 | 有 |
| **ElasticNet** | 0.010282 | 2.563646\*\* | 0.000095 | 0.004167 | 有 |

由表2.3可知，由上述12个变量构造的多因子模型无论是通过OLS方法还是机器学习方法得出的检测结果都具有统计显著性，且具有经济显著性，因此这个多因子模型可以具有预测性。

2.3 000676收益率可预测性实证检验

(一) 已实现偏度的构建方法

已实现偏度的计算分为以下几个步骤:

第一步，从利用 5 分钟的日内高频指数价格构造日度股票收益率:

(2-8)

其中，表示第 t 日 i 时刻的对数收益率, 表示第t日i时刻的对数价格。本文选用每个交易日9 ∶ 30－11 ∶ 30和13 ∶ 00－15 ∶ 00的市场价格。

第二步，计算日度已实现偏度。

首先，日度已实现方差为日内收益率的平方之和:

(2-9)

其中，N 为日内收益率的观测值，N = 48。其次，根据日度已实现偏度(ＲDSkewt ) 为:

(2-10)

2.3.1单因子可预测性模型

单变量模型的回归式如下：

(2-12)

(2-13)

按照2.1.2的方法进行回归，得到的结果如表2.5、表2.6和表2.7所示：

**表2.5 13个变量的单因子模型样本内检测结果**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **预测变量** | **b** | **k** | **预测变量** | **b** | **k** |
| **VOL** | 0.011918 | -0.000000 | **EPS** | 0.010996 | -0.056846 |
| **Trdsum** | 0.018069 | -0.000000 | **ROE** | 0.014760, | -0.003312 |
| **Turnover** | 0.002280 | 0.000117 | **IncomePS** | 0.006311, | 0.000618 |
| **Rfree** | 0.045411 | -14.591653 | **波动率** | 0.027662, | -0.050748 |
| **PE** | 0.008895 | -0.000009 | **high** | 0.147963, | -0.152401 |
| **rskew** | 0.006134 | -0.017396 | **Beta** | 0.015084 | -0.007384 |
| **liqulity** | 0.017033 | -2.067285 |  |  |  |

**表2.6 12个变量的单因子模型样本外检测结果**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **预测变量** |  | **MSFE-adjusted统计量** | **预测变量** |  | **MSFE-adjusted统计量** |
| **VOL** | -0.071984 | -0.589222 | **ROE** | -0.036375 | -0.513740 |
| **Trdsum** | -0.039604 | -0.485474 | **IncomePS** | -0.045270 | -0.347819 |
| **Turnover** | -0.041808 | 0.301881 | **波动率** | 0.037989 | 0.143608 |
| **PE** | -0.083731, | 0.417757 | **High** | 0.006229 | -0.534013 |
| **EPS** | -0.020552 | 0.251534 | **RSkew** | -0.003285 | -3.386123 |
| **Beta** | -0.334675 | -0.334675 | **liqulity** | -0.007271 | 0.705878 |

**表2.7 10个变量的经济显著性检验结果**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **预测变量** |  | **Umean** | **经济显著性** | **预测变量** |  | **Umean** | **经济显著性** |
| **VOL** | -0.000302 | 0.002569 | 无 | **ROE** | -0.000604 | 0.002268 | 无 |
| **Trdsum** | 0.000123 | 0.002989 | 有 | **IncomePS** | -0.000892 | -0.000892 | 无 |
| **Turnover** | 0.000027 | 0.002899 | 有 | **波动率** | 0.000318 | 0.003189 | 有 |
| **PE** | 0.000230 | 0.003101 | 有 | **High** | 0.000257 | 0.002961 | 有 |
| **EPS** | -0.000357 | 0.002514 | 无 | **RSkew** | -0.000010 | 0.002694 | 无 |
| **Beta** | -0.000105 | 0.002599 | 无 | **liqulity** | 0.000084 | 0.002788 | 有 |

表2.4和表2.5分别代表了12个变量单因子模型的样本内和样本外检测结果。b和k分别表示单因子模型样本内检测的常数项和系数，\*，\*\*和\*\*\*分别表示10%，5%和1%水平的显著性。由公式2-3计算得出**，>0，MFSEpvalue<0.1 表示有样本外预测效果**。MSFE-adjusted统计量是回归di和i的常数项的t统计量，di由公式2-4得出。由表2.5可知，在13个因子中，因子**Trdsum**和E**PS波动率high**具有显著的样本内预测能力，由表2.6可知，在12个因子中，没有因子显著的样本外预测能力。**波动率、High虽然**综合来看，没有同时通过了样本内预测和样本外预测检验。Liqulity、high、波动率PE、 Turnover 、Trdsum**具有经济意义 > 0**

2.3.2多因子可预测性模型

多变量模型的回归式如下，按照2.1.3的方法对多因子模型进行回归：

(2-14)

(2-15)

(2-16)

得到的结果如表2.8所示：

**表2.8 10个变量的多因子模型检测结果**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **方法** |  | **MSFE-adjusted统计量** |  | **Umean** | **经济显著性** |
| **OLS** | -0.238188 | 0.080018 | -0.000631 | 0.002240 | 无 |
| **Ridge** | -0.011109 | -0.390431 | -0.000091 | 0.002781 | 无 |
| **LASSO** | -0.009692 | -0.824109 | -0.000115 | 0.002756 | 无 |
| **ElasticNet** | -0.010883 | 0.129487 | **0.000031** | 0.002902 | 有 |

由表2.8可知，由上述12个变量构造的多因子模型只有**ElasticNet方法得到经济显著性检验其他都没有**

3 结果与讨论

在参考文献的基础上，我们对比了美股市场和中国A股市场的收益率可预测性。经过美股市场上的实证研究表明，在所选的12个因子中，DP、BOND和TERM三个因子在样本内和样本外都通过了统计检验，且总12个因子的多因子模型通过了统计与经济的双重检验。

而在中国A股市场上，由于本文只选取了一只股票000676，可能因为每只股票的特性不同，没法直接表示中国A股市场的完整样貌，000676没有表现出很好的可预测性性质，也没有任何因子通过了样本内和样本外可预测性检验，另外这12个因子的多Liqulity、high、波动率PE、 Turnover 、Trdsum具有经济意义。总的来说，在实证过程中，有可能因为个股差异问题，使得本文选取的因子不适用于预测本支股票的收益率，在理论上因子上需要重新做出调整。

但是，股市收益率可预测性的实证检验依然具有重要意义，它是对有效市场假说的一个有力挑战，也为投资者选股和基金业绩评价提供了一种新思路，可预测性对投资者有一定的意义。可以调整自己的投资策略，最大化收益并降低风险。其次，可预测性也有助于投资者更好地理解市场趋势和波动性，从而更好地处理市场的波动，之后应该根据我国A股市场的特点和运行机制来重新选取因子，进行实证研究，促进我国资本市场理论和实操发展。值得注意的是，可预测性是一个对实践具有重要指导意义的问题，它的出现意味着投资者运用恰当的策略可以获得高于市场平均回报的收益。但是，从本文的实证结果来看，股市收益率的可预测性也存在一定的局限性。因为是一个复杂的系统，受到许多因素的影响，包括经济政策、公司业绩、国际局势等等。预测可能会出现误差。

4 参考文献

1. 马树才,赵丰义.中国沪深股市可预测性研究[J].当代经济管理,2007(03):103-106.
2. 蒋志强,田婧雯,周炜星.中国股票市场收益率的可预测性研究[J].管理科学学报,2019,22(04):92-109.
3. 杨光艺.中国股市可预测性的稳健性检验[J].金融发展研究,2018,(12):3-9.
4. 姜富伟,凃俊,David E.Rapach,Jack K.Strauss,周国富.中国股票市场可预测性的实证研究[J].金融研究,2011(09):107-121.
5. David E. RAPACH, Guofu ZHOU. Time-series and Cross-sectional Stock Return Forecasting: New Machine Learning Methods[J]. Machine Learning for Asset Management: New Developments and Financial，2020:1-33.
6. 陈坚,张轶凡.中国股票市场的已实现偏度与收益率预测[J].金融研究,2018(09):107-125.

