华东理工大学 2023 –2024 学年第 2 学期

《大数据与金融计算》实验报告

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 $\gamma = 5$。（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。 3. 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 2001-2022），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 $\gamma = 5$。（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）

本研究针对美国与中国股票市场的收益率可预测性进行了深入探究，借鉴并扩展了既有文献的理论框架与方法论。

在对美国股票市场的收益率预测中，我们遵循了 Rapach D E 和 Zhou G 在 2020 年研究中所采用的预测因子。并且采用线性回归，Lasso 回归，岭回归和弹性网络多种模型进行预测，研究发现某些单一因子具有一定的预测能力，多因子联合预测在结合弹性网络某些的情况下具有样本外预测能力。

对中国股票市场的收益率预中也选取了多个预测因子，利用支持向量机、随机森林、神经网络等多种机器学习算法，对于所选取的一只股票进行预测。实验发现对于所选取的股票来说这些因子无法有效地对其进行预测。对此现象从经济角度分析，可能如 R.David Mclean, Jeffrey Pontiff.在 2016 的研究中所证明的，这些因子在被提出后被大量金融市场参与者应用于收益率预测中，因此导致因子无效。从机器学习的角度分析，可能是股票收益率数据包含大量噪声和非线性特征，在数据量小的情况下使用这些模型无法很好的学习全部特征，导致预测效果差。

通过本次对美国与中国股票市场收益率可预测性进行的深入研究，我收获了丰富的理论知识与实践经验，深化了对金融市场复杂性的认识，提升了数据分析与建模能力。不仅仅学习了预测因子的选取和构造方法，也对于不同的机器学习模型有了更深的了解，让我学习到了金融时间序列预测中的研究方法。

教师批阅：　　实验成绩：

教师签名:　　日期：

实验报告正文：

（每次实验报告均为一篇小论文，因此，统一按照学术论文的要求完成实验报告正文，应包括：题目、摘要、文献综述、模型和方法、结果和讨论、参考文献、附录，具体格式如下：

中国股市收益率可预测性的实证检验

摘要：收益率预测一直是金融实证研究领域中的热门议题，对其研究主要分为样本内预测能力检验和样本外预测能力检验。样本内主要检验因子对于收益率的解释能力，样本外检验考察因子对于未来收益率的预测能力，可分为统计检验和经济意义检验。本研究旨在分析美国与中国股票市场收益率的可预测性，选取对于市场的多个潜在的预测因子，结合机器学习技术，通过实证检验发现了在美国市场中所选取的因子具有一定的预测能力，但是在中国市场所选取的因子对于选取的股票基本无预测能力。

1 文献综述

金融预测一直是金融领域所瞩目的焦点问题。作为金融市场中的重要组成部分，由于股票市场潜在的高额利润与重要金融价值，大量投资者和学者致力于研究股市中的预测问题[1]。有效的金融预测不仅可以增加金融市场参与者的收益，也为金融发展和决策奠定基础[2]。其中关于收益率预测的研究最受关注。在理论金融的研究中，Fama 提出的有效市场理论认为价格及时地反映了市场所有信息，因而无法通过过去的信息预测未来的收益率。然而现实中真正有效的市场往往并不存在，这意味着人们可以找到某些历史信息来预测未来的收益率，因此研究者们不断寻找有效的历史信息，又称为因子。这类研究的基础之一是 FamaFrech 的三因子模型[3]，其提出了市值因子 SMB，账面市值比因子 HML，市场风险因子 MKT，实证研究证明了这三个因子对于市场有较强的解释能力。此后有大量可解释收益率的因子被挖掘，通过实证研究发现收益率和这些因子之间存在一定的规律，这些规律在样本内成立。但是将这些因子应用于样本外预测时并不一定能够取得较好的效果，而样本外预测往往才有现实意义。因此越来越多的研究不仅仅局限于样本内解释能力检验，同时也会进行样本外预测能力的检验。

