学校代码：10255
学号：2191712

东华大学

基于 XGBoost 的高频交易选股研究

RESEARCH ON STOCK SELECTION OF HIGH FREQUENCY TRADING BASED ON XGBOOST

学科专业：应用统计
作者姓名：王怡宁
指导教师：胡良剑
答辩日期：2021 年 5 月 22 日

东华大学学位论文原创性声明

本人郑重声明：我恪守学术道德，崇尚严谨学风。所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已明确注明和引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品及成果的内容。论文为本人亲自撰写，我对所写的内容负责，并完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：　　年　　月　　日

东华大学

硕士学位论文答辩委员会成员名单

姓名	职称	职务	工作单位	备注
朱仲义	教授	答辩委员会主席	复旦大学	
刘二谋	高级工程师	答辩委员会委员	深圳市天软科技开发有限公司	
宋晖	教授	答辩委员会委员	东华大学	
刘晓强	教授	答辩委员会委员	东华大学	
杜明	副教授	答辩委员会委员	东华大学	
钟德俊	讲师	答辩委员会秘书	东华大学	

基于 XGBoost 的高频交易选股研究

摘 要

中国股市的飞速发展吸引了越来越多投资者的目光，人们也开始注意到量化投资所具有的纪律性、准确性、时效性和系统性等独特优势，为大量的投资机构带来利好。同时伴随着计算机科学的飞速发展，人工智能与各个学科的融合逐渐变得普遍，越来越多的学者将机器学习方法应用于量化投资当中去。本文旨在将机器学习方法与多因子选股模型相结合，通过引入对日内高频数据的研究，构建基于 XGBoost 算法的量化选股模型。

首先选取沪深 300 指数成分股 2011 年 1 月 1 日至 2020 年 12 月 31 日的 1 分钟数据来构造收益率分布、成交量分布以及量价复合三类高频因子，分别进行单因子有效性检验，并将具有明显选股效果的高频因子加入到多因子选股模型当中，接下来采用 XGBoost 算法构建多因子选股模型，同时使用主成分分析法对数据进行降维，使用网格搜索法进行模型的参数调优，最后进行模型的预测与评价，从而给出收益率前 30 名的股票供投资者参考。

实证结果表明高频因子中携带了传统因子所不具备的新信息，对于提升选股模型预测准确率有明显效果。对高频因子的研究有助于丰富和延伸多因子模型的内涵，促进其理论体系的进一步发展。

同时多因子选股模型在实证检验中也可以促进量化投资理论的不断更新，给予投资者更多新的研究思路，进而对推动我国证券市场的发展，维护证券市场的稳定性和有效性等方面都具有重要的意义。

关键词：高频数据，多因子模型，XGBoost 算法，量化选股

RESEARCH ON STOCK SELECTION OF HIGH FREQUENCY TRADING BASED ON XGBOOST

ABSTRACT

The rapid development of China's stock market has attracted the attention of more and more investors. People began to notice the unique advantages of discipline, accuracy, timeliness, and systematicness of quantitative investment, which brought benefits to a large number of investment institutions. At the same time, with the rapid development of computer science, it has become very common to integrate artificial intelligence with various disciplines. More and more scholars are applying machine learning methods to quantitative investment. This article aims to combine the machine learning method with the multi-factor stock selection model. The stock selection model based on XGBoost is constructed by introducing research on intraday high frequency data.

First of all, we collect the 1-minute data of the constituent stocks of the CSI 300 Index from January 1, 2011 to December 31, 2020 to construct three types of high-frequency factors, including the return distribution factor, the volume distribution factor and the volume-price composite factor. We carry out the single-factor validity test separately, and then add high-frequency factors with obvious stock selection effects to the multi-factor stock selection model. Next, we use the XGBoost to build a multi-factor stock selection model, and principal component analysis to reduce the dimensionality of the data, as well as the grid search method to optimize the parameters of the model. Finally, by predicting and evaluating the model, the top 30 stocks of yield are proposed to the investors.

The empirical results show that the high-frequency factors carry new information that traditional factors do not have, which can significantly improve the accuracy of stock selection models. The study of high-frequency factors will help enrich the connotation of the multi-factor model and promote the further development of its theoretical system. The multi-factor stock selection model can also promote the continuous update of quantitative investment theories in empirical testing, and give investors more new research ideas. Furthermore, it is of great

significance for promoting the development of our country's securities market and maintaining the stability and effectiveness of the securities market.

Wang YiNing (Applied Statistics)

Supervised by Hu LiangJian

KEY WORDS: high-frequency data, multi-factor model, XGBoost, quantitative stock selection

目录

第一章 绪论

1.1 研究背景与意义

1.1.1 研究背景

中国股票交易市场自 1990 年 11 月上海证券交易所成立正式登上历史舞台。在过去这三十多年里飞速发展，吸引了大量的投资者，国家对于的股市的监管也日趋完善。我国在 2010 年 4 月正式推出了第一支股指期货——沪深 300 指数，满足了更多投资者和融资者的需求，同时也填补了我国金融市场不能进行双边交易的空白。这为广大的投资者提供了丰富多样的投资方式，满足了不同投资者对于投资的多样化需求，同时还促进了更多的交易策略的诞生。

量化投资理论最早来源于西方，1952 年著名经济学家马科维兹（Harry M. Markowitz）提出投资组合理论，标志着量化投资这一概念的诞生。巴克莱国际投资管理公司（Barclays Global Investors，简称 BGI）是世界上最早涉猎量化投资的公司，在 1971 年该公司发行了世界上第一支量化基金，这支量化基金被认为是量化投资的鼻祖。自此量化投资手段开始逐渐走入大众的视野，越来越多的投资者将其应用于相关产品的设计当中去。

在量化投资发展的这 50 多年中，学者们相继提出了各种经典理论与投资策略，使得整个量化投资的理论体系逐渐丰富，如资本资产定价模型（1964 年），无风险套利定价理论（1976 年），VAR 风险管理模型（1993 年），Fama-French 三因子（1992 年）、五因子（2013 年）模型，α 系统策略，价值和动量策略、股指期货套利策略以及配对交易策略等。与国外相比，我国的量化投资因金融市场起步较晚等原因出现的也比较晚，直到 21 世纪初期，才发行了第一支量化基金，该基金由华宝信托发行，这标志着我国金融市场正式踏入了量化投资的大门。受限于金融市场政策地制约，国内早期的量化产品大部分以套利策略为主，由于套利策略存在规模限制的问题，所以在此后五年的时间内量化基金的发行一度处于空窗期。直到 2010 年我国正式推出沪深 300 股指期货，为量化基金提供了可行的对冲工具，这一现象才得以缓解，这一年也被称为是中国量化投资元年。在此后的数年中，越来越多的海外投资人员陆续回国，为国内量化投资的发展贡献力量，国内量化投资进入高速发展的时代，同时量化投资也受到越来越多投资者的青睐，正在一步步成为国内资本市场的发展方向。

多因子选股模型是在量化投资领域应用较广泛的选股模型。其基本原理是寻找那些与股票收益率相关程度最高的影响因子，进而将选取出来的因子应用于选股。随着对多因子模型研究的不断深入，因子构造以及选择的技术也越来越成熟，目前的研究则更多的集中在如何让有效的因子组合在一起为投资者带来超额收益，即采用数量化的方法来构造出有效的因子选股策略。因子模型的研究已经从最初由威廉·夏普（William Sharp）提出的单因子模型逐步扩展到 Fama-French 三因子模型、五因子模型以及多因子模型，从线性模型扩展到非线性模型。股票市场往往存在很多复杂的现象，这时传统的线性模型很难全面地挖掘出大量股票数据背后所蕴藏的收益规律。相比较而言，人工智能方法更能够更敏锐地挖掘出非线性数据背后的信息，并为投资者带来稳定的预测能力。从国内的整个量化投资市场来看，将人工智能方法应用于因子挖掘策略地构建，会逐步成为未来量化投资的发展趋势。表 1-1 展示了部分近年来人工智能在量化投资领域的具体应用：

表 1-1 人工智能在量化投资领域的应用

时间	具体应用案例
2007 年	美国公司 Rebellion Research 推出了第一个纯 AI 投资基金
2008 年	中国首个大数据基金广发百发 100 指数基金推出
2008 年	首个智能投顾平台 Wealthfront 诞生
2009 年	管理着 900 亿美元的对冲基金 Cerebellum 使用 AI 进行辅助交易预测，并且自 2009 年以来每年均是盈利的
2011 年	基于 Twitte 信息判断用户情绪的基金 Derwent Capital Markets 成立
2015 年	JP Morgan 与 Sentient 合作创建 AI 策略
2017 年 6 月	华夏基金与微软共同设立微软亚洲研究院，宣布对金融服务领域人工智能的应用展开战略合作研究与探索
2017 年 10 月	我国智能投顾产业首次出海——香港富卫集团与品钦集团进行合作，双方在新加坡成立金融科技公司 PIVOT
2017 年 11 月	工商银行推出智能投顾品牌“A 工投”
2018 年 3 月	贝莱德推出 7 支人工智能 ETF
2019 年 3 月	同花顺旗下资管公司“同花顺阿尔法一号私募证券投资基金”完成私募基金备案
2019 年 12 月	中国首支人工智能 ETF——平安智能 ETF 发行