5 附录

import numpy as np  
import pandas as pd  
import baostock as bs  
import numpy as np  
import statsmodels.api as sm  
from math import pi  
import statsmodels.formula.api as smf  
import matplotlib.pyplot as plt  
import sklearn.linear\_model as sklm  
import warnings  
warnings.filterwarnings('ignore')  
#处理变量  
# data=pd.read\_excel('1EData\_PredictorData2019.xlsx',sheet\_name='Monthly')  
# data['DP']=data['D12'].apply(np.log)-data['Index'].apply(np.log)  
# data['EP']=data['E12'].apply(np.log)-data['Index'].apply(np.log)  
#  
# data['VOL']=data['CRSP\_SPvw'].abs().rolling(window=12).mean()\*np.sqrt(pi/6)  
# data['BILL']=data['tbl']-data['tbl'].rolling(window=12).mean()  
# data['BOND']=data['lty']-data['lty'].rolling(window=12).mean()  
# data['TERM']=data['lty']-data['tbl']  
# data['CREDIT']=data['AAA']-data['lty']  
# data['MA112']=data['Index']>=data['Index'].rolling(window=12).mean()  
# data['MA312']=data['Index'].rolling(window=3).mean()>=data['Index'].rolling(window=12).mean()  
# data['MOM6']=data['Index']>=data['Index'].shift(periods=6)  
# data['PPIG']=data['PPIG']  
# data['IPG']=data['IPG']  
# #  
# data['ExRet']=data['CRSP\_SPvw']-data['Rfree']  
# data[['MA112','MA312','MOM6']]=data[['MA112','MA312','MOM6']].astype(int)  
# #ppig ipg  
# data=pd.concat([data[['yyyymm','CRSP\_SPvw','Rfree','ExRet',  
# 'DP','EP','VOL','BILL','BOND','TERM','CREDIT','PPIG','IPG',  
# 'MA112','MA312','MOM6']],  
# data[['DP','EP','VOL','BILL','BOND','TERM','CREDIT','PPIG','IPG',  
# 'MA112','MA312','MOM6']].shift(periods=1)],axis=1)  
# print(data)  
#  
# data.columns=['yyyymm','Ret','Rfree','ExRet',  
# 'DP','EP','VOL','BILL','BOND','TERM','CREDIT','PPIG','IPG',  
# 'MA112','MA312','MOM6',  
# 'DPL1','EPL1','VOLL1','BILLL1','BONDL1','TERML1','CREDITL1','PPIGL1','IPGL1',  
# 'MA112L1','MA312L1','MOM6L1']  
#  
# data=data[data['yyyymm']>=192701]  
# data.reset\_index(drop=True,inplace=True)  
#  
# data['date']=pd.to\_datetime(data['yyyymm'],format='%Y%m')  
# print(data)  
# plt.figure(1)  
# plt.plot(data['date'],data['DP'])  
# plt.title('DP')  
# plt.show()  
# plt.plot(data['date'],data['EP'])  
# plt.title('EP')  
# plt.show()  
# plt.plot(data['date'],data['VOL'])  
# plt.title('VOL')  
# plt.show()  
# plt.plot(data['date'],data['BILL'])  
# plt.title('BILL')  
# plt.show()  
# plt.plot(data['date'],data['BOND'])  
# plt.title('BOND')  
# plt.show()  
# plt.plot(data['date'],data['TERM'])  
# plt.title('TERM')  
# plt.show()  
# plt.plot(data['date'],data['CREDIT'])  
# plt.title('CREDIT')  
# plt.show()  
# plt.plot(data['date'],data['PPIG'])  
# plt.title('PPIG')  
# plt.show()  
# plt.plot(data['date'],data['IPG'])  
# plt.title('IPG')  
# plt.show()  
# plt.plot(data['date'],data['MA112'])  
# plt.title('MA112')  
# plt.show()  
# plt.plot(data['date'],data['MA312'])  
# plt.title('MA312')  
# plt.show()  
# plt.plot(data['date'],data['MOM6'])  
# plt.title('MOM6')  
# plt.show()  
# # 单因子模型(双变量预测模型)：  
# def myfun\_stat\_gains(rout,rmean,rreal):  
# R2os=1-np.sum((rreal-rout)\*\*2)/np.sum((rreal-rmean)\*\*2)  
# d=(rreal-rmean)\*\*2-((rreal-rout)\*\*2-(rmean-rout)\*\*2)  
# x=sm.add\_constant(np.arange(len(d))+1)  
# model=sm.OLS(d,x)  
# # MSFE-adjusted 统计量：回归di和i，常数项的t统计量  
# fitres=model.fit()  
# MFSRadj=fitres.tvalues[0]  
# pvalue\_MFSEadj=fitres.pvalues[0]  
# if(R2os>0)&(pvalue\_MFSEadj<=0.01):  
# jud='在1%的显著性水平下有样本外预测能力'  
# elif(R2os>0)&(pvalue\_MFSEadj>0.01)&(pvalue\_MFSEadj<=0.05):  
# jud = '在5%的显著性水平下有样本外预测能力'  
# elif (R2os > 0) & (pvalue\_MFSEadj > 0.05) & (pvalue\_MFSEadj <= 0.1):  
# jud = '在10%的显著性水平下有样本外预测能力'  
# else:  
# jud='无样本外预测能力'  
# print('Stat gains:R2os={:f},MFSEadj={:f},MFSEpvalue={:f}'.format(R2os,MFSRadj,pvalue\_MFSEadj))  
# print('Inference:{:s}'.format(jud))  
#  
# return R2os,MFSRadj,pvalue\_MFSEadj  
#  
# def myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5):  
# # Out-of-sample tests 经济显著性检验  
# omg\_out=rout/volt2/gmm#风险系数=5  
# rp\_out=rfree+omg\_out\*rreal  
# Uout=np.mean(rp\_out)-0.5\*gmm\*np.var(rp\_out)  
# omg\_mean=rmean/volt2/gmm  
# rp\_mean=rfree+omg\_mean\*rreal  
# Umean=np.mean(rp\_mean)-0.5\*gmm\*np.var(rp\_mean)  
# DeltaU=Uout-Umean  
# # 检验𝒓 ̂的均值不为0，计算utility gain=Uout-Umean  
# if DeltaU<10\*\*-6:  
# jud='没有经济意义'  
# else:  
# jud='有经济意义'  
# print('Econ Gains:Delta U={:f},Umean={:f}'.format(DeltaU,Uout,Umean))  
# print('Inference:{:s}'.format(jud))  
#  
# return Uout,Umean,DeltaU  
# # 因子构建预测效力的指标  
# #样本内检验  
# #单因子模型：OLS线性拟合  
# factor='DP'  
# model=smf.ols('ExRet~DPL1',data=data[['ExRet','DPL1']])  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['DPL1']  
# rg\_DP\_pvalue=results.pvalues['DPL1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='EP'  
# model=smf.ols('ExRet~EPL1',data=data[['ExRet','EPL1']])  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['EPL1']  
# rg\_DP\_pvalue=results.pvalues['EPL1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='VOL'  
# model=smf.ols('ExRet~VOLL1',data=data[['ExRet','VOLL1']])#可以指定回归模型是什么样子  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['VOLL1']  
# rg\_DP\_pvalue=results.pvalues['VOLL1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='BILL'  
# model=smf.ols('ExRet~BILLL1',data=data[['ExRet','BILLL1']])#可以指定回归模型是什么样子  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['BILLL1']  
# rg\_DP\_pvalue=results.pvalues['BILLL1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='BOND'  
# model=smf.ols('ExRet~BONDL1',data=data[['ExRet','BONDL1']])#可以指定回归模型是什么样子  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['BONDL1']  
# rg\_DP\_pvalue=results.pvalues['BONDL1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='TERM'  
# model=smf.ols('ExRet~TERML1',data=data[['ExRet','TERML1']])#可以指定回归模型是什么样子  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['TERML1']  
# rg\_DP\_pvalue=results.pvalues['TERML1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='CREDIT'  
# model=smf.ols('ExRet~CREDITL1',data=data[['ExRet','CREDITL1']])#可以指定回归模型是什么样子  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['CREDITL1']  
# rg\_DP\_pvalue=results.pvalues['CREDITL1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='MA112'  
# model=smf.ols('ExRet~MA112L1',data=data[['ExRet','MA112L1']])#可以指定回归模型是什么样子  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['MA112L1']  
# rg\_DP\_pvalue=results.pvalues['MA112L1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='MA312'  
# model=smf.ols('ExRet~MA312L1',data=data[['ExRet','MA312L1']])  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['MA312L1']  
# rg\_DP\_pvalue=results.pvalues['MA312L1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='MOM6'  
# model=smf.ols('ExRet~MOM6L1',data=data[['ExRet','MOM6L1']])#可以指定回归模型是什么样子  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['MOM6L1']  
# rg\_DP\_pvalue=results.pvalues['MOM6L1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='PPIG'  
# model=smf.ols('ExRet~PPIGL1',data=data[['ExRet','PPIGL1']])#可以指定回归模型是什么样子  