陈坚在中国股票市场尾部风险与收益率预测研究中，选取了 2001 年 1 月至 2012 年 12 月上证和深证所有的 A 股非 ST 股票的市值加权市场投资组合，使用 Copula 方法和极值理论方法分别构建尾部风险 VaR，研究 VaR 对于收益率的预测能力，实证研究表明，基于极值方法的 VaR 对于中国市场有样本内预测能力，且预测能力最强，但是 Copula 方法所得 VaR 没有解释能力。在样本外预测研究中发现，基于极值的 VaR 单因子在统计意义上优于历史平均值的预测，且预测能力明显高于其他经济变量。最终证明了通过极值 VaR 构建的组合夏普比率最高[4]。陈坚，张轶凡在 2018 的研究中利用高频股指数据

构造了中国市场的已实现偏度，通过实证检验发现该因子在样本内和样本外都具有较好的预测能力[5]。蒋志强, 田婧雯, 周炜星在 2019 年的研究中选取 2006 年 7 月到 2016 年 6 月沪深 A 股的全部股票超额收益率为研究对象，构造不同投资组合并且进行分组，选取了 8 个预测因子：账面市值比、股利分配率、股息价格比、股息收益率、每股收益价格比、现金收益价格比、通货膨胀率、股票波动率。通过研究发现这些因子在不同组合中都有一定的预测能力，并且发现总体市场和行业组合的收益率在牛市阶段的可预测性远强于熊市，但是账面市值比组合和市值组合的则完全相反[6]。Rapach D E, Zhou G.在 2020 年的研究中将组合弹性网络方法应用于收益率预测研究中，使用 S&P 500 指数的月度超额回报作为预测目标，构造了 12 个预测因子，发现组合弹性网络方法预测在统计上意义和经济意义上相比传统模型都有显著的增强[7]。R.David Mclean, Jeffrey Pontiff 在 2016 年的研究中从另一个角度研究收益率的可预测性，其分析了 97 个各个期刊中发表的预测变量。研究比较了每个预测变量在三个不同时间段的回报：原始研究的样本期间、原始样本之后但出版之前的期间，以及出版之后的期间，研究发现学术研究的发表可能会导致投资者利用这些研究进行交易，从而减少了这些股票的预期回报，降低因子的预测能力[8]。

本文对于美国和中国股票市场的收益率可预测性进行研究，在对于美国市场的研究中，采用和 Rapach D E, Zhou G.在 2020 年的研究中相同的方法。在对于中国市场的研究，随机选取市场中的一只股票，构造多个因子，利用不同的机器学习方法进行预测性研究。

2 模型和方法

2.1 线性回归

线性回归在机器学习中是一种有监督学习方法，也是一种常用的数据预测与分析方法，其被用来描述因变量与一个或多个自变量之间的线性关系，即因变量可以通过自变量的线性组合来表示。线性回归模型可以表示为：

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \tag{1}$$

式 1 中，x_i为自变量，y 是因变量，$w_i(i > 0)$表示各自变量的权重，w_0为常数项，又称为截距。

线性回归的损失函数被定义为真实值和拟合值之间的平方误差之和：

$$L(y, \hat{y}) = \sum_{i=1}^{n} \quad (y_i - \hat{y}_i)^2 \tag{2}$$

式中，$\hat{y}$是模型拟合值，y 是真实值。

其参数的确定采用最小二乘法，即通过最小化这个损失函数来找到最佳的模型参数，使得模型预测值与真实值之间的差异平方和达到最小。

线性回归模型形式简单，容易理解。它假设因变量与自变量之间存在简单线性关系，使得模型易于解释。同时其计算效率高，特别是在数据量较大的情况下，由于其优化目标的解析解可得，训练过程通常比许多其他复杂模型更快。

2.2 Lasso

Lasso 回归是一种线性回归分析方法，它通过引入 L1 正则化项对模型进行约束。L1 正则化是一种防止模型过拟合的方法，在 L1 正则化中，引入了 L1 范数（又称正则化项），将 L1 范数加入损失函数中，使得损失函数变为公式 3 所示：

$$Loss = \sum_{i=1}^{m} (\hat{y_i} - y_i)^2 + \lambda(|w_1| + |w_2| + \ldots + |w_n|) \quad (3)$$