1.1.2 研究意义

我国证券市场由于诸多因素的影响呈现比较复杂的态势，仅仅通过传统的分析方法，如基本面分析、技术分析等手段，已经不能够满足投资者获得稳定超额收益的需求，而量化投资因其具有独特的稳定性而脱颖而出，成为更多投资者青睐的投资决策方式。与此同时，伴随着计算机科学的飞速发展，人工智能算法、云计算以及大数据等技术也日趋成熟，为量化投资的策略研究提供了新的技术保障。通过选取不同类型的因子来构建多因子模型是量化选股领域中最普遍的方式，即通过探究各类因子与股票未来收益之间存在的关系，来获取更多的超额收益。然而，股票市场往往极其复杂，获取到的股票数据具有非线性、高维度等特点，传统的线性多因子模型在处理这样复杂的数据时有一定的局限性，不能很好地解决股票未来收益率的预测问题。

本文将机器学习方法与量化投资理论相结合，同时加入对高频因子的构建与研究，具体的理论意义与现实意义如下。

（1）理论意义：多因子模型旨在探究不同类型的因子与收益率之间的关系来预测股票价格的走势，对投资者的交易行为做出指导。在对传统因子的研究中，学者们一般选择较低频的数据来进行处理与构造，本文将日内高频数据与低频数据相结合，在传统因子的基础上，加入对于高频因子的研究，探究高频因子的选股能力，为后续的多因子模型带来更多新信息。同时，采用具有良好预测效果的XGBoost 算法进行选股模型的构建，通过机器学习方法发掘出有效因子与股票未来收益之间的内在联系，以丰富和延伸多因子模型的内涵，促进其理论体系的进一步发展。

（2）现实意义：对于各类选股模型的研究最终目的还是为投资者提供更多的投资策略。本文通过实证分析探究了构建的高频因子与 XGBoost 算法在沪深300 指数市场的有效性与可行性，并较为详细的给出了 XGBoost 算法在量化投资过程中的建模流程。同时多因子选股模型在实证检验中也可以促进量化投资理论的不断更新，给予投资者更多新的研究思路，进而对于推动我国证券市场的发展，提高金融市场的投资管理水平以及资产定价水平，维护证券市场的稳定性和有效性等方面都具有很积极的现实意义。

1.2 研究内容与论文框架

1.2.1 研究内容

本文首先综述了因子模型以及机器学习在量化投资中的应用，然后介绍了量化投资以及机器学习相关理论，具体包括 XGBoost 算法原理、模型参数优化方

法以及模型评价方法。实证分析主要以多因子模型和集成学习算法 XGBoost 为理论基础，包含高频因子构建和 XGBoost 量化选股两个部分。

首先引入对高频因子地构建，选取沪深 300 指数成分股 2011 年 1 月 1 日至 2020 年 12 月 31 日共十年的 1 分钟频率数据，构造三类高频因子，分别为收益率分布因子、成交量分布因子以及量价复合因子。其中，收益率分布因子包括高频收益方差、高频收益偏差、高频收益峰度以及高频收益波动分解，成交量分布因子包括日内成交量分布，量价复合因子包括高频量价相关性。对上述构造的因子分别进行单因子有效性检验，通过计算 IC 值（Information Coefficient，信息系数）来衡量单因子的选股效果，最终挑选出具有明显选股效果的高频因子。

由于新因子的研究最终还是要服务于多因子模型，接下来将具有明显选股效果的高频因子加入到多因子模型中，采取 XGBoost 方法进行量化选股，选取沪深 300 指数成分股日频数据来建模分析，输入变量为行情因子、技术指标以及高频因子，目标变量为收益率，最终挑选出收益率前 30 名的股票供投资者参考。

模型构建具体步骤为：首先对选取的数据进行预处理，具体包括缺失值的处理、标准化处理和因子的降维处理。本文采用直接删除的方式来处理缺失值，采用 Min-Max 标准化来解决数据量纲不同的问题，以及采用主成分分析法进行降维。接下来进行模型的训练与参数的优化，使用网格搜索的方法进行调优。最后进行模型的预测与评价，本文采用均方误差（MSE）、均方根误差（RMSE）以及平均绝对误差（MAE）来衡量模型的预测效果。

1.2.2 论文框架

论文一共分为五章，每一章具体内容如下：

第一章为绪论部分，在该部分主要介绍了论文的研究背景与研究意义，探讨了量化投资的起源与发展、多因子模型的广泛应用，同时，从理论意义与现实意义两个方面给出本文的研究意义，最后给出整篇论文的研究内容与行文框架。

第二章为文献综述与涉及理论的介绍，从因子模型和机器学习在量化投资方面的应用两个角度进行综述，同时介绍了文章涉及到的量化投资、机器学习相关理论，具体包括量化选股、XGBoost 算法原理、模型参数优化以及模型评价方法等内容。

第三章为高频因子构建与研究部分，对日内高频数据进行研究，详细介绍了三类共十六个高频因子的构建思路以及具体的计算方式，并从 IC 值的角度分别进行单因子有效性检验，从中选取出具有一定选股效果的高频因子加入到后续的多因子选股模型当中。

第四章为量化选股实证研究部分，选取行情因子、技术指标以及第三章中选取出的具有选股效果的高频因子作为输入指标，股票对数收益率作为输出指标进行 XGBoost 多因子选股。并进行了两个对比研究，一是对加入高频因子和未加

入高频因子的模型预测效果进行对比，探究使用高频因子的必要性；二是对比研究 XGBoost 算法与传统线性回归和逻辑回归模型的预测效果，探究将机器学习方法应用于量化投资领域的必要性。

第五章为总结与展望部分，首先总结了本文的具体工作内容以及研究结果，接下来讨论了本文在研究中的局限性与下一步的工作计划。

本文整体框架如图 1-1 所示：

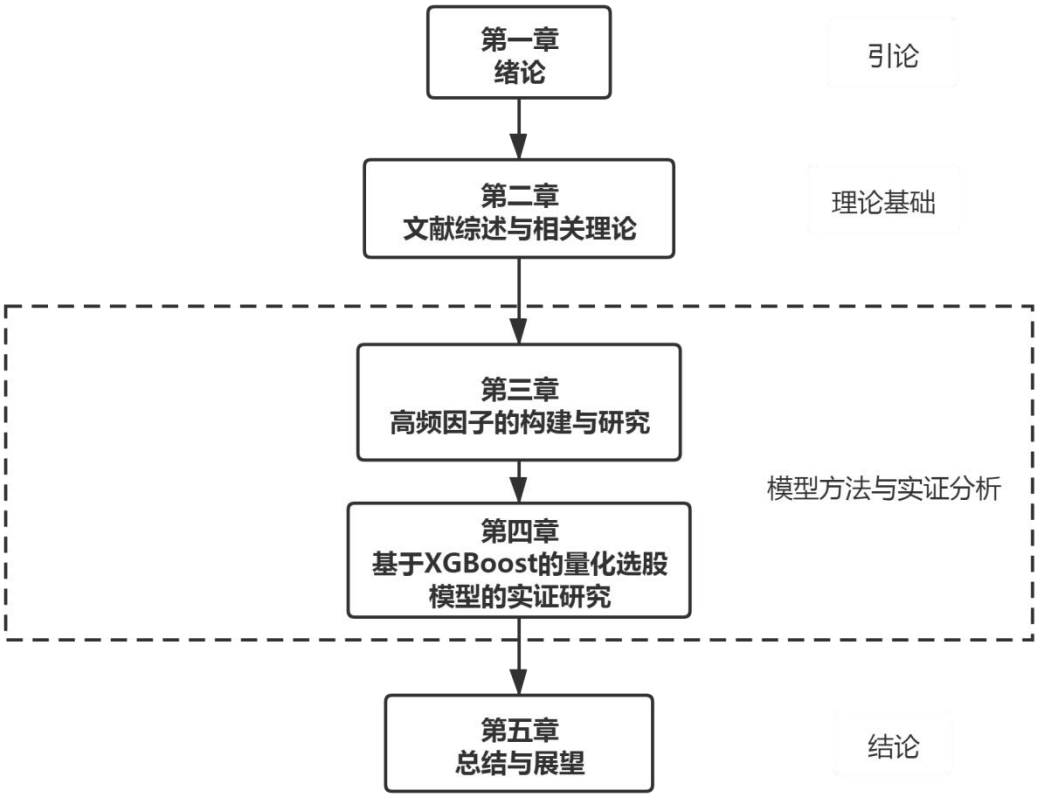

图 1-1 论文整体框架

第二章 文献综述与相关理论

2.1 文献综述

2.1.1 因子模型在量化投资方面的应用

在众多的量化投资策略当中，针对多因子模型的研究与应用可以说是最为广泛的。1964 年学者威廉·夏普（William Sharpe）、约翰·林特纳（John Lintner）、杰克·特雷诺（Jack Treynor）以及简·莫辛（Jan Mossin）共同提出了著名的资本资产定价模型（CAPM），该模型为股票的收益提供了相应的解释。但由于在金融市场中各种各样复杂的情况层出不穷，这就使得人们发现股票中存在超额收益，然而这些超额收益并不能通过市场因子来进行解释，多因子模型应运而生。1993 年 Fama 和 French[1]共同提出了最早的多因子模型——三因子模型，学者们通过对美国证券市场中不同股票收益率差异的决定性因素进行研究发现，市场中的β值并不能有效的解释不同股票之间收益率的差异，然而上市公司的市盈率、市值以及账面市值比例却能够对股票的收益率差异做出合理的解释，因此得到结论，超额收益是对 CAPM 中β值未反映的风险因素地补偿。三因子模型能够为股票市场的各类表现提供比较良好的解释，但随着股票市场的进一步发展，其复杂度日益提高，相继出现许多仅采用三因子模型难以进行解释的现象，因此在 2015 年 Fama 和 French[2]对三因子模型进行了优化，提出了五因子模型，添加了上市公司的盈利水平和投资水平两个因子。在此之后，学者们不断丰富多因子模型理论，陆续将更多因子加入其中，形成了六因子模型、八因子模型等。表 2-1 列举了部分常见的多因子模型：