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['PPIGL1']  
# rg\_DP\_pvalue=results.pvalues['PPIGL1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
#  
# factor='IPG'  
# model=smf.ols('ExRet~IPGL1',data=data[['ExRet','IPGL1']])#可以指定回归模型是什么样子  
# results=model.fit()  
# rg\_con=results.params['Intercept']  
# rg\_con\_pvalue=results.pvalues['Intercept']  
# rg\_DP=results.params['IPGL1']  
# rg\_DP\_pvalue=results.pvalues['IPGL1']  
# if rg\_DP\_pvalue<=0.01:  
# jud='在1%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
# jud='在5%的显著性水平下有样本内预测能力'  
# elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
# jud = '在10%的显著性水平下有样本内预测能力'  
# else:  
# jud='无样本内预测能力'  
# print('In-sample tests for one factor model with OLs:')  
# print('Predictor:{:s}'.format(factor))  
# print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
# print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
# print('Inference:{:s}'.format(jud))  
# #样本外检验  
# #单因子模型：OLS线性拟合  
# factor\_out='DP'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','DP','DPL1']].copy(deep=True)#年份,收益率,无风险超额,DP,DPLOG  
# n\_in=np.sum(datafit['yyyymm']<=195612)#样本内  
# n\_out=np.sum(datafit['yyyymm']>195612)#样本外  
# rout=np.zeros(n\_out)#预测的坑  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)#波动率  
# for i in range(n\_out):#预测nout次  
# model=smf.ols('ExRet~DPL1',data=datafit[['ExRet','DPL1']].iloc[:(n\_in+i),:])#往前滚  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['DPL1']  
# f=datafit['DP'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='EP'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','EP','EPL1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~EPL1',data=datafit[['ExRet','EPL1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['EPL1']  
# f=datafit['EP'].iloc[n\_in+i-1]#前一天的因子  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)#最近12个月的平方和  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='VOL'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','VOL','VOLL1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~VOLL1',data=datafit[['ExRet','VOLL1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['VOLL1']  
# f=datafit['VOL'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='BILL'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','BILL','BILLL1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~BILLL1',data=datafit[['ExRet','BILLL1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['BILLL1']  
# f=datafit['BILL'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='BOND'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','BOND','BONDL1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~BONDL1',data=datafit[['ExRet','BONDL1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['BONDL1']  
# f=datafit['BOND'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='TERM'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','TERM','TERML1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~TERML1',data=datafit[['ExRet','TERML1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['TERML1']  
# f=datafit['TERM'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='CREDIT'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','CREDIT','CREDITL1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~CREDITL1',data=datafit[['ExRet','CREDITL1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['CREDITL1']  
# f=datafit['CREDIT'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='MA112'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','MA112','MA112L1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~MA112L1',data=datafit[['ExRet','MA112L1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['MA112L1']  
# f=datafit['MA112'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='MA312'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','MA312','MA312L1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~MA312L1',data=datafit[['ExRet','MA312L1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['MA312L1']  
# f=datafit['MA312'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='MOM6'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','MOM6','MOM6L1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~MOM6L1',data=datafit[['ExRet','MOM6L1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['MOM6L1']  
# f=datafit['MOM6'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='PPIG'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','PPIG','PPIGL1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~PPIGL1',data=datafit[['ExRet','PPIGL1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['PPIGL1']  
# f=datafit['PPIG'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# factor\_out='IPG'  
# datafit=data[['yyyymm','Ret','Rfree','ExRet','IPG','IPGL1']].copy(deep=True)  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
# for i in range(n\_out):  
# model=smf.ols('ExRet~IPGL1',data=datafit[['ExRet','IPGL1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# b=results.params['Intercept']  
# k=results.params['IPGL1']  
# f=datafit['IPG'].iloc[n\_in+i-1]  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
# rout[i]=k\*f+b  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
# print()  
# print('Out-of-sample tests for one factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
# #样本外检验  
# #多因子模型：OLS线性拟合  
# factor\_out='DP,EP,VOL,BILL,BOND,TERM,CREDIT,PPIG,IPG,MA112,MA312,MOM6'  
# datafit=data.copy(deep=True)  
#  
# n\_in=np.sum(datafit['yyyymm']<=195612)  
# n\_out=np.sum(datafit['yyyymm']>195612)  
# rout=np.zeros(n\_out)  
# rmean=np.zeros(n\_out)  
# rreal=np.zeros(n\_out)  
# rfree=np.zeros(n\_out)  
# volt2=np.zeros(n\_out)  
#  
# for i in range(n\_out):  
# model=smf.ols('ExRet~DPL1+EPL1+VOLL1+BILLL1+BONDL1+TERML1+CREDITL1+'  
# 'PPIGL1+IPGL1+MA112L1+MA312L1+MOM6L1',  
# data=datafit[['ExRet','DPL1','EPL1','VOLL1','BILLL1','BONDL1','TERML1',  
# 'CREDITL1','PPIGL1','IPGL1','MA112L1','MA312L1','MOM6L1']].iloc[:(n\_in+i),:])  
# results=model.fit()  
# k=results.params.values  
# f=datafit[['DP','EP','VOL','BILL','BOND','TERM','CREDIT','PPIG',  
# 'IPG','MA112','MA312','MOM6']].iloc[n\_in+i-1,:].values  
# f=np.concatenate((np.array([1]),f))  
# rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
# rfree[i]=datafit['Rfree'].iloc[n\_in+i]  
# rout[i]=np.sum(k\*f)  
# rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
#  
# print()  
# print('Out-of-sample tests for multi-factor model with OLS:')  
# print('Predictor:{:s}'.format(factor\_out))  
# R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
# Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