式 3 中，$\hat{y_i}$是预测值，y_i是真实值，w_i是各个系数。

Lasso 回归作为线性模型，其优化过程是使损失函数最小化。模型的过拟合往往是模型参数相对输入数据过于复杂，数据样本不足以代表全部特征，但是这些局部特征被模型学习到，导致其在样本外预测中出现较大的偏差。在损失函数中加入了正则化项后，使得损失函数增大，为了降低损失函数，除了使预测值更接近真实值之外，还需要降低模型参数的大小，甚至丢弃某些参数(参数系数为 0)，可以降低模型的复杂度，降低过拟合风险。

2.3 岭回归

岭回归也是一种引入正则化操作的线性模型，然而岭回归引入的是 L2 正则化，其损失函数中加入了 L2 正则化项，可以表示为公式 4。

$$Loss = \frac{1}{2m}\sum_{i=1}^{m} (\hat{y_i} - y_i)^2 + \alpha \sum_{j=1}^{n} \theta_j^2 \quad (4)$$

式 4 中，$\hat{y_i}$是预测值，y_i是真实值，θ_j^2是各个系数的平方。

岭回归通过最小化损失函数获取线性模型的参数从而进行预测。采用 L2 正则化的好处是 L2 正则化倾向于将参数值缩小到接近零但不为零，而 L1 正则化会将参数推向 0，使参数变得稀疏。下面是对此现象的解释：

假设没有加入正则化项的损失函数对于某个参数的偏导在 w=0 处的值为 d_0，加入 L1 正则化项后，在 w=0 处其偏导为 d_0-λ或者 d_0+λ，而 L2 正则化在 w=0 处偏导为 d_0+$2 \times \alpha \times w$ =d_0，可见在 w=0 处的导数仍是 d_0，然而在 L1 正则化中，在 w=0 处会有一个突变，如果 d_0-λ和 d_0+λ异号，则在 0 处存在一个极小值点，导致模型可能取该极小值点作为最终优化结果，容易使参数为 0。

2.4 弹性网络

弹性网络(ElasticNet)是一种统计建模方法，特别适用于线性回归问题，其设计目的是为了综合 Lasso 回归（L1 正则化）和岭回归（L2 正则化）的优点，同时克服它们各自的局限性。其同时将 L1 范数和 L2 范数正则化引入损失函数，如公式 5 所示。

$$Loss = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y})^2 + \lambda(\sum_{j=1}^{n}|w_j| + (1-\alpha)\sum_{j=1}^{n}w_j^2) \quad (5)$$

式 5 中$\hat{y_i}$是预测值，y_i是真实值，θ_j^2是各个系数的平方。w_j是系数。

其综合了 Lasso 和岭回归的优势，适应性更强。通过调整 α 参数，可以在 Lasso 和 Ridge 之间进行平滑过渡，根据数据特性和需求选择合适的正则化方式。当特征之间存在高度相关性时，Lasso 可能仅选择其中一个相关特征，而忽略其他有价值的信息。Elastic Net 通过同时施加 L1 和 L2 惩罚，能够保留一组相关特征，避免了这种“全部或全无”的选择问题。

2.5 R^2os 和 MSFE-adjusted 统计量

R^2os 统计量和 MSFE-adjusted 统计量都是从统计角度检验模型是否具有样本外预测能力的统计量，其中 R^2os 的计算如公式 6 所示：

$$R_{\mathrm{OS}}^2 = 1 - \frac{\sum_{i=s}^{n}(r_i - \hat{r_i})^2}{\sum_{i=s}^{n}(r_i - \overline{r}_i)^2} \quad (6)$$

式 6 中r_i是收益率的真实值，$\hat{r_i}$是收益率的预测值，$\overline{r}_i$是历史收益率的平均值。

从公式 6 中可以看出，R2os 统计量使用收益率预测与未来真实值的误差和历史收益率样本的平均值和未来真实值的误差进行比较，如果该统计量的值大于 0，则表示模型预测值的误差小于采用历史样本平均值的误差，否则则表明模型预测误差不如直接采用历史样本平均。