表 2-1 常见的多因子模型

模型	出处	所含因子
Fama-French 三因子	Fama and Farench（1993）	市场、规模、价值
Carhart 四因子	Carhart（1997）	市场、规模、价值、动量
Novy-Marx 四因子	Novy-Marx（2013）	市场、规模、价值、盈利
Fama-French 五因子	Fama and Farench（2015）	市场、规模、价值、盈利、投资
Hou-Xue-Zhang 四因子	Hou et al	市场、规模、盈利、投资
Stambaugh-Yuan 四因子	Stambaugh and Yuan（2017）	市场、规模、管理、表现
Daniel-Hirshleifer-Su 三因子	Daniel et al（2020）	市场、长周期行为、短周期行为

目前针对多因子的研究已经十分丰富了，主要可以将其分为基本面因子、技术面因子、宏观经济因子以及市场情绪因子四个大类。其中基本面因子主要是对公司的财务状况进行描述，如估值类指标；技术面因子则大多是由过去的价格、成交量以及其他可获得的金融信息所构建的，如超买超卖型因子、均线型因子等；宏观经济因子是基于经济指标来构建的，如 GDP 增速、失业率以及通货膨胀率等，它们几乎会影响到金融市场的每一个角落；市场情绪因子是近几年比较新兴的因子，即投资者的情绪会对股票价格造成一定的波动，如新股发行占比、IPO 收益率因子等。对于如何选择候选因子，一直以来是学者们研究的重点，这主要取决于市场的内在逻辑以及投资者的经验，如何能够选取含有丰富信息的有效因子是构建良好选股模型以及获得超额收益的重要步骤。

Barth、Beaver 和 Landsman[3]（1993）研究结果显示，上市公司的股票价格受到诸多因素的影响，具体包括公司的净利润、资产与负债的账面价值等，同时，公司股票价格的波动与资产规模呈现正相关关系，与负债规模呈现负相关关系，并且这种相关关系并不稳定，是会随着公司的资产以及负债的变动而发生相应的变动的。Piotroski[4]（2000）提出了在多因子选股中使用较普遍的打分法，共选取九个财务指标对股票进行打分，选取的指标主要涉及到公司的盈利能力、资金的来源、资金的流动性、杠杆率以及公司运营效率等方面，根据这九个指标的好坏程度对股票进行打分并按照分数进行排序，选取分数排名靠前的股票构建不同的投资组合，实证研究结果表明基本面数据较好的公司往往有更好的收益以及较小的风险。刘毅[5]（2012）选取 2000 年到 2012 年的股票数据进行实证分析，在成长、估值、质量以及动量因子四类因子中，挑选了 25 个对股票收益率影响较大的因子，最终从中挑选出 8 个具有良好选股效果的因子构建了最优因子组合，同时与其他四类不同的投资组合进行对比，实证结果表明最优因子组合明显优于其他四种投资组合。高鑫[6]（2017）借鉴了 Fama-French 三因子模型的构建原理，将投资者情绪因素融入资产定价模型当中，并在理论的基础上选择两阶段回归模型进行实证分析，实证结果表明情绪因子不仅对资产的定价有着显著的影响，同时能够合理的解释股票市场的各种金融异象，情绪因子与市场、规模、账面市值比、流动性以及动量因子共同成为对资产定价产生重要影响的几大因素。陈志文[7]（2019）在考虑上市公司财务指标的同时，将基于市场的预测因子也作为了备选因子，一方面按照打分法对财务指标进行打分，另一方面给出股票的预测情况，选取全部 A 股来进行实证分析，同时给予预测因子更高的权重，研究结果表明构造的新型选股模型具有良好的表现，得到的投资组合收益率均高于市场平均表现以及 A 股指数。

2.1.2 机器学习在量化投资方面的应用

随着我国经济和计算机科学的飞速发展，大数据时代已经悄然走近，采用传统线性方法来探究股票价格与收益率背后的复杂规律往往得不到令人满意的结果。机器学习方法在面对海量非线性数据有着独特的优势，将其应用于量化投资，既能够为投资者带来更多利好，推动量化投资理论的发展，又顺应了时代发展的新潮流。

Kim 和 Han[8]（2000）构造了基于人工神经网络的股票价格预测模型，采用遗传算法对选取的因子进行降维，研究结果表明该模型在具有良好预测效果的同时还可有进行有效的特征选择。苏治和傅晓媛[9]（2013）采用核主成分分析算法以及遗传算法对支持向量机进行了改进，并选取沪深股票行情数据以及基本面数据进行了回归分析，分别从短期和中长期来对模型预测性能进行测试，研究发现改进后的模型预测准确度更高并且有更加稳定的预测效果，同时该模型针对中长期的预测效果较好。李想[10]（2017）较全面地研究了各类指标，包括财务、红利、动量、规模、估值、宏观、债券以及楼市相关指标共计 307 个，使用 XGBoost 算法进行建模，同时对随机森林、支持向量机和 XGBoost 在模型预测准确率等方面进行了比较，结果表明 XGBoost 预测效果以及稳定性最好。黄恒秋[11]（2016）针对日内高频数据构建了量化择时模型，将沪深 300 指数 1 分钟数据加工成 10 分钟数据，使用 10 分钟数据计算输入特征，股票涨跌情况作为输出特征，构建基于支持向量机的择时模型，实证结果表明该模型在股票呈现下跌趋势时有更好的预测效果。杨健[12]（2017）在实际应用的基础之上，选取机器学习方法构建了一个量化投资策略分析系统，该系统中模块完备，设有股票行情数据库，同时能够实现技术指标的计算以及趋势、分类预测等量化择时功能，通过实践证明该系统能够为投资者提供有效的择时策略。王华琴[13]（2018）对于机器学习方法做了较全面的总结，包括支持向量机、K 近邻、逻辑回归、随机森林、XGBoost 共五种方法，采用不同的方法进行股票涨跌地预测并回测进行对比分析，其中 XGBoost 与随机森林的预测准确率与回测效果最好，其次是支持向量机与 K 近邻方法，逻辑回归预测效果最差。黄志辉[14]（2019）通过构建技术指标来增加时间序列数据维度，将股票时间序列的预测问题转化成为更适合神经网络的二维图像分类问题，同时采用窗口滚动的方法提高数据的利用率，构建基于卷积神经网络的选股模型，并同逻辑回归、BP 神经网络及 LSTM 神经网络的预测效果进行对比，研究发现卷积神经网络算法的预测效果明显优于其他三种方法，能够提供有效的选股策略。

2.2 量化投资相关理论

在我国量化投资是一种比较新的交易方式，最近几年有越来越多的投资者参与到量化投资的行列当中，其主要是通过数量化方式和计算机程序化发出买卖指令。量化投资往往需要依赖大型的投资机构来完成，个人投资者难以实现，一方面是由于其涉及的策略技术较复杂，另一方面则是由于其面对的交易量往往是巨大的，需要先进电脑的运算速度和容量，因此计算机技术的飞速发展也为量化投资提供了便利条件。一般情况下，量化投资交易策略有五大类型，具体包括：量化选股、量化择时、无风险套利、统计套利以及高频交易。

（1）量化选股

量化选股是采用数量化的方法判断某支股票是否值得买入的行为。量化选股的方法有很多种，其中基本面量化选股和技术面量化选股为最常见的两种选股策略。基本面量化选股的指导思想为“价格围绕价值上下波动”，主要依靠的是投资者在市场中获得的经验，通过对公司所在的行业情况、财务状况、经营情况等多方面的综合考量来得到结论，判断其股票的市场价格与理论价格之间的差距，买入被低估的股票，卖出被高估的股票，从而获得超额收益。技术面量化选股则更加依赖数学模型，构建基于市场化指标的模型来预测市场未来一段时间的走势，从而确定合适的投资策略，并做出相应的投资行为。