# del datafit  
#  
#  
#  
# #样本外检验  
# #多因子模型：LASSO回归，Ridge回归，ElasticNet回归  
# factor\_out = 'DP, EP, VOL, BILL, BOND, TERM, CREDIT, PPIG, IPG, MA112, MA312, MOM6'  
# factor\_list = np.array(['DP', 'EP', 'VOL', 'BILL', 'BOND', 'TERM', 'CREDIT', 'PPIG', 'IPG', 'MA112', 'MA312', 'MOM6'])  
#  
# datafit = data.copy(deep=True)  
#  
# n\_in = np.sum(datafit['yyyymm'] <= 195612)  
# n\_out = np.sum(datafit['yyyymm'] > 195612)  
# rout = np.zeros(n\_out)  
# rmean = np.zeros(n\_out)  
# rreal = np.zeros(n\_out)  
# rfree = np.zeros(n\_out)  
# volt2 = np.zeros(n\_out)  
# #Ridge  
# reg = sklm.RidgeCV(cv=10, fit\_intercept=True, normalize=True)  
# for i in range(n\_out):  
# X = datafit[['DPL1', 'EPL1', 'VOLL1', 'BILLL1', 'BONDL1', 'TERML1',  
# 'CREDITL1', 'PPIGL1', 'IPGL1', 'MA112L1', 'MA312L1', 'MOM6L1']].iloc[:(n\_in+i), :].values  
# y = datafit['ExRet'].iloc[:(n\_in+i)].values  
# reg.fit(X, y)  
# # print(factor\_list[np.abs(reg.coef\_) != 0])  
# k = np.concatenate((np.array([reg.intercept\_]), reg.coef\_))  
# f = datafit[['DP', 'EP', 'VOL', 'BILL', 'BOND', 'TERM', 'CREDIT', 'PPIG',  
# 'IPG', 'MA112', 'MA312', 'MOM6']].iloc[n\_in+i-1, :].values  
# f = np.concatenate((np.array([1]), f))  
# rreal[i] = datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in+i]  
# rout[i] = np.sum(k\*f)  
# rmean[i] = np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i] = np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
#  
# print()  
# print('Out-of-sample tests for multi-factor model with ML method:')  
# print('Predictor: {:s}'.format(factor\_out))  
# R2os, MFSEadj, pvalue\_MFSEadj = myfun\_stat\_gains(rout, rmean, rreal)  
# Uout, Umean, DeltaU = myfun\_econ\_gains(rout, rmean, rreal, rfree, volt2, gmm=5)  
# del datafit  
#  
# #Lasso  
# factor\_out = 'DP, EP, VOL, BILL, BOND, TERM, CREDIT, PPIG, IPG, MA112, MA312, MOM6'  
# factor\_list = np.array(['DP', 'EP', 'VOL', 'BILL', 'BOND', 'TERM', 'CREDIT', 'PPIG', 'IPG', 'MA112', 'MA312', 'MOM6'])  
#  
# datafit = data.copy(deep=True)  
#  
# n\_in = np.sum(datafit['yyyymm'] <= 195612)  
# n\_out = np.sum(datafit['yyyymm'] > 195612)  
# rout = np.zeros(n\_out)  
# rmean = np.zeros(n\_out)  
# rreal = np.zeros(n\_out)  
# rfree = np.zeros(n\_out)  
# volt2 = np.zeros(n\_out)  
# reg = sklm.LassoCV(random\_state=0, cv=10, fit\_intercept=True, normalize=True, precompute='auto', copy\_X=True, n\_jobs=-1, max\_iter=10\*\*9, tol=10-6)  
# # reg\_lasso = linear\_model.LassoLarsCV(cv=10, fit\_intercept=True, normalize=True, precompute='auto', copy\_X=True, n\_jobs=-1, max\_iter=10000000)  
# # reg = sklm.ElasticNetCV(random\_state=0, cv=10, fit\_intercept=True, normalize=True, precompute='auto', copy\_X=True, n\_jobs=-1, max\_iter=10\*\*9, tol=10-6)  
# for i in range(n\_out):  
# X = datafit[['DPL1', 'EPL1', 'VOLL1', 'BILLL1', 'BONDL1', 'TERML1',  
# 'CREDITL1', 'PPIGL1', 'IPGL1', 'MA112L1', 'MA312L1', 'MOM6L1']].iloc[:(n\_in+i), :].values  
# y = datafit['ExRet'].iloc[:(n\_in+i)].values  
# reg.fit(X, y)  
# # print(factor\_list[np.abs(reg.coef\_) != 0])  
# k = np.concatenate((np.array([reg.intercept\_]), reg.coef\_))  
# f = datafit[['DP', 'EP', 'VOL', 'BILL', 'BOND', 'TERM', 'CREDIT', 'PPIG',  
# 'IPG', 'MA112', 'MA312', 'MOM6']].iloc[n\_in+i-1, :].values  
# f = np.concatenate((np.array([1]), f))  
# rreal[i] = datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in+i]  
# rout[i] = np.sum(k\*f)  
# rmean[i] = np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i] = np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
#  
# print()  
# print('Out-of-sample tests for multi-factor model with ML method:')  
# print('Predictor: {:s}'.format(factor\_out))  
# R2os, MFSEadj, pvalue\_MFSEadj = myfun\_stat\_gains(rout, rmean, rreal)  
# Uout, Umean, DeltaU = myfun\_econ\_gains(rout, rmean, rreal, rfree, volt2, gmm=5)  
# del datafit  
#  
# #ElasticNet  
# factor\_out = 'DP, EP, VOL, BILL, BOND, TERM, CREDIT, PPIG, IPG, MA112, MA312, MOM6'  
# factor\_list = np.array(['DP', 'EP', 'VOL', 'BILL', 'BOND', 'TERM', 'CREDIT', 'PPIG', 'IPG', 'MA112', 'MA312', 'MOM6'])  
#  
# datafit = data.copy(deep=True)  
#  
# n\_in = np.sum(datafit['yyyymm'] <= 195612)  
# n\_out = np.sum(datafit['yyyymm'] > 195612)  
# rout = np.zeros(n\_out)  
# rmean = np.zeros(n\_out)  
# rreal = np.zeros(n\_out)  
# rfree = np.zeros(n\_out)  
# volt2 = np.zeros(n\_out)  
# reg = sklm.ElasticNetCV(random\_state=0, cv=10, fit\_intercept=True, normalize=True, precompute='auto', copy\_X=True, n\_jobs=-1, max\_iter=10\*\*9, tol=10-6)  
# for i in range(n\_out):  
# X = datafit[['DPL1', 'EPL1', 'VOLL1', 'BILLL1', 'BONDL1', 'TERML1',  
# 'CREDITL1', 'PPIGL1', 'IPGL1', 'MA112L1', 'MA312L1', 'MOM6L1']].iloc[:(n\_in+i), :].values  
# y = datafit['ExRet'].iloc[:(n\_in+i)].values  
# reg.fit(X, y)  
# # print(factor\_list[np.abs(reg.coef\_) != 0])  
# k = np.concatenate((np.array([reg.intercept\_]), reg.coef\_))  
# f = datafit[['DP', 'EP', 'VOL', 'BILL', 'BOND', 'TERM', 'CREDIT', 'PPIG',  
# 'IPG', 'MA112', 'MA312', 'MOM6']].iloc[n\_in+i-1, :].values  
# f = np.concatenate((np.array([1]), f))  
# rreal[i] = datafit['ExRet'].iloc[n\_in+i]  
# rfree[i] = datafit['Rfree'].iloc[n\_in+i]  
# rout[i] = np.sum(k\*f)  
# rmean[i] = np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
# volt2[i] = np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
#  
# print()  
# print('Out-of-sample tests for multi-factor model with ML method:')  
# print('Predictor: {:s}'.format(factor\_out))  
# R2os, MFSEadj, pvalue\_MFSEadj = myfun\_stat\_gains(rout, rmean, rreal)  
# Uout, Umean, DeltaU = myfun\_econ\_gains(rout, rmean, rreal, rfree, volt2, gmm=5)  
# del datafit  
#第三题  
data2=pd.read\_csv('实验4（3）数据.csv',encoding='GB2312')  
beta=pd.read\_csv('beta数据.csv',encoding='GB2312')  
beta=pd.DataFrame(beta)  
beta.columns=['Stkcd','日期\_Date','Beta']  
beta['日期\_Date']=pd.to\_datetime(beta['日期\_Date'])  
data2=pd.DataFrame(data2)  
data2['日期\_Date']=pd.to\_datetime(data2['日期\_Date'])  
data2['yearmonth'] = data2['日期\_Date'].dt.strftime('%Y%m')#记得先转换成datime  
print(data2)  
#代码 日期 收盘价  
daily1=pd.read\_csv('日度数据1.csv',encoding='GB2312')#2001-2010  
daily1=pd.DataFrame(daily1)  
daily2=pd.read\_csv('日度数据2.csv',encoding='GB2312')#2010-2015  
daily2=pd.DataFrame(daily2)  
daily3=pd.read\_csv('日度数据3.csv',encoding='GB2312')#2015-2020  
daily3=pd.DataFrame(daily3)  
daily4=pd.read\_csv('日度数据4.csv',encoding='GB2312')#2020-2021  
daily4=pd.DataFrame(daily4)  
dailydata=pd.concat([daily1,daily2,daily3,daily4],axis=0,join='outer')#拼在下面  
# axis=0 表示按行连接，axis=1 表示按列连接；join 参数指定合并方式  
dailydata.columns=['Stkcd','date','Clpr','4']  
# # 月股价高点（当月股价最高值与前三个 月股价的最大值的比值，用日度数据计算）  
# TypeError: first argument must be an iterable of pandas objects, you passed an object of type "DataFrame"  
# 出错原因就是，在使用pandas.concat(a,b)进行合并的时候，需要是list的形式。因此改成pandas.concat([a,b]),就可以成功合并  
#删除列  
dailydata.drop(columns=['4'],axis=0,inplace=True)  
dailydata['date']=pd.to\_datetime(dailydata['date'])  
dailydata['yearmonth'] = dailydata['date'].dt.strftime('%Y%m')#记得先转换成datime  
dailydata.index=dailydata['date']#设置所以索引  
s=np.unique(dailydata['yearmonth'].values)  
high=np.zeros(len(s))  
# # 重要复习月股价高点（当月股价最高值与前三个 月股价的最大值的比值，用日度数据计算）  
#整行数据能使用  
for i in range(len(s)):  
 a=dailydata[dailydata['yearmonth']==s[i]]#属于每个月的数据  
 high[i]=max(a.iloc[:,2].values)#收盘价数据  
print(dailydata)  
high\_data=pd.DataFrame(high,columns=['high'])  
print(high\_data)#每个月的月股价高点  
  