MSFE-adjusted 统计量可以由如下计算过程得出：首先计算d_i，其计算如公式 7 所示：

$$d_i = (r_i - \overline{r}_i)^2 - [(r_i - \hat{r_i})^2 - (\overline{r}_i - \hat{r_i})^2], i = s, \cdots, n \quad (7)$$

接着将d_i和样本编号 i 进行线性回归，对回归得到的常数项进行 t 检验，所得 t 检验统计量就是 MSFE-adjusted 统计量。

3 结果与讨论

3.1 美国市场收益率可预测性研究

3.1.1 因子选择

选取 Time-series and cross-sectional stock return forecasting: New machine learning methods 论文[]中的 12 个因子来预测标普 500 超额收益率。表 1 是因子介绍：

表 1 因子表

因子名称	计算
对数股息价格比 DP	标普 500 市场中的 12 个月移动股息和的对数除以标普 500 指数对数值
对数盈利价格比 EP	标普 500 市场中的 12 个月移动盈利的对数除以标普 500 指数对数值
波动率 VOL	第 t 个月的年化波动率
国库券收益率 Bill	三个月国库券收益率减去三个月国库券收益率的 12 个月移动平均值
长期国债收益 BOND	十年期国债收益率减去十年期国债收益率的 12 个月移动平均值
期限价差 Term	10 年期国库券和 3 个月期国库券的收益率之差
信用利差 Credit	AAA 级公司债券和十年期国债的收益率之差
通货膨胀 PPIG	生产者价格指数(PPI)通货膨胀率
工业生产增长 IPG	工业生产的增长率
MA(1,12)	标准普尔 500 价格指数大于或等于(小于)标准普尔 500 价格指数的 12 个月移动平均值，它的值为 1(零)
MA(3,12)	标准普尔 500 价格指数的三个月移动平均值大于或等于(小于)标准普尔 500 价格指数的 12 个月移动平均值，则该指标变量的值为 1(零)。
动量技术信号 MOM(6)	标准普尔 500 指数大于或等于(小于)六个月前的价格指数，它的值为 1(零)

3.1.2 单因子样本内解释能力检验

样本内解释能力通过线性回归方法检验，将单个因子的时序数据作为自变量 X 矩阵，标普 500 收益率作为因变量 Y 矩阵进行线性回归，对于因子系数的显著性水平进行检验，如果显著则认为该因子有解释能力。

表 2 是选取的 12 个因子的样本内解释能力检验结果：

表 2 因子样本内预测能力检验表

因子	系数	T 统计量 P 值
DP	0.0066	0.0560*
EP	0.0080	0.0385**
VOL	0.0256	0.1493
Bill	-0.2417	0.2548
BOND	-1.2917	0.0005***
Term	0.1198	0.3362
Credit	0.2784	0.4746
PPIG	-0.0006	0.6649
IPG	0.0008	0.3677
MA(1,12)	0.0051	0.1505
MA(3,12)	0.0038	0.2712

MOM(6)	0.0043	0.2054

注：*表示 10%水平下显著，**表示在 5%水平下显著，***表示在 1%水平下显著。

从表 2 中可以看出 BOND 因子在 1%显著性水平下显著，EP 因子在 5%显著性水平下显著，DP 因子在 10%显著性水平下显著，这说明这三个因子对于样本内的收益率有解释能力。

3.1.3 单因子样本外预测能力检验

样本外预测是指使用过去的因子数据预测下一期的数据，即使用 t-1 期之前的数据预测第 t 期，检验样本外预测能力的方法有两种：统计检验以及经济意义检验。统计检验指的是使用 R2os 等统计量进行检验，经济检验是指通过预测得到的收益率在风险资产与无风险资产之间构造投资组合，再通过平均值构造组合，计算两个组合的效用并且相减得到检验量，最后设定阈值检验是否有经济意义，在本研究中，阈值设定为 10^-6，小于这个阈值则没有经济意义。统计量的含义已在 2.6 以及 2.7 中说明，表 3 是各因子样本外预测能力检验表。