（2）量化择时

量化择时是指使用数量化的方法对股票市场未来一段时间内的走势做出判断，进而为投资者挑选出合适的买卖时机。股市的走势是否能够预测依赖于有效市场假说，如果该理论成立，则股票价格能够全面反映所有信息，即股票价格变化服从随机游走，那么对于价格的预测就是无意义的。然而我国的股市的收益中存在非线性的相关性，从而不满足随机游走假设，即股票价格波动不是完全随机，在其复杂的表面背后隐藏着某种确定性的规律，也就是说通过对历史数据信息的充分分析是可以预测其未来走势的。

随着近几年来研究的不断深入，学者们发现股票的价格变动受到非常多复杂因素的影响，传统线性模型的预测效果并不能够满足投资者的需求，因此陆续提出了一些具有更好预测效果的模型，如支持向量机、神经网络、灰色预测模型等。与传统模型相比，新模型具有更加快速的学习能力以及更广泛的适应能力，因而被广泛的应用于股票时间序列的预测当中，为投资者提供更多有效的量化投资策略。

（3）无风险套利

无风险套利是跨期套利的一种，套利方式为将资本投资于外汇中并规定远期汇率，取得外汇存款收益后按既定汇率换回本币，从而获得高于国内存款利率的

收益。无风险套利最吸引投资者的地方在于既满足了套利的需求，又满足了保值的需求，当然在实际市场中这种能够获得稳健收益的无风险套利机会并不是很常见。

（4）统计套利

统计套利不同于无风险套利，是一种风险套利。通过证券价格的历史信息中所蕴藏的规律来进行套利，而从历史信息中获得的规律是否在能够在未来稳定的持续下去是其面对的主要风险。统计套利可以分为两类，一类是β中性策略，该策略利用股票收益率进行建模分析，以组合的β值等于零为前提来实现α收益；第二类是协整策略，该策略利用股票价格的协整关系进行建模。

（5）高频交易

高频交易顾名思义，是指从股票买入与卖出之间极为微小的价格变化中寻求获利的交易策略，很多时候交易的时间单位是由毫秒来进行计算的，因此高频交易通常都会由专业的投资机构来完成。一般情况下，高频交易会在一天内完成大量的交易，这也需要良好的计算机运算速度以及容量作为技术支撑。由于交易时间间隔短，其每笔交易的收益一般较低，但其交易次数往往较多，因此总体上能够获得稳定的回报。

2.3 机器学习相关理论

2.3.1 XGBoost 算法原理

XGBoost 算法由陈天奇博士[15]在 2014 年提出，该算法一经提出就受到了广泛的关注，在近几年的 Kaggle、天池等各类大赛中均取得了令人瞩目的成绩。XGBoost 算法是在 Adaboost 算法和 GBDT 算法的基础上改进得到的提升算法，与单个决策树不同的是，XGBoost 是集成算法，其实现依赖于弱分类器的不断迭代。

假设一个训练集含有n条记录，m个解释变量，利用算法迭代每次生成一颗决策树，第i个样本的预测值$\hat{y}_i$可表示为式（2-1）：

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{2-1}$$

其中，将第k次迭代生成的决策树表示为f_k，F表示决策树的全集。

一般情况下，模型最终预测是否准确主要依赖于目标函数的优化情况。目标函数的优化主要通过两个方面来实现，一方面是模型得到的预测值与真实值越接

近越好，这可以通过最小化损失函数来实现；另一方面是模型的泛化能力越强越好，这可以通过在最小化损失函数的过程中加入惩罚项来实现，惩罚项的作用是控制模型的复杂程度，如L_1, L_2正则化项。通过优化目标函数来达到模型的误差和复杂度均为最优的目的。XGBoost 算法的目标函数为式（2-2）：

$$Obj(\theta) = L(\theta) + \Omega(\theta) \tag{2-2}$$

其中$L(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$表示模型训练误差，函数$l$为损失函数，可以进行不同选择，如平方损失函数；$y_i$表示目标值，$\hat{y}_i$表示模型对应预测值；$\Omega(\theta) = \sum_{k=1}^{K} \Omega(f_k)$则是添加的惩罚项，即正则项，目的是控制树的复杂程度，以达到提升算法鲁棒性的目的。$\Omega(f_k)$表示对第k棵树的惩罚，计算方法如式（2-3）：

$$\Omega(f_k) = \gamma T_k + \frac{1}{2}\lambda \sum_{j=1}^{T_k} \omega_{k,j}^2 \tag{2-3}$$

其中T_k表示第k棵树的叶节点数，γ是T_k的系数，$\omega_{k,j}$表示第k棵树第j个叶节点的得分值，λ和ω构成了惩罚项。

由于该模型是以加法方式进行训练，令$\hat{y}_i^{(t)}$表示第t次迭代时的第i个树的输出，则能够推出式（2-4）：

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{2-4}$$

进一步可将目标函数写为式（2-5）：

$$Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \sum_{k=1}^{t} \Omega(f_k) \tag{2-5}$$

XGBoost 算法采用二阶泰勒展开对目标函数进行近似，近似结果如式（2-6）:

$$Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right) + \sum_{k=1}^{t} \Omega(f_k) \tag{2-6}$$

其中g_i和h_i分别表示损失函数的一阶和二阶导数：

$$g_i = \frac{\partial l\left(y_i, \hat{y}_i^{(t-1)}\right)}{\partial \hat{y}_i^{(t-1)}}, h_i = \frac{\partial^2 l\left(y_i, \hat{y}_i^{(t-1)}\right)}{\partial^2 \hat{y}_i^{(t-1)}}$$

去掉常数项整理得到第t次迭代的目标函数式（2-7）：

$$Obj^{(t)}=\sum_{i=1}^{n}\left(g_i f_t(x_i)+\frac{1}{2}h_i f_t^2(x_i)\right)+\Omega(f_t) \tag{2-7}$$

每棵树$f(x)$有独立的树结构q和叶节点得分ω：

$$f(x)=\omega_{q(x)},\omega\in R^T,q:R^d\to\{1,2,\cdots,T\} \tag{2-8}$$

设第j个叶节点的样本集为$I_j=\{i|q(x_i)=j\}$，可将目标函数写成式（2-9）：

$$\begin{aligned}Obj^{(t)}&=\sum_{i=1}^{n}\left(g_i f_t(x_i)+\frac{1}{2}h_i f_i^2(x_i)\right)+\Omega(f_t)\\&=\sum_{i=1}^{n}\left(g_i\omega_{t,q(x_i)}+\frac{1}{2}h_i\omega_{t,q(x_i)}^2\right)+\gamma T_t+\frac{1}{2}\lambda\sum_{j=1}^{T_t}\omega_{t,j}\\&=\sum_{j=1}^{T_t}\left[G_j\omega_{t,j}+\frac{1}{2}(H_j+\lambda)\omega_{t,j}\right]+\gamma T_t\end{aligned} \tag{2-9}$$

其中$G_j=\sum_{i\in I_j}g_i$，$H_j=\sum_{i\in I_j}h_i$分别为第j叶节点下与其对应的各样本的一阶、二阶导数和。

当树结构$q(x)$固定时，对式（2-9）求导可得相应最小值的解$\omega_{t,j}^*$：

$$\omega_{t,j}^*=-\frac{G_j}{H_j+\lambda} \tag{2-10}$$

将叶节点的最小值带入目标函数，最终可优化为式（2-11）：

$$Obj^{(t)}=-\frac{1}{2}\sum_{j=1}^{T_t}\frac{G_j{}^2}{H_j+\lambda}+\lambda T_t \tag{2-11}$$

类似于决策树中使用基尼不纯度，目标函数（2-11）可作为得分指标，进行节点分裂特征的选择以及对树质量的评价，该得分函数值越小代表着树的结构越好。假设分裂后左边和右边节点的样本集分别为I_L,I_R,且$I=I_L\cup I_R$，对某个节点进行分裂处理后的新损失函数表示为式（2-12）：

$$L_{split}=\frac{1}{2}\left[\frac{G_L{}^2}{H_L+\lambda}+\frac{G_R{}^2}{H_R+\lambda}-\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right]-\gamma \tag{2-12}$$

其中，G_L表示左子树一阶导数和，H_L表示左子树二阶导数和，G_R表示右子树一阶导数和，H_R表示右子树二阶导数和，$\frac{G_L{}^2}{H_L+\lambda}$表示左叶节点的值，$\frac{G_R{}^2}{H_R+\lambda}$表

示右叶节点的值，$\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}$表示为分裂前的值，$\gamma$是引入了多一个的叶节点增加的复杂度。

XGBoost 算法采用贪婪算法或近似贪婪算法来选择特征属性进行节点分裂，遍历所有特征的划分点，分别计算对应分裂后的目标函数的信息增益值，选择最优的特征进行分裂。当前树停止生长需满足达到设定的最大深度或由分裂带来的信息增益小于设定的阈值。