# print(len(s))#252  
dailydata.dropna(inplace=True)  
dailydata.columns=['Stkcd','date','Clpr','yearmonth']  
k=[]  
# .iloc根据行号索引3grouppby函数  
df\_grouped = dailydata.groupby(['yearmonth'])['Clpr'].std().reset\_index()#月波动率  
print(df\_grouped)  
df\_grouped.columns=['yearmonth','month\_vol']  
dailydata = pd.merge(dailydata, df\_grouped, on='yearmonth', how='inner')#将波动率拼回去日度数据  
print(dailydata)  
dailydata.set\_index('date', inplace=True)#日度数据转为月度数据  
dailydata2=dailydata.resample('M').mean()#日度数据转为月度数据  
dailydata2.reset\_index()#日度数据转为月度数据  
print(dailydata2)#日度数据转为月度数据  
dailydata2['date']=dailydata2.index  
dailydata2['yearmonth'] = dailydata2['date'].dt.strftime('%Y%m')#记得先转换成datime  
#拼回去上面的月度数据  
matrix=pd.merge(left=data2,right=dailydata2[['yearmonth','month\_vol']],  
 on='yearmonth',how='inner',sort=True)  
  
print(matrix)#拼起来  
matrix['high']=high\_data  
high\_point=matrix['high'].values  
for i in range(3,252):  
 high\_point[i]=(high[i]/max(high[i-1],high[i-2],high[i-3]))  
  
high\_point[0]= 26.68/27.07  
high\_point[1]=27.07/27.47  
high\_point[2]=27.47/26.62  
matrix['high']=high\_point  
print(matrix)  
from scipy.stats import skew  
  
  
  
  
  
# # # 登陆系统  
# # # lg = bs.login()  
# # # # 显示登陆返回信息  
# # # print(lg.error\_code)  
# # # print(lg.error\_msg)  
# # # rs = bs.query\_history\_k\_data("sz.000676",  
# # # "date,code,close",  
# # # start\_date='2001-01-01', end\_date='2021-12-31',  
# # # frequency='5', adjustflag="3")  
# # # print(rs.error\_code)  
# # # print(rs.error\_msg)  
# # # # 获取具体的信息  
# # # result\_list = []  
# # # while (rs.error\_code == '0') & rs.next():  
# # # # 分页查询，将每页信息合并在一起  
# # # result\_list.append(rs.get\_row\_data())  
# # # result = pd.DataFrame(result\_list, columns=rs.fields)  
# # # bs.logout()  
# # # result.to\_excel('日内数据.xlsx')  
result=pd.read\_excel('日内数据.xlsx')  
result['date']=pd.to\_datetime(result['date'])  
result['yearmonthdate']=result['date'].dt.strftime('%Y%m%d').astype(int)  
result['return']=np.log(result['close']/result['close'].shift(1))  
result = result.drop(result.columns[0], axis=1)  
result.dropna(inplace=True)  
result.index=result['date']  
result['return\_sq'] = result['return'] \*\* 2  
# 计算每个月的实现偏度  
skews = result.groupby(pd.Grouper(freq='M'))['close'].apply(skew)  
# 输出结果  
print(skews)  
skews=pd.DataFrame(skews)  
skews['日期']=skews.index  
skews['日期']=pd.to\_datetime(skews['日期'])  
skews.columns=['月偏度','日期']  
skews['yearmonth'] = skews['日期'].dt.strftime('%Y%m')#记得先转换成datime  
print(skews)  
# 使用resample函数将日度数据转换为月度数据  
# result\_monthly = result.resample('M').mean()  
# print(result\_monthly)  
matrix2=pd.merge(left=matrix,right=skews[['yearmonth','月偏度']], on='yearmonth',how='inner',sort=True)  
matrix2=pd.merge(left=matrix2,right=beta[['日期\_Date','Beta']], on='日期\_Date',how='inner',sort=True)  
print(matrix2)  
# matrix2.to\_excel('data数据.xlsx')  
print(matrix2.columns)  
  