表 3 因子样本内预测能力检验表

因子名称	R2os	MFSEpvalue	经济意义
DP	-0.005143	0.039526**	无
EP	-0.015106	0.640065	无
VOL	0.004113	0.036293**	无
Bill	-0.000322	0.005307*	有
BOND	0.011894	0.001961***	有
Term	-0.001091	0.080391*	有
Credit	-0.000930	0.462316	有
PPIG	-0.001019	0.388908	无
IPG	0.000464	0.341275	有
MA(1,12)	-0.001704	0.828172	有
MA(3,12)	-0.001900	0.890707	有
MOM(6)	-0.001505	0.623818	有

注：*表示 10%水平下显著，**表示在 5%水平下显著，***表示在 1%水平下显著。

从表 3 中可以看出只有 BOND 因子在统计检验和经济检验两个检验中都展现出样本外预测能力，DP，EP，PPIG 三个因子在两个检验中都没有样本外预测能力，剩下的因子只在一个检验中具有样本外预测能力。和样本内检验结合起来看，具有样本内解释能力的因子在样本外不一定具有预测能力。从表 3 中本身可以看出统计检验结果和经济检验结果是不相关的，很多因子会出现只有一个检验有效的情况。

3.1.4 多因子样本外预测能力检验

3.1.1 到 3.1.3 都是对单个因子进行样本内和样本外检验，且都是采用的线性回归方法，在本小节研究中对这 12 个因子联合预测能力进行检验，同时又采用 Lasso，Ridge 和 ElasticNet 方法进行检验。研究结果如表 4 所示：

表 4 多因子样本外预测能力检验表

模型名称	R2os	MFSEpvalue	经济意义
Linear Regression	-0.041573	0.004093***	有
Lasso	-0.014242	0.884220	无
Ridge	-0.022879	0.183879	无
ElasticNet	0.010729	0.025683**	有

注：*表示 10%水平下显著，**表示在 5%水平下显著，***表示在 1%水平下显著。

从表 4 中可以看出运用弹性网络(ElasticNet)联合 12 个因子对收益率进行预测时在统计检验和经济检验上都可以取得较好的效果，使用线性模型虽然在无法通过统计检验，但是预测出的收益率仍然具有经济意义。Lasso 和 Ridge 在统计检验和经济检验上表现都不理想。

3.2 中国市场收益率可预测性研究

本节对于中国股票市场的收益率可预测性进行研究，研究采用 600694 大商股份公式的超额收益率为预测目标，构造了月超额收益、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，月已实现偏度这 11 个因子用于预测。其中月波动率计算方法为月内日收益率的平方和，月流动性的计算方法为月收益率除以月成交额的对数的绝对值，月股价高点的计算方法为当月股价最高值与前三个月股价的最大值的比值，月以实现偏度的计算方法如公式 8 和 9 所示。

$$RDVar_t = \sum_{i=1}^{N} r_{it}^2 \tag{8}$$

$$RDVSkew_t = \frac{\sqrt{N}\sum_{i=1}^{N} r_{it}^3}{RDVar_t^{3/2}} \tag{9}$$

公式 8 利用月内每日收益率计算日收益率的平方和，随后使用公式 9 计算月已实现偏度。

本节研究采用线性回归(Linear Regression)，Lasso，ElasticNet，支持向量回归(SVR)，决策树回归(DecisionTreeRegression)对样本外预测能力进行检验。

首先是各个单一因子的预测能力检验，使用线性回归模型，结果如表 5 所示。

表 5 单一因子样本外预测能力检验表

因子	R2os	MFSEpvalue	经济意义
流动性	-0.002858	0.655338	无
换手率	-0.179068	0.337637	无
超额收益	-0.007077	0.308106	无
市盈率	-0.003625	0.349348	无
月每股收益	-0.046999	0.466296	无
ROE	-0.015978	0.144804	无
每股营业收入	-0.017949	0.292283	无
波动率	-4.492966	0.794808	无
已实现偏度	-0.008051	0.318928	无
股价高点	-0.024609	0.364748	无
β 系数	0.003286	0.686811	无