图 2-1 为 XGBoost 模型结构示意图：

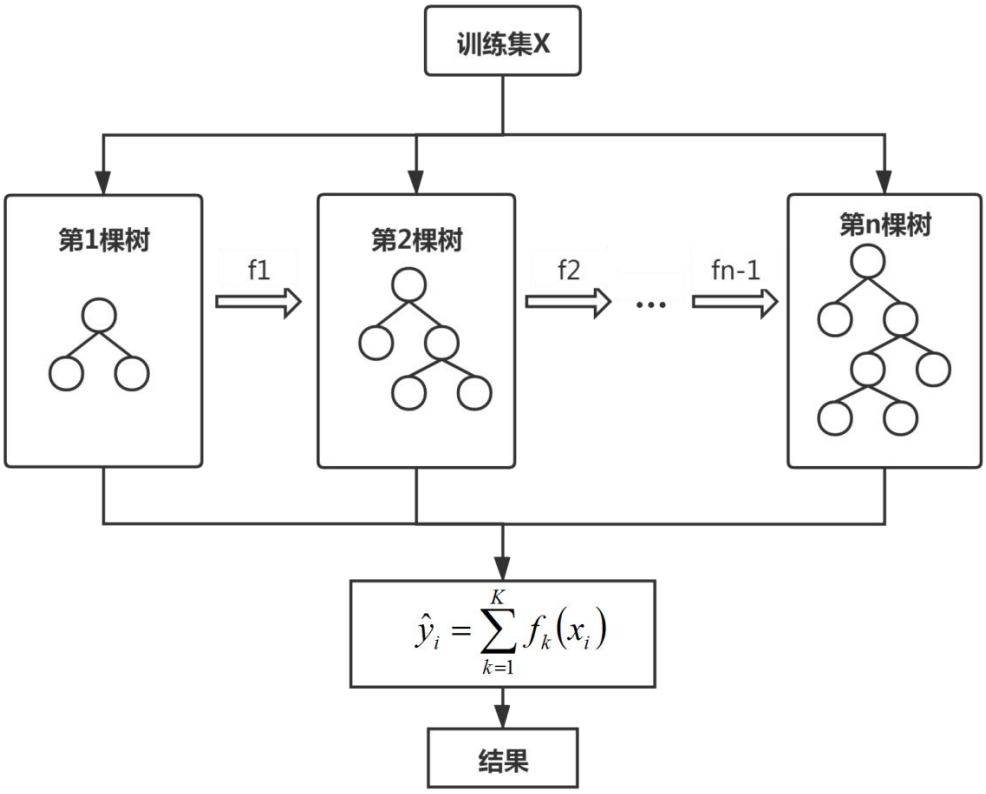

图 2-1　XGBoost 算法结构示意图

2.3.2 模型参数优化方法

模型参数优化是机器学习过程中重要的一步，其目的是提高模型在未知的样本数据上预测的准确率，通常使用泛化能力来进行衡量。模型的复杂程度是影响模型泛化能力主要因素，当模型过于复杂时，对于袋内的数据呈现较好的拟合效果，但对于袋外的数据往往表现较差，出现过拟合现象；但是当模型过于简单时，又无法较好的拟合数据，出现欠拟合现象。无论是过拟合现象还是欠拟合现象，模型的预测效果都不理想，其泛化能力都比较差，只有当模型的复杂度达到最优

值时才能够满足泛化误差处于最小值的目标。图 2-2 展示了模型的泛化能力与模型复杂度之间的关系。

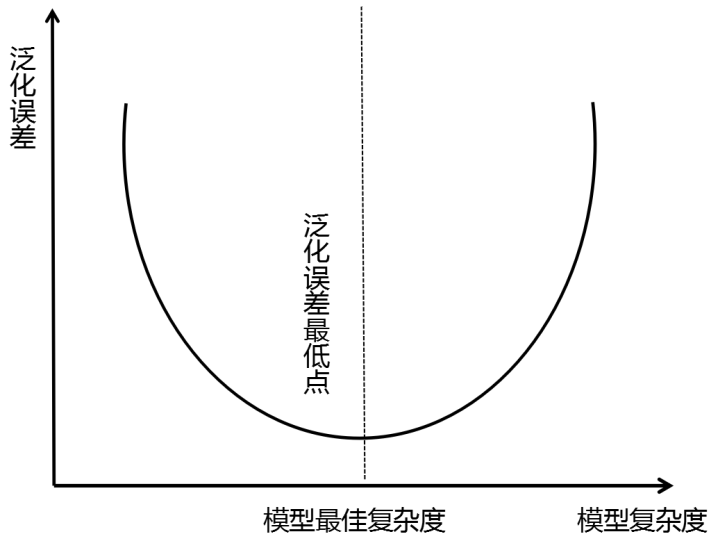

图 2-2 模型泛化能力与模型复杂度关系图

研究模型各个超参数对模型复杂度的影响是进行参数优化之前的重要步骤，通常可以采用网格搜索、AUC 面积等方式来进行参数优化。此外，交叉验证在参数调整的过程中也经常被使用到。

在建模过程中，通常将原始的数据集分为训练集、验证集和测试集。训练集的作用是为模型提供训练样本，验证集的作用是为模型参数调整提供验证样本，测试集的作用是为模型提供测试样本，用于判断模型的好坏。当样本较少时，上述划分数据集的方式不太适用，往往会采用交叉验证的方法来进行数据集的划分，k 折交叉验证指的是将数据集划分为 k 份。将数据集划分成 k 份，依次取一份做测试集，剩余 $k-1$ 份为训练集，多次进行训练模型来观察模型稳定性。

下面对十折交叉验证的过程为例来进行介绍。第一步将原始数据集划分为训练集 X 和测试集 y，采用随机抽样方法将训练集 X 划分成十个互斥子集。第二步依次取一个子集作为验证集，剩下九个子集作为训练集，进行十次模型训练并返回十次验证集的平均误差。图 2-3 展示了十折交叉验证的具体实现过程。

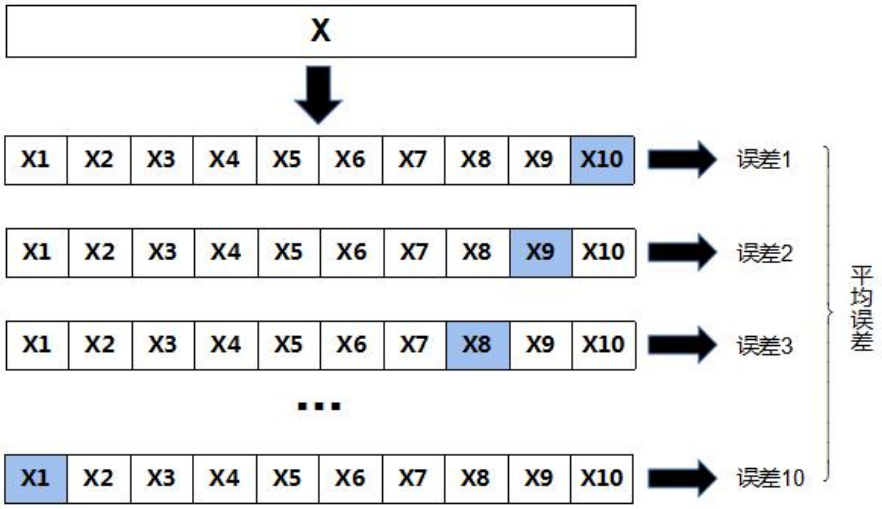

图 2-3 十折交叉验证过程示意图

2.3.3 模型评价方法

（1）分类模型评估指标

混淆矩阵可以理解为分别统计分类模型归错类、归对类的个数，把结果放在一个表里展示出来，如表 2-2 所定义:

表 2-2 混淆矩阵

混淆矩阵		真实值	
		Positive	Negative
预测值	Positive	TP	FN Type I
	Negative	FP Type II	TN

其中，FN（False Negative）表示真实值是 positive 且被模型认为是 negative 的数量，即第二类错误（Type II Error）；FP（False Positive）表示真实值是 negative 且被模型认为是 positive 的数量，即第一类错误（Type I Error）；TN（True Negative）表示真实值是 negative 且被模型认为是 negative 的数量；TP（True Positive）表示真实值是 positive 且被模型认为是 positive 的数量。

衡量分类模型好坏的指标非常多，常用的有准确率、精确度、召回率、F1 等，下面对这几个指标进行简单的说明。

（i）准确率（Accuracy）：是用来衡量模型的整体效果，其计算方式如式（2-14）所示:

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{2-14}$$

（ii）精确率（Precision）：是指预测正确的正例数占预测为正例总量的比例，一般情况下精确率越高，说明模型的效果越好，计算公式如式（2-15）所示：

$$P=\frac{TP}{TP+FP} \tag{2-15}$$

（iii）召回率（Recall）：预测正确的正例数占所有正例的比例，一般情况下召回率越高，说明有更多的正类样本被模型预测正确，即所选的模型的预测效果越好。计算公式如式（2-16）所示：

$$R=\frac{TP}{TP+FN} \tag{2-16}$$

（iv）F_1是一个由精确率和召回率共同构成的评价指标。计算公式如式（2-17）所示：

$$\frac{2}{F_1}=\frac{1}{P}+\frac{1}{R}$$
$$F_1=\frac{2TP}{2TP+FP+FN} \tag{2-17}$$

（2）回归模型评估指标

针对回归问题的评价指标也是很丰富的，常用的有误差平方和、均方误差、均方根、平均绝对误差等，下面对这几个指标进行简单的说明，令y_i是第i个样本的真实值，$\hat{y}_i$是第i个样本的预测值，n是样本的个数。