matrix2.columns=[['Stkcd','date','close','VOL','Trdsum','Turnover','Ret','Rfree','PE',  
 'EPS','ROE','IncomePS','yyyymm','波动率','high','rskew','Beta']]  
print(matrix2)  
matrix2.dropna(inplace=True)  
# 月流动性（|月收益率 / lg(月成交额)|  
matrix2['liqulity']=abs((matrix2['Ret'].values/np.log(matrix2['Trdsum']).values))#注意一下一开始有报错  
print(matrix2)  
matrix2['ExRet']=matrix2['Ret'].values-matrix2['Rfree'].values  
data=pd.concat([matrix2[['VOL','Trdsum','Turnover','Ret','Rfree','PE',  
 'EPS','ROE','IncomePS','yyyymm','波动率','high','rskew','Beta','liqulity','ExRet']],  
 matrix2[['VOL','Trdsum','Turnover','Ret','Rfree','PE',  
 'EPS','ROE','IncomePS','波动率','high','rskew','Beta','liqulity','ExRet']].shift(periods=1)],axis=1)#按列也就是右边  
data.columns=['VOL','Trdsum','Turnover','Ret','Rfree','PE',  
 'EPS','ROE','IncomePS','yyyymm','波动率','high','rskew','Beta','liqulity','ExRet',  
 'VOL1','Trdsum1','Turnover1','Ret1','Rfree1','PE1',  
 'EPS1','ROE1','IncomePS1','波动率1','high1','rskew1','Beta1','liqulity1','ExRet1']  
data['date']=pd.to\_datetime(data['yyyymm'],format='%Y%m')  
data.reset\_index(drop=True,inplace=True)  
data.dropna(inplace=True)  
print(data)  
data.to\_excel('datacharles.xlsx')  
def myfun\_stat\_gains(rout,rmean,rreal):  
 R2os=1-np.sum((rreal-rout)\*\*2)/np.sum((rreal-rmean)\*\*2)  
 d=(rreal-rmean)\*\*2-((rreal-rout)\*\*2-(rmean-rout)\*\*2)#是y  
 x=sm.add\_constant(np.arange(len(d))+1)  
 model=sm.OLS(d,x)  
 fitres=model.fit()  
 MFSRadj=fitres.tvalues[0]  
 pvalue\_MFSEadj=fitres.pvalues[0]  
  
 if(R2os>0)&(pvalue\_MFSEadj<=0.01):  
 jud='在1%的显著性水平下有样本外预测能力'  
 elif(R2os>0)&(pvalue\_MFSEadj>0.01)&(pvalue\_MFSEadj<=0.05):  
 jud = '在5%的显著性水平下有样本外预测能力'  
 elif (R2os > 0) & (pvalue\_MFSEadj > 0.05) & (pvalue\_MFSEadj <= 0.1):  
 jud = '在10%的显著性水平下有样本外预测能力'  
 else:  
 jud='无样本外预测能力'  
 print('Stat gains:R2os={:f},MFSEadj={:f},MFSEpvalue={:f}'.format(R2os,MFSRadj,pvalue\_MFSEadj))  
 print('Inference:{:s}'.format(jud))  
  
 return R2os,MFSRadj,pvalue\_MFSEadj  
#统计显著性函数  
def myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5):  
 omg\_out=rout/volt2/gmm#rout 风险偏好 rout 波动率 gmm风险偏好==》t时刻资产配置的比率  
 rp\_out=rfree+omg\_out\*rreal#组合投资收益率  
 Uout=np.mean(rp\_out)-0.5\*gmm\*np.var(rp\_out)#计算投资效用  
 #以下考量均值模型  
 omg\_mean=rmean/volt2/gmm  
 rp\_mean=rfree+omg\_mean\*rreal  
 Umean=np.mean(rp\_mean)-0.5\*gmm\*np.var(rp\_mean)  
 DeltaU=Uout-Umean  
  
 if DeltaU<10\*\*-6:  
 jud='没有经济意义'  
 else:  
 jud='有经济意义'  
 print('Econ Gains:Delta U={:f},Umean={:f}'.format(DeltaU,Uout,Umean))  
 print('Inference:{:s}'.format(jud))  
  
 return Uout,Umean,DeltaU  
  
#样本内检验  
#单因子模型：OLS线性拟合  
factor='VOL'  
# smf跟sm差别在于smf可以直接指定谁跟谁的关系  
model=smf.ols('ExRet~VOL1',data=data[['ExRet','VOL1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['VOL1']  
rg\_DP\_pvalue=results.pvalues['VOL1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))#b是截距 k是斜率  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
factor='Trdsum'  
model=smf.ols('ExRet~Trdsum1',data=data[['ExRet','Trdsum1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['Trdsum1']  
rg\_DP\_pvalue=results.pvalues['Trdsum1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
factor='Turnover'  
model=smf.ols('ExRet~Turnover1',data=data[['ExRet','Turnover1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['Turnover1']  
rg\_DP\_pvalue=results.pvalues['Turnover1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
  
factor='Rfree'  
model=smf.ols('ExRet~Rfree1',data=data[['ExRet','Rfree1']])  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['Rfree1']  
rg\_DP\_pvalue=results.pvalues['Rfree1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
factor='PE'  
model=smf.ols('ExRet~PE1',data=data[['ExRet','PE1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['PE1']  
rg\_DP\_pvalue=results.pvalues['PE1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
factor='EPS'  
model=smf.ols('ExRet~EPS1',data=data[['ExRet','EPS1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['EPS1']  
rg\_DP\_pvalue=results.pvalues['EPS1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
factor='ROE'  
model=smf.ols('ExRet~ROE1',data=data[['ExRet','ROE1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['ROE1']  
rg\_DP\_pvalue=results.pvalues['ROE1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
factor='IncomePS'  
model=smf.ols('ExRet~IncomePS1',data=data[['ExRet','IncomePS1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['IncomePS1']  
rg\_DP\_pvalue=results.pvalues['IncomePS1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
factor='波动率'  
model=smf.ols('ExRet~波动率1',data=data[['ExRet','波动率1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['波动率1']  
rg\_DP\_pvalue=results.pvalues['波动率1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
factor='high'  
model=smf.ols('ExRet~high1',data=data[['ExRet','high1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['high1']  
rg\_DP\_pvalue=results.pvalues['high1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
factor='rskew'  
model=smf.ols('ExRet~rskew1',data=data[['ExRet','rskew1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['rskew1']  
rg\_DP\_pvalue=results.pvalues['rskew1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
factor='Beta'  
model=smf.ols('ExRet~Beta1',data=data[['ExRet','Beta1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['Beta1']  
rg\_DP\_pvalue=results.pvalues['Beta1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
  