从表 5 中可以看出选取的因子在单个预测的情况均没有预测能力。

接着使用多个模型对于多因子联合预测进行检验，结果如表 6 所示。

表 6 多因子样本外预测能力检验表

模型	R2os	MFSEpvalue	经济意义
Linear	-39.880869	0.642763	无
Lasso	-0.025021	0.919505	无
Ridge	-0.234877	0.085647	无
ElasticNet	-0.003178	0.332732	无
SVR	-0.053376	0.522272	无
DecisionTree	-1.706495	0.623001	无

从表 6 中可以看出多因子联合预测的情况下所有模型都无法进行有效地的样本外预测，因此对于 600694 大商股份来说这些因子无法有效地对其进行预测，从经济角度分析，可能如 R.David Mclean, Jeffrey Pontiff.在 2016 的研究中所证明的[8]，这些因子在被提出后被大量金融市场参与者应用于收益率预测中，因此导致因子无效。从机器学习的角度分析，可能是股票收益率数据包含大量噪声和非线性特征，在数据量小的情况下使用这些模型无法很好的学习全部特征，导致预测效果差。

4 参考文献

[1] Shaveta Dargan et al. A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning[J]. Archives of Computational Methods in Engineering, 2019, 27(4) : 1-22.

[2] J. B. Heaton and N. G. Polson and J. H. Witte. Deep learning for finance: deep portfolios[J].

Applied Stochastic Models in Business and Industry, 2017, 33(1) : 3-12.
[3] Fama, E. F., and Kenneth, R. (1993). French, 1993, Common risk factors in the returns on stocks and bonds. Journal of Financial Economics, 33(1), 3-56.
[4] 陈坚. 中国股票市场尾部风险与收益率预测——基于 Copula 与极值理论的 VaR 对比研究[J]. 厦门大学学报(哲学社会科学版), 2014(04):45-54.
[5] 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.
[6] 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.
[7] Rapach D E, Zhou G. Time‐series and Cross‐sectional Stock Return Forecasting: New Machine Learning Methods[M]. John Wiley & Sons, Ltd, 2020.
[8] R.David Mclean, Jeffrey Pontiff. Does Academic Research Destroy Stock Return Predictability?[J]. The Journal of Finance, 2016, 71(1) : 5-31.