（i）误差平方和（SSE）：指的是拟合数据和原始数据对应点的误差的平方和，计算公式如式（2-18）所示：

$$SSE=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 \tag{2-18}$$

（ii）均方误差（MSE）：指的是预测数据和原始数据对应点误差的平方和的均值，计算公式如式（2-19）所示：

$$MSE=\frac{SSE}{n}=\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2 \tag{2-19}$$

（iii）均方根误差（RMSE）：其又被称为 RMSD（root mean square deviation），计算公式如式（2-20）所示：

$$RMSE=\sqrt{MSE}=\sqrt{\frac{SSE}{n}}=\sqrt{\frac{\sum_{i=0}^{n}\left(y_i-\hat{y}_i\right)^2}{n}} \tag{2-20}$$

RMSE 使用的是欧式距离。该指标虽然应用广泛，但是由于使用的是平均误差，导致其对异常点较敏感。

（iv）平均绝对误差（MAE）：指的是预测值与真实值之间平均相差的大小，计算公式如式（2-21）所示：

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2-21}$$

第三章 高频因子的构建与研究

通过阅读文献和查找相关资料可以发现，目前针对传统因子的研究已经十分的深入，从基本面因子、技术面因子、宏观经济因子到分析师预测因子、市场情绪因子等，各类因子几乎都已经被涉及到，同时在对传统因子的研究当中，学者们一般选择较低频的数据来进行因子的处理与构造，如基本面因子主要刻画了一个公司的财务状况，这就需要上市公司三大财务报表的数据，因此大都以季度数据来进行构造，而宏观经济因子则有一些需要年度数据来刻画，如 GDP 增速、失业率以及通货膨胀率等。因此，仅通过使用日频数据已经很难再发掘出因子中多余的选股能力了。本章将引入对日内高频数据的研究，通过构建高频因子来刻画股票日内特征，并探究其选股能力，为后续的多因子模型带来更多新信息。

本章选取了沪深 300 指数成分股 1 分钟频率行情数据来构建相关因子，具体沪深 300 指数成分股名单以 2021 年最新版为准，其中获取的行情数据指标包括开盘价（open）、收盘价（close）、最高价（high）、最低价（low）、成交量（volume）以及成交额（money），主要构建了收益率分布因子、成交量分布因子以及量价复合因子三大类，各类因子还可以进一步细化成为图 3-1 中的高频因子：

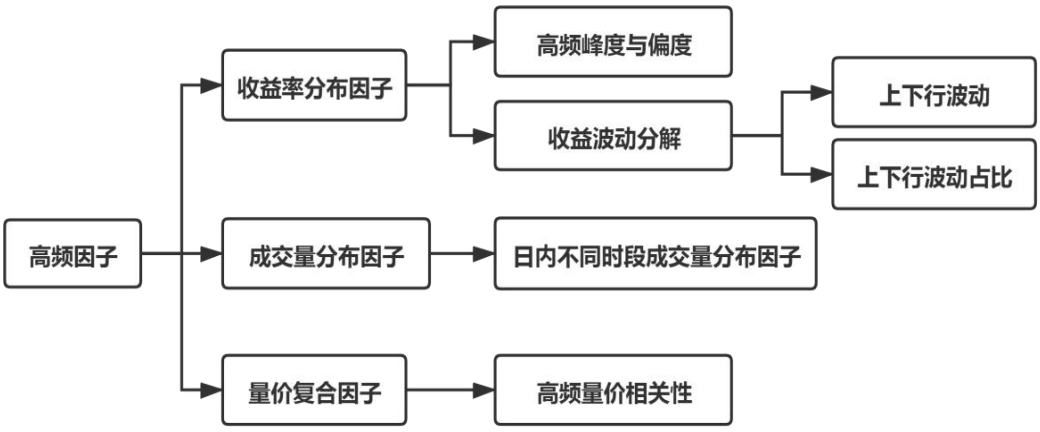

图 3-1 高频因子梳理

本章将从两个方面进行展开，第一个方面主要对高频因子的构建原理以及计算方式进行说明；第二个方面将对每个高频因子进行单因子分析，探究其是否具有选股能力，选取 IC 值（Information Coefficient，信息系数）作为评价指标，挑选出具有良好选股能力的高频因子，作为一类新因子加入到后续的多因子选股模

型当中。

3.1 高频因子构建

本文中构建高频因子的思路主要借鉴了海通证券研究所在过去三年中陆续发布的《选股因子系列研究》[16-18]以及《金融工程专题报告》[19-20]两个系列的研究报告。

3.1.1 收益率分布因子

（1）高频收益偏度与峰度

通过偏度值与峰度值可以来刻画股票收益率日内的分布情况，在任意交易日，可以计算出股票高频收益序列的方差、偏度以及峰度三个指标。常见的方差、偏度、峰度计算方法有如式（3-1）和（3-2）所示：

方式一：

$$\begin{aligned} &\text{高频收益方差：} RVar_i = \sum_{j=1}^{N} r_{ij}^2 \\ &\text{高频收益偏度：} RSkew_i = \frac{\sqrt{N}\sum_{j=1}^{N} r_{ij}^2}{RVar_i^{3/2}} \\ &\text{高频收益峰度：} RKurt_i = \frac{N\sum_{j=1}^{N} r_{ij}^4}{RVar_i^2} \end{aligned} \tag{3-1}$$

方式二：

$$\begin{aligned} &\text{高频收益方差：} RVar_i = \sum_{j=1}^{N} \left(r_{ij} - \bar{r}_i\right)^2 \\ &\text{高频收益偏度：} RSkew_i = \frac{\sqrt{N}\sum_{j=1}^{N} \left(r_{ij} - \bar{r}_i\right)^2}{RVar_i^{3/2}} \\ &\text{高频收益峰度：} RKurt_i = \frac{N\sum_{j=1}^{N} \left(r_{ij} - \bar{r}_i\right)^4}{RVar_i^2} \end{aligned} \tag{3-2}$$

其中 r_{ij} 为股票 i 的日内 1 分钟对数收益序列，$\bar{r}_i$ 为股票 i 的日内 1 分钟对数收益均值，同时，上述因子值为前 N 日指标的均值。

在实际操作中，量化选股往往是以月度为单位来进行的，因此本文中对高频因子进行计算时也选取的是过去一个月的均值。本文采用方式一来进行因子值的计算。

（2）收益波动分解

自 1952 年马科维兹（Harry M. Markowitz）首次提出使用收益率的标准差来度量资产的波动情况之后，收益波动率被广泛的应用，成为风险度量最常用的方法之一。在传统的资产定价理论中，资产波动率一直是人们关注的重点，当波动率的值越大时，意味着预期收益的不确定性越高，同时也会有更高的公允价值来作为不确定性的风险补偿。本文使用日内高频数据来刻画收益率波动情况，具体构建方法如下。

（i）高频上下行波动

从股票收益率上、下行波动的角度对股票高频波动进行分解，对于股票 i 在某段时间的高频收益序列，其上、下行波动的计算方式如式（3-3）所示：

$$\begin{aligned} \text{高频上行波动} &= \left(\sum_t \left(r_i^t I_{\{r_i^t>0\}} \right)^2 \right)^{\frac{1}{2}} \\ \text{高频下行波动} &= \left(\sum_t \left(r_i^t I_{\{r_i^t<0\}} \right)^2 \right)^{\frac{1}{2}} \end{aligned} \tag{3-3}$$

（ii）高频上下行波动占比

与高频上、下行波动类似，构建股票 i 上、下行波动占比，计算方式如式（3-4）所示：

$$\begin{aligned} \text{高频上行波动占比} &= \frac{\left(\sum_t \left(r_i^t I_{\{r_i^t>0\}} \right)^2 \right)^{\frac{1}{2}}}{\left(\sum_t \left(r_i^t \right)^2 \right)^{\frac{1}{2}}} \\ \text{高频下行波动占比} &== \frac{\left(\sum_t \left(r_i^t I_{\{r_i^t<0\}} \right)^2 \right)^{\frac{1}{2}}}{\left(\sum_t \left(r_i^t \right)^2 \right)^{\frac{1}{2}}} \end{aligned} \tag{3-4}$$

其中 r_i^t 表示 t 时刻股票 i 的收益率，I 表示示性函数，使用股票过去一个月的指标均值作为因子值。

3.1.2 成交量分布因子

通过观察股票日内成交量走势情况可以发现，一般情况下成交量呈现“U”型或者“W”型走势。“U”型走势表示的是在开盘和收盘时段的成交量更高；“W”型走势表示的是除开盘和收盘时段外，午间的休市也导致了下午开盘时成交量更高。因此，不同时间段的成交量分布情况中蕴含着投资者的行为特征，同时在技术指标中，也有专门针对成交量构建的一类因子。本文也将从成交量的角