factor='liqulity'  
model=smf.ols('ExRet~liqulity1',data=data[['ExRet','liqulity1']])#可以指定回归模型是什么样子  
results=model.fit()  
rg\_con=results.params['Intercept']  
rg\_con\_pvalue=results.pvalues['Intercept']  
rg\_DP=results.params['liqulity1']  
rg\_DP\_pvalue=results.pvalues['liqulity1']  
if rg\_DP\_pvalue<=0.01:  
 jud='在1%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue>0.01)&(rg\_DP\_pvalue<=0.05):  
 jud='在5%的显著性水平下有样本内预测能力'  
elif (rg\_DP\_pvalue > 0.05) & (rg\_DP\_pvalue <= 0.1):  
 jud = '在10%的显著性水平下有样本内预测能力'  
else:  
 jud='无样本内预测能力'  
print('In-sample tests for one factor model with OLs:')  
print('Predictor:{:s}'.format(factor))  
print('Regressing Results:b={:f},k={:f}'.format(rg\_con,rg\_DP))  
print('Regressing Results:p={:f},p={:f}'.format(rg\_con\_pvalue,rg\_DP\_pvalue))  
print('Inference:{:s}'.format(jud))  
data['yyyymm']=data['yyyymm'].astype('int64')  
# #样本外检验  
# #单因子模型：OLS线性拟合  
factor\_out='VOL'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','VOL','VOL1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=201012)  
n\_out=np.sum(datafit['yyyymm']>201012)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~VOL1',data=datafit[['ExRet','VOL1']].iloc[:(n\_in+i),:])#往前滚先用n\_in数据预测n\_in+i的数据  
 results=model.fit()  
 b=results.params['Intercept']#截距  
 k=results.params['VOL1']#斜率  
 f=datafit['VOL'].iloc[n\_in+i-1]#因子值  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]#真实收益率  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]#无风险利率  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)#波动率最近12个月的收益率平方和  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
# 'Turnover','Ret','Rfree','PE',  
# # 'EPS','ROE','IncomePS','yyyymm','波动率','high','rskew','Beta','liqulity','ExRet'  
  
factor\_out='Trdsum'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','Trdsum','Trdsum1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=201011)  
n\_out=np.sum(datafit['yyyymm']>201011)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~Trdsum1',data=datafit[['ExRet','Trdsum1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['Trdsum1']  
 f=datafit['Trdsum'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
  
factor\_out='Turnover'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','Turnover','Turnover1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=201012)  
n\_out=np.sum(datafit['yyyymm']>201012)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~Turnover1',data=datafit[['ExRet','Turnover1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['Turnover1']  
 f=datafit['Turnover'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
# # 'EPS1','ROE1','IncomePS1','波动率1','high1','rskew1','Beta1','liqulity1','ExRet1'  
factor\_out='PE'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','PE','PE1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=201012)  
n\_out=np.sum(datafit['yyyymm']>201012)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~PE1',data=datafit[['ExRet','PE1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['PE1']  
 f=datafit['PE'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
# 'EPS1','ROE1','IncomePS1','波动率1','high1','rskew1','Beta1','liqulity1','ExRet1'  
factor\_out='EPS'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','EPS','EPS1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=201012)  
n\_out=np.sum(datafit['yyyymm']>201012)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~EPS1',data=datafit[['ExRet','EPS1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['EPS1']  
 f=datafit['EPS'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
# ,'IncomePS1','波动率1','high1','rskew1','Beta1','liqulity1','ExRet1'  
factor\_out='ROE'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','ROE','ROE1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=201012)  
n\_out=np.sum(datafit['yyyymm']>201012)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~ROE1',data=datafit[['ExRet','ROE1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['ROE1']  
 f=datafit['ROE'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
  
factor\_out='IncomePS'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','IncomePS','IncomePS1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=201012)  
n\_out=np.sum(datafit['yyyymm']>201012)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~IncomePS1',data=datafit[['ExRet','IncomePS1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['IncomePS1']  
 f=datafit['IncomePS'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
  
factor\_out='波动率'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','波动率','波动率1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=201012)  
n\_out=np.sum(datafit['yyyymm']>201012)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~波动率1',data=datafit[['ExRet','波动率1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['波动率1']  
 f=datafit['波动率'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
  
factor\_out='high'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','high','high1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=200512)  
n\_out=np.sum(datafit['yyyymm']>200512)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~high1',data=datafit[['ExRet','high1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['high1']  
 f=datafit['high'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
# ,'liqulity1','ExRet1'  
factor\_out='rskew'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','rskew','rskew1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=200512)  
n\_out=np.sum(datafit['yyyymm']>200512)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~rskew1',data=datafit[['ExRet','rskew1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['rskew1']  
 f=datafit['rskew'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
factor\_out='Beta'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','Beta','Beta1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=200512)  
n\_out=np.sum(datafit['yyyymm']>200512)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~Beta1',data=datafit[['ExRet','Beta1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['Beta1']  
 f=datafit['Beta'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
  
factor\_out='liqulity'  
datafit=data[['yyyymm','Ret','Rfree','ExRet','liqulity','liqulity1']].copy(deep=True)  
n\_in=np.sum(datafit['yyyymm']<=200512)  
n\_out=np.sum(datafit['yyyymm']>200512)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
for i in range(n\_out):  
 model=smf.ols('ExRet~liqulity1',data=datafit[['ExRet','liqulity1']].iloc[:(n\_in+i),:])#往前滚  
 results=model.fit()  
 b=results.params['Intercept']  
 k=results.params['liqulity1']  
 f=datafit['liqulity'].iloc[n\_in+i-1]  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in + i]  
 rout[i]=k\*f+b  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
print()  
print('Out-of-sample tests for one factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
  
  
  
#样本外检验  
#多因子模型：OLS线性拟合  
factor\_out='PE,Turnover,VOL,EPS,ROE,IncomePS,Beta,liqulity,high,rskew'  
datafit=data.copy(deep=True)  
  
n\_in=np.sum(datafit['yyyymm']<=201012)  
n\_out=np.sum(datafit['yyyymm']>201012)  
rout=np.zeros(n\_out)  
rmean=np.zeros(n\_out)  
rreal=np.zeros(n\_out)  
rfree=np.zeros(n\_out)  
volt2=np.zeros(n\_out)  
  
for i in range(n\_out):  
 model=smf.ols('ExRet~PE1+Turnover1+VOL1+EPS1+ROE1+IncomePS1+Beta1+liqulity1+high1+rskew1',  
 data=datafit[['ExRet','PE1','Turnover1','VOL1','EPS1','ROE1','IncomePS1','Beta1','liqulity1','high1','rskew1']].iloc[:(n\_in+i),:])  
 results=model.fit()  
 k=results.params.values  
 f=datafit[['PE','Turnover','VOL','EPS','ROE','IncomePS','Beta','liqulity','high','rskew']].iloc[n\_in+i-1,:].values#因子数  
 f=np.concatenate((np.array([1]),f))#将常数项放进来  
 rreal[i]=datafit['ExRet'].iloc[n\_in+i]  
 rfree[i]=datafit['Rfree'].iloc[n\_in+i]  
 rout[i]=np.sum(k\*f)  
 rmean[i]=np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i]=np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
  
print()  
print('Out-of-sample tests for multi-factor model with OLS:')  
print('Predictor:{:s}'.format(factor\_out))  
R2os,MFSEadj,pvalue\_MFSEadj=myfun\_stat\_gains(rout,rmean,rreal)  
Uout,Umean,DeltaU=myfun\_econ\_gains(rout,rmean,rreal,rfree,volt2,gmm=5)  
del datafit  
#可能存在多重共线性，使预测能力下降  
  