5 附录

```
import numpy as np
import pandas as pd
from math import pi

import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.linear_model import *
import matplotlib.pyplot as plt
from sklearn.svm import SVR
from sklearn.preprocessing import normalize

from Util import *
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'].shift(periods=12)
data['MA312'] = data['Index'].rolling(window=3).mean() >=
data['Index'].rolling(window=12).mean()
data['MOM6'] =  data['Index'] >= data['Index'].shift(periods=6)
data['ExRet'] = data['CRSP_SPvw'] - data['Rfree']
```

```
data[['MA112', 'MA312', 'MOM6']] = data[['MA112', 'MA312',
'MOM6']].astype(int)

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(1)),axis=1)
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)
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)
    fit_model = model.fit()
    MFSEadj = fit_model.tvalues[0]
    p_valueMFSE = fit_model.pvalues[0]

    if R2os>0 and p_valueMFSE<=0.01:
        jud = "在 1%的显著性水平下有样本外预测能力"
    elif R2os>0 and p_valueMFSE>0.01 and p_valueMFSE<=0.05:
        jud = "在 5%的显著性水平下有样本外预测能力"
    elif R2os>0 and p_valueMFSE>0.05 and p_valueMFSE<=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, MFSEadj, p_valueMFSE))
    print('Inference: {:s}'.format(jud))
    return R2os, MFSEadj,p_valueMFSE
def myfun_econ_gains(rout,rmean,rreal,rfree,volt2,gmm=5):
    omg_out = rout/volt2/gmm
    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
    if DeltaU<10**-6:
        jud = "没有经济意义"
    else:
        jud = "有经济意义"
    print('Econ Gains: Delta U = {:f}, Upred = {:f}, Umean =
{:f}'.format(DeltaU, Uout, Umean))
    print('Inference: {:s}'.format(jud))
    return Uout,Umean,DeltaU



factor_out = 'DP, EP, VOL, BILL, BOND, TERM, CREDIT, PPIG, IPG, MA112, MA312,
MOM6'
datafit = data.copy(deep=True)
# muti_out_predict(factor_out,datafit)
factor_out = 'DP, EP, VOL, BILL, BOND, TERM, CREDIT, PPIG, IPG, MA112, MA312,
MOM6'
datafit = data.copy(deep=True)
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)


reg = 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)
    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))
    rout[i] = np.sum(k*f)
    rmean[i] = np.mean(datafit['ExRet'].iloc[:n_in+i].values)
    rreal[i] = datafit['ExRet'].iloc[n_in+i]
    rfree[i] = datafit['Rfree'].iloc[n_in+i]
    volt2[i] = np.sum(datafit['Ret'].iloc[(n_in+i-12):(n_in+i)].values**2)

print('Out-of-sample tests for multi-factor model with ML method:')
print('Predictor: {:s}'.format(factor_out))
R2os, MSFEadj, pvalue_MSFEadj = myfun_stat_gains(rout, rmean, rreal)
Uout, Umean, DeltaU = myfun_econ_gains(rout, rmean, rreal, rfree, volt2,
gmm=5)

import numpy as np
import pandas as pd
import pickle
import datetime
from dateutil.relativedelta import relativedelta
with open("date.pkl","rb") as f:
    date = pickle.load(f)
#日度数据
daily_data1 = pd.read_excel("2001-2010.xls")
daily_data2 = pd.read_excel("2011-2015.xls")
daily_data3 = pd.read_excel("2016-2020.xls")
daily_data4 = pd.read_excel("2021-2023.xls")
daily_data =
pd.concat((daily_data1,daily_data2,daily_data3,daily_data4),axis=0)
daily_data.columns = ['code','name','date','close']
daily_data.dropna(inplace=True)
daily_data['date'] = pd.to_datetime(daily_data['date'])
daily_data.sort_values(by="date",inplace=True)
daily_data['return'] = np.log(daily_data['close']) -
np.log(daily_data['close'].shift(1))
daily_data.dropna(inplace=True)
date0  =pd.to_datetime("20010131")
#计算月波动率
df = pd.DataFrame()
month_list = []
bodong_list = []
piandu_list = []
for i in range(len(date)-1):
    date_s = date[i]
    date_b = date[i+1]
```

```
    data_month = daily_data.loc[(daily_data['date']>pd.to_datetime(date_s)) &
(daily_data['date']<=pd.to_datetime(date_b))]
    #计算波动率
    r = data_month['return']
    r2 = r**2
    bodong_i = np.sum(r2)
    if bodong_i == 0.0:
        continue
    #计算月已实现偏度
    r3 = r**3
    r3 = np.sqrt(len(r3)) * np.sum(r3)
    RDSkew = r3/(np.sqrt(bodong_i ** 3))
    month_list.append(date_b)
    bodong_list.append(bodong_i)
    piandu_list.append(RDSkew)
df['date'] = month_list
df['bodong'] = bodong_list
df['piandu'] = piandu_list
df.dropna(inplace=True)
# 计算月股价高点
gaodian = []