度出发，构建日内成交量分布因子，用于对日内成交量分布的刻画，具体构建方法如下。

成交量分布因子的构建以半小时为时间间隔，将一天划分成八个时间区间，分别计算每个区间的成交量占比，即共有八个不同的成交量分布因子，使用股票过去一个月指标均值作为因子值，具体公式如式（3-5）所示：

$$VolumeRatio_t = \frac{Volume_t}{Volume_{total}} \tag{3-5}$$

其中t表示具体的某个时间区间，如 9:30-10:00。

3.1.3 量价复合因子

在量价关系中，量指的是股票在单位时间内的成交量；价指的是股票的价格，一般情况下指的是收盘价。投资者往往可以通过对股票价格的涨跌与其成交量大小之间存在的内在联系进行分析，来判断目前的形势，并进行交易。

本文通过高频量价相关性来刻画量价关系，即每一交易日取股票日内 1 分钟频率的价格P_t和成交量V_t序列，计算其 Pearson 相关系数，使用股票过去一个月的指标均值作为因子值，具体公式如式（3-6）所示：

$$\rho = corr(P_t, V_t) \tag{3-6}$$

3.2 单因子有效性检验

本部分将对上述构建的 16 个高频因子进行单因子选股效果的分析，在量化投资中，通常会选取 IC 值和 IR 值[21]作为衡量因子好坏的一个标准，在多因子模型中，一般认为 IC 的绝对值大于 0.03 时该因子具有一定的选股能力，即和收益率的相关性较大。因此本文采用这两个指标来判断高频因子的有效性。

3.2.1 IC 和 IR 相关理论[22]

（1）IC 值

IC 值为信息系数（Information Coefficient），表示所选股票因子值与股票下期收益的截面相关系数，往往可以采用 IC 值的大小来判断单因子对股票未来收益的预测能力。信息系数的取值在-1 到 1 之间，绝对值越大，表明该单因子预测效果越好，即有越好的选股能力。若 IC 值为负，则表示该因子值越小越好，若 IC 值为正，则表示该因子值越大越好。

（2）IR 值

IR 值为信息比率（Information Ratio），是超额收益的均值与标准差之比，可以根据 IC 值近似计算，具体公式如式（3-7）所示：

$$IR \approx \frac{\overline{IC_t}}{std(IC_t)} \tag{3-7}$$

其中 $\overline{IC_t}$ 表示超额收益均值，$std(IC_t)$ 表示超额收益标准差。

IR 值的计算是从超额收益出发，逐步推导得到的。IR 值等于 IC 值的多周期均值除以 IC 值的标准方差，代表因子获取稳定 α 的能力。因此，IR 值既能体现由 IC 值代表因子的选股能力，又能体现由 IC 值标准差的倒数代表因子选股能力的稳定性。一般认为，当 IR 值大于 0.5 时因子稳定获取超额收益能力较强。

（3）Rank IC 值

Rank IC 值是因子的排序值与收益的排序值之间的相关系数，Rank IC 值和 IC 值唯一的不同点就是在求相关系数时，换成秩相关系数。目前，更多的学者在研究的过程中会选取 Rank IC 值来代替普通的 IC 值，这主要是因为普通 IC 值的计算需要数据满足服从正态分布的条件，然而股票数据往往并不能够满足这个要求，因此更多的采用秩相关系数也就是 Rank IC 来判断单因子的有效性。

3.2.2 高频因子有效性检验

（1）结果汇总

在对 IC 值、IR 值相关知识建立了充分理解的基础上，选取沪深 300 指数成分股 2011 年 1 月 1 日至 2020 年 12 月 31 日的 1 分钟频率交易数据，交易数据具体包括开盘价（open）、收盘价（close）、最高价（high）、最低价（low）、成交量（volume）以及成交额（money）。按照 3.1 中的构建方式来计算 16 个高频因子的具体值，并分别进行单因子有效性检验。本文在计算 IC 值过程中均选取的是 Spearman 相关系数，表 3-1 展示了 16 个高频因子 IC、Rank IC 以及 IR 值的汇总情况。

从表 3-1 中可以看出 IC 值与 Rank IC 值的差距并不是很大，说明选取的数据正态性较好。从 IC 值的角度来看，1 分钟频率下的高频方差因子、上下行波动、10:00-10:30 成交量占比、11:00-11:30 成交量占比、13:30-14:00 成交量占比、14:00-14:30 成交量占比以及高频量价相关性不具有选股能力；1 分钟频率下的高频偏度因子、上下行波动占比、10:30-11:00 成交量占比以及 13:00-13:30 成交量占比具有弱选股能力；1 分钟频率下的高频峰度因子、9:30-10:00 成交量占比、14:30-15:00 成交量占比具有较好的选股能力。

表 3-1 高频因子检验结果汇总

因子名称	IC	Rank IC	IR
高频收益方差	-0.000381776	-0.000381776	-0.005088034
高频收益偏度	**0.005711718**	**0.005711718**	**0.178893602**
高频收益峰度	**-0.013839758**	**-0.013839758**	**-0.279941322**
上行波动	-0.001123326	-0.001121578	-0.014691641
下行波动	-0.001404651	-0.001402846	-0.018377969
上行波动占比	**0.005007846**	**0.0050106**	**0.067358847**
下行波动占比	**-0.006014761**	**-0.006018349**	**-0.182042402**
9:30-10:00 成交量占比	**0.024126895**	**0.023737333**	**0.4242572**
10:00-10:30 成交量占比	0.003777314	0.003052515	0.062930986
10:30-11:00 成交量占比	**-0.005835916**	**-0.005903501**	**-0.181750233**
11:00-11:30 成交量占比	0.003011439	0.003266169	0.042498052
13:00-13:30 成交量占比	**0.005027444**	**0.0051405**	**0.094139999**
13:30-14:00 成交量占比	-0.000315022	-0.000307489	-0.004318232
14:00-14:30 成交量占比	-0.003110437	-0.002916477	-0.044719794
14:30-15:00 成交量占比	**0.032225968**	**0.033288571**	**0.549459068**
高频量价相关性	-0.001035464	-0.001035255	-0.012981669

（2）有效因子分析

（i）高频收益偏度和峰度

图 3-2 和图 3-3 分别展示了高频偏度因子和高频峰度因子从 2011 年 1 月至 2020 年 12 月共 10 年的 Rank IC 值，因子值为过去一个月的指标均值，对应给出以月度为单位的 120 个月的 Rank IC 值。

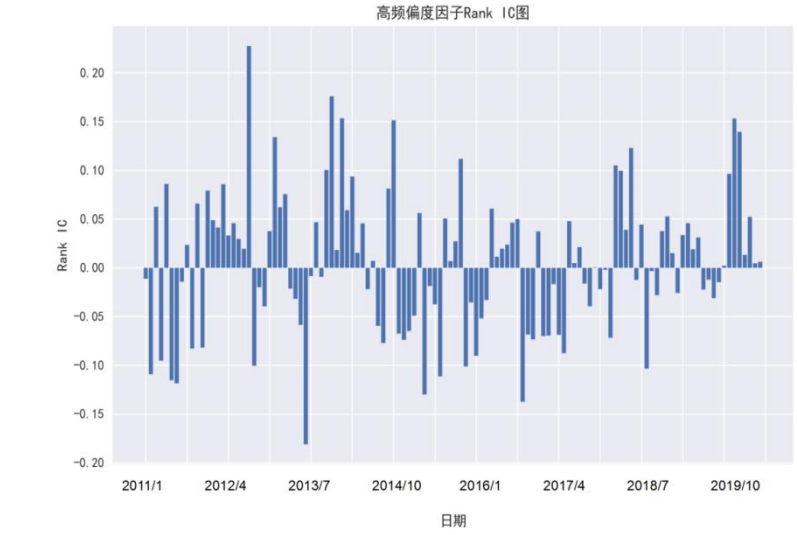

图 3-2 高频偏度因子 Rank IC 值

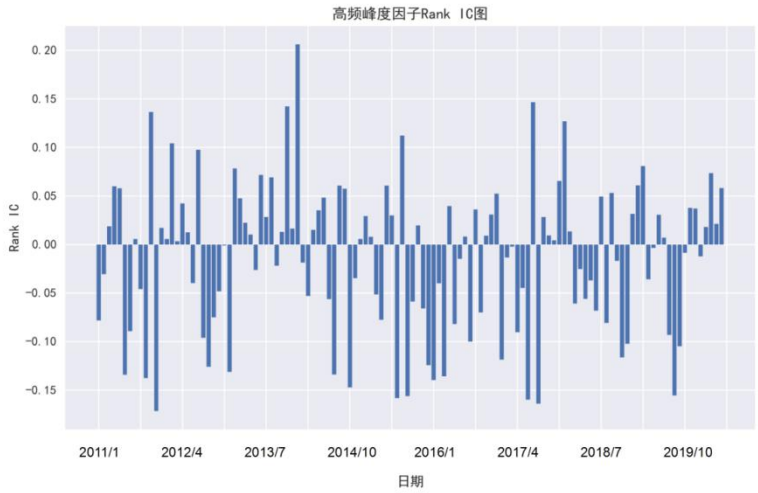

图 3-3 高频峰度因子 Rank IC 值

图 3-2 和图 3-3 分别展示了从 2010 年以来高频偏度因子和高频峰度因子的月度选股效果，可以看出在大部分的月份里，高频偏度因子具有一定的选股能力，Rank IC 最大值超过了 0.2；高频峰度因子的表现更佳一点，具有更好的选股效果，整体上与下期的收益呈现负相关的关系，该项因子较大的股票，在下期出现下跌的可能性较大。