#样本外检验  
#多因子模型：LASSO回归，Ridge回归，ElasticNet回归  
factor\_out='PE,Turnover,VOL,EPS,ROE,IncomePS,Beta,liqulity,high,rskew'  
factor\_list = np.array(['PE','Turnover','VOL','EPS','ROE','IncomePS','Beta','liqulity','high','rskew'])  
  
datafit = data.copy(deep=True)  
  
n\_in = np.sum(datafit['yyyymm'] <= 201012)  
n\_out = np.sum(datafit['yyyymm'] > 201012)  
rout = np.zeros(n\_out)  
rmean = np.zeros(n\_out)  
rreal = np.zeros(n\_out)  
rfree = np.zeros(n\_out)  
volt2 = np.zeros(n\_out)  
#Ridge  
reg = sklm.RidgeCV(cv=10, fit\_intercept=True, normalize=True)#sklear的包  
for i in range(n\_out):  
 X = datafit[['PE1','Turnover1','VOL1','EPS1','ROE1','IncomePS1','Beta1','liqulity1','high1','rskew1']].iloc[:(n\_in+i), :].values  
 y = datafit['ExRet'].iloc[:(n\_in+i)].values  
 reg.fit(X, y)  
 # print(factor\_list[np.abs(reg.coef\_) != 0])  
 k = np.concatenate((np.array([reg.intercept\_]), reg.coef\_))  
 f = datafit[['PE','Turnover','VOL','EPS','ROE','IncomePS','Beta','liqulity','high','rskew']].iloc[n\_in+i-1, :].values  
 f = np.concatenate((np.array([1]), f))  
 rreal[i] = datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in+i]  
 rout[i] = np.sum(k\*f)  
 rmean[i] = np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i] = np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
  
print()  
print('Out-of-sample tests for multi-factor model with ML method:')  
print('Predictor: {:s}'.format(factor\_out))  
R2os, MFSEadj, pvalue\_MFSEadj = myfun\_stat\_gains(rout, rmean, rreal)  
Uout, Umean, DeltaU = myfun\_econ\_gains(rout, rmean, rreal, rfree, volt2, gmm=5)  
del datafit  
  
#Lasso  
factor\_out='PE,Turnover,VOL,EPS,ROE,IncomePS,Beta,liqulity,high,rskew'  
factor\_list = np.array(['PE','Turnover','VOL','EPS','ROE','IncomePS','Beta','liqulity','high','rskew'])  
  
datafit = data.copy(deep=True)  
  
n\_in = np.sum(datafit['yyyymm'] <= 201012)  
n\_out = np.sum(datafit['yyyymm'] > 201012)  
rout = np.zeros(n\_out)  
rmean = np.zeros(n\_out)  
rreal = np.zeros(n\_out)  
rfree = np.zeros(n\_out)  
volt2 = np.zeros(n\_out)  
reg = sklm.LassoCV(random\_state=0, cv=10, fit\_intercept=True, normalize=True, precompute='auto', copy\_X=True, n\_jobs=-1, max\_iter=10\*\*9, tol=10-6)  
# reg\_lasso = linear\_model.LassoLarsCV(cv=10, fit\_intercept=True, normalize=True, precompute='auto', copy\_X=True, n\_jobs=-1, max\_iter=10000000)  
# reg = sklm.ElasticNetCV(random\_state=0, cv=10, fit\_intercept=True, normalize=True, precompute='auto', copy\_X=True, n\_jobs=-1, max\_iter=10\*\*9, tol=10-6)  
for i in range(n\_out):  
 X = datafit[['PE1','Turnover1','VOL1','EPS1','ROE1','IncomePS1','Beta1','liqulity1','high1','rskew1']].iloc[:(n\_in+i), :].values  
 y = datafit['ExRet'].iloc[:(n\_in+i)].values  
 reg.fit(X, y)  
 # print(factor\_list[np.abs(reg.coef\_) != 0])  
 k = np.concatenate((np.array([reg.intercept\_]), reg.coef\_))  
 f = datafit[['PE','Turnover','VOL','EPS','ROE','IncomePS','Beta','liqulity','high','rskew']].iloc[n\_in+i-1, :].values  
 f = np.concatenate((np.array([1]), f))  
 rreal[i] = datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in+i]  
 rout[i] = np.sum(k\*f)  
 rmean[i] = np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i] = np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
  
print()  
print('Out-of-sample tests for multi-factor model with ML method:')  
print('Predictor: {:s}'.format(factor\_out))  
R2os, MFSEadj, pvalue\_MFSEadj = myfun\_stat\_gains(rout, rmean, rreal)  
Uout, Umean, DeltaU = myfun\_econ\_gains(rout, rmean, rreal, rfree, volt2, gmm=5)  
del datafit  
  
#ElasticNet  
factor\_out='PE,Turnover,VOL,EPS,ROE,IncomePS,Beta,liqulity,high,rskew'  
factor\_list = np.array(['PE','Turnover','VOL','EPS','ROE','IncomePS','Beta','liqulity','high','rskew'])  
  
datafit = data.copy(deep=True)  
  
n\_in = np.sum(datafit['yyyymm'] <= 201012)  
n\_out = np.sum(datafit['yyyymm'] > 201012)  
rout = np.zeros(n\_out)  
rmean = np.zeros(n\_out)  
rreal = np.zeros(n\_out)  
rfree = np.zeros(n\_out)  
volt2 = np.zeros(n\_out)  
reg = sklm.ElasticNetCV(random\_state=0, cv=10, fit\_intercept=True, normalize=True, precompute='auto', copy\_X=True, n\_jobs=-1, max\_iter=10\*\*9, tol=10-6)  
  
for i in range(n\_out):  
 X = datafit[['PE1','Turnover1','VOL1','EPS1','ROE1','IncomePS1','Beta1','liqulity1','high1','rskew1']].iloc[:(n\_in+i), :].values  
 y = datafit['ExRet'].iloc[:(n\_in+i)].values  
 reg.fit(X, y)  
 # print(factor\_list[np.abs(reg.coef\_) != 0])  
 k = np.concatenate((np.array([reg.intercept\_]), reg.coef\_))  
 f = datafit[['PE','Turnover','VOL','EPS','ROE','IncomePS','Beta','liqulity','high','rskew']].iloc[n\_in+i-1, :].values  
 f = np.concatenate((np.array([1]), f))  
 rreal[i] = datafit['ExRet'].iloc[n\_in+i]  
 rfree[i] = datafit['Rfree'].iloc[n\_in+i]  
 rout[i] = np.sum(k\*f)  
 rmean[i] = np.mean(datafit['ExRet'].iloc[:(n\_in+i)].values)  
 volt2[i] = np.sum(datafit['Ret'].iloc[(n\_in+i-12):(n\_in+i)].values\*\*2)  
  
print()  
print('Out-of-sample tests for multi-factor model with ML method:')  
print('Predictor: {:s}'.format(factor\_out))  
R2os, MFSEadj, pvalue\_MFSEadj = myfun\_stat\_gains(rout, rmean, rreal)  
Uout, Umean, DeltaU = myfun\_econ\_gains(rout, rmean, rreal, rfree, volt2, gmm=5)  
del datafit