date_list2 = []
dfg = pd.DataFrame()
for i in range(0,len(date)-4):
    date_s = date[i]
    date_m = date[i+3]
    date_b = date[i+4]
    daily_data_1_3 = daily_data[(daily_data['date']>pd.to_datetime(date_s)) &
(daily_data['date']<=pd.to_datetime(date_m))]
    daily_data_4 = daily_data[(daily_data['date']>pd.to_datetime(date_m)) &
(daily_data['date']<=pd.to_datetime(date_b))]
    #计算前三个月股价最高点
    max_1_3 = np.max(daily_data_1_3['close'])
    #当月股价最高点
    max_4 = np.max(daily_data_4['close'])
    #计算比值
    result = max_4/max_1_3
    gaodian.append(result)
    date_list2.append(date_b)
dfg['date'] = date_list2
dfg['gaodian'] = gaodian
dfg.dropna(inplace=True)
final_df = pd.merge(left=df,right=dfg,on='date')
print(final_df)
```

```
with open("data_daily.pkl","wb") as f:
    pickle.dump(final_df,f)

import pandas as pd
import numpy as np
import pickle
from sklearn.linear_model import *
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from Util import *
# with open("date.pkl","wb") as f:
#     pickle.dump(data_from_month['date'],f)
with open("data_daily.pkl","rb") as f:
    data_from_daily = pickle.load(f)
data_from_month = pd.read_excel("月数据.xls")
data_from_month.columns =
['code','name','date','close','trdsum','turn','return','rf','pe','eps','roe','
income']
data_from_month['return'] = data_from_month['return'] - data_from_month['rf']
#beta 数据
beta = pd.read_excel("月 beta.xls")
beta.columns = ['code','name','date','beta']
final_data =
pd.merge(left=data_from_month,right=data_from_daily,on='date',how='inner')
final_data =
pd.merge(left=final_data,right=beta[['date','beta']],on='date',how='inner')
final_data.dropna(inplace=True)
final_data['trdsum'] =
np.abs(final_data['return']/np.log(final_data['trdsum']))

final_data['return1'] = final_data['return'].shift(-1)
final_data.dropna(inplace=True)
model_name =
["Linear","Ridge","ESnet","SVR","DSTree","KNN","Lasso","MLP","GBRT"]
model_name = "DSTree"
if model_name=="Linear":
    model = LinearRegression()
elif model_name=="Ridge":
    model = RidgeCV(cv=10)
elif model_name=="ESnet":
    model = ElasticNet(
        alpha=1,              # 正则化强度，或 `l1_ratio` 控制 L1/L2 比例
        l1_ratio=0.8,           # L1 正则化相对于 L2 正则化的权重
```

```
        fit_intercept=True,     # 是否包含截距项
        normalize=False,        # 是否对数据进行标准化
        max_iter=10000,         # 最大迭代次数
        tol=1e-5,               # 收敛阈值
        random_state=42,
    )
elif model_name=="SVR":
    model = SVR()
elif model_name=="DSTree":
    model = DecisionTreeRegressor()
elif model_name=="KNN":
    model = KNeighborsRegressor()
elif model_name=="Lasso":
    model = LassoCV(cv=5)
train_data = final_data.loc[0:int(len(final_data)*0.5),'trdsum':'return11']
#多因子检验
predict_list = []
real_list = []
mean_list = []
volt_list = []
rf_list = []
factor_list =
['trdsum','turn','return','pe','eps','roe','income','bodong','piandu','gaodian
','beta']
first = 0
end = 11
print(factor_list[first:end])
for i in range(int(len(final_data)*0.5)+1,int(len(final_data))-2):
    train_data = np.array(final_data.loc[0:i,factor_list[first:end]])
    train_label = np.array(final_data.loc[0:i,'return11'])
    train_label = np.expand_dims(train_label,axis=1)
    fit_model = model.fit(train_data,train_label)
    test_data = np.array(final_data.loc[i+1,factor_list[first:end]])
    test_data = test_data.reshape(1,len(test_data))
    #预值
    predict = model.predict(test_data).item()
    predict_list.append(predict)
    #真实值
    real = final_data.loc[i+1,'return11']
    real_list.append(real)
    mean = np.mean(final_data.loc[0:i+1,'return'])
    mean_list.append(mean)
    #volt
    if i<12:
```

```
        volt = np.sum(final_data.loc[0:i,'return'].values**2)
        volt_list.append(volt)
    else:
        volt = np.sum(final_data.loc[i-12:i,'return'].values**2)
        volt_list.append(volt)
    #无风险利率
    rf = final_data.loc[i+2,'rf']
    rf_list.append(rf)
predict_np = np.array(predict_list)
real_np = np.array(real_list)
mean_np = np.array(mean_list)
volt_np = np.array(volt_list)
rf_np = np.array(rf_list)
print(factor_list[first:end])
myfun_stat_gains(predict_np,mean_np,real_np)
myfun_econ_gains(predict,mean_np,real_np,rf_np,volt_np)
```