（ii）高频上下行波动

图 3-4 和图 3-5 分别展示了上行波动占比因子和下行波动占比因子从 2011 年 1 月至 2020 年 12 月共 10 年的 Rank IC 值，因子值为过去一个月的指标均值，对应给出以月度为单位的 120 个月的 Rank IC 值。

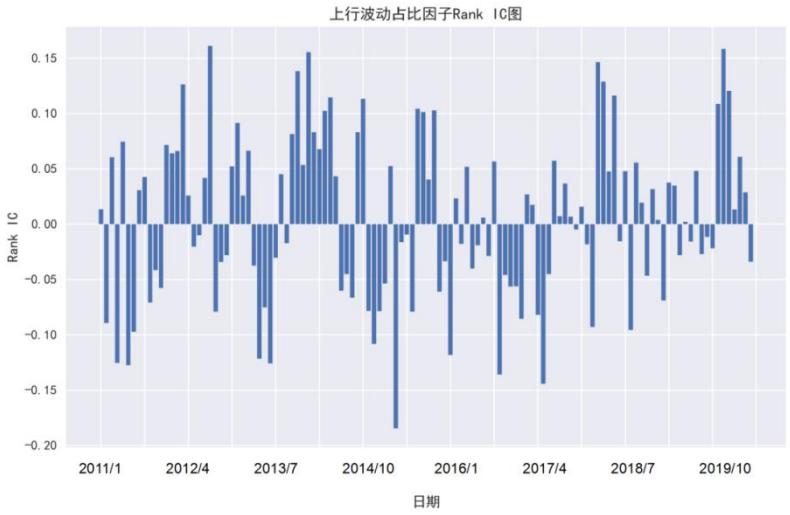

图 3-4 上行波动占比因子 Rank IC 值

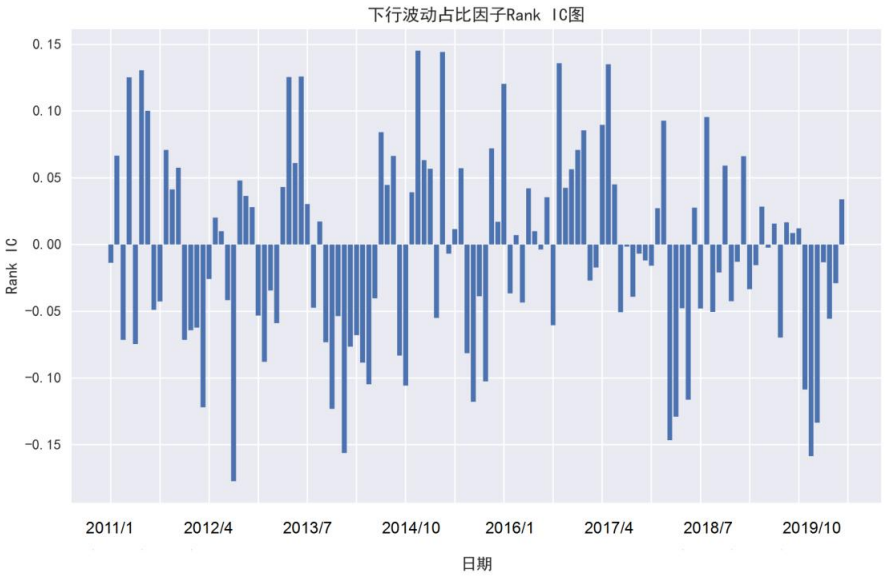

图 3-5 下行波动占比 Rank IC 值

图 3-4 和图 3-5 分别展示了从 2011 年以来上行波动占比因子和下行波动占比因子的月度选股效果，可以看出两者都具有一定的选股效果，整体上上行波动占比与下期收益呈现正相关关系，即前期股票高频上行波动占比越大，未来 1 个月收益表现越好；而下行波动占比与下期收益呈现负相关关系，即前期股票高频下行波动占比越大，未来一个月收益表现越差。股票高频收益的上行波动衡量了股票价格拉升的特征，简单来说，股票 *A* 和股票 *B* 在过去的一段时间里呈现出相同的涨幅，但是两支股票涨幅构成情况不同，股票 *A* 的涨幅是由持续稳定的小涨幅构成，而股票 *B* 的涨幅是由短期的大幅拉升构成的，那么可以得到股票 *B* 更

有可能出现收益反转的情况，同时其上行波动率占比的因子值也会相对较高。

（iii）日成交量占比

图 3-6、图 3-7、图 3-8 和图 3-9 分别展示了 9:30-10:00 成交量占比、10:30-11:00 成交量占比、13:00-13:30 成交量占比、14:30-15:00 成交量占比因子从 2011 年 1 月至 2020 年 12 月共 10 年的 Rank IC 值，因子值为过去一个月的指标均值，对应给出以月度为单位的 120 个月的 Rank IC 值。

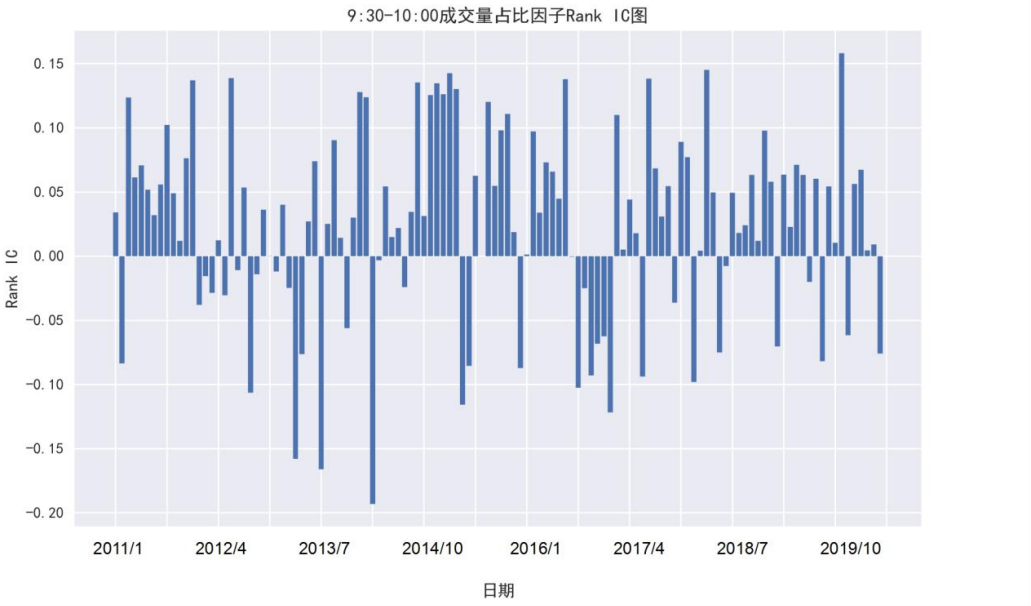

图 3-6 9:30-10:00 成交量占比因子 Rank IC 值

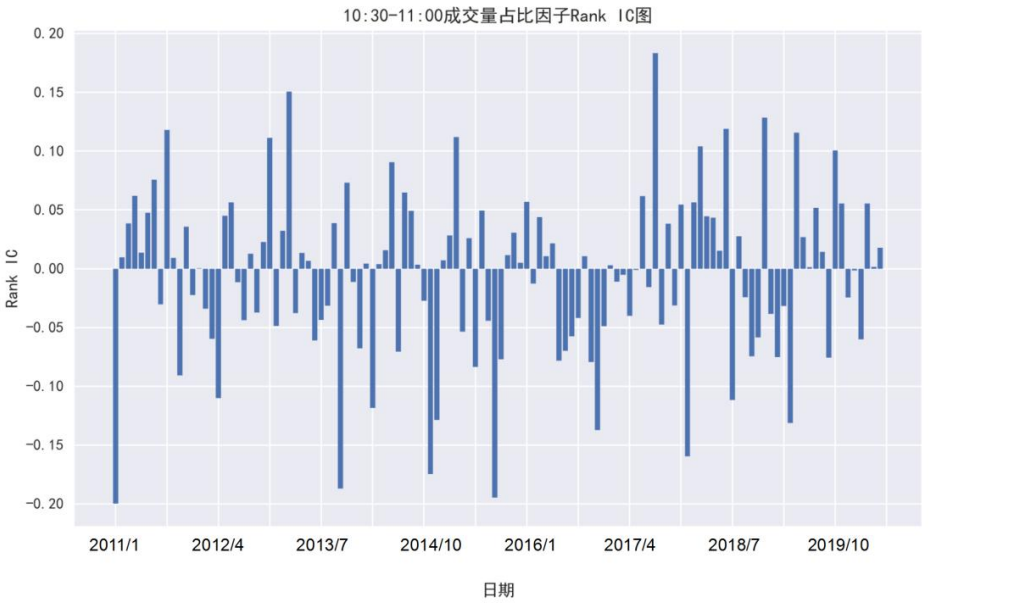

图 3-7 10:30-11:00 成交量占比因子 Rank IC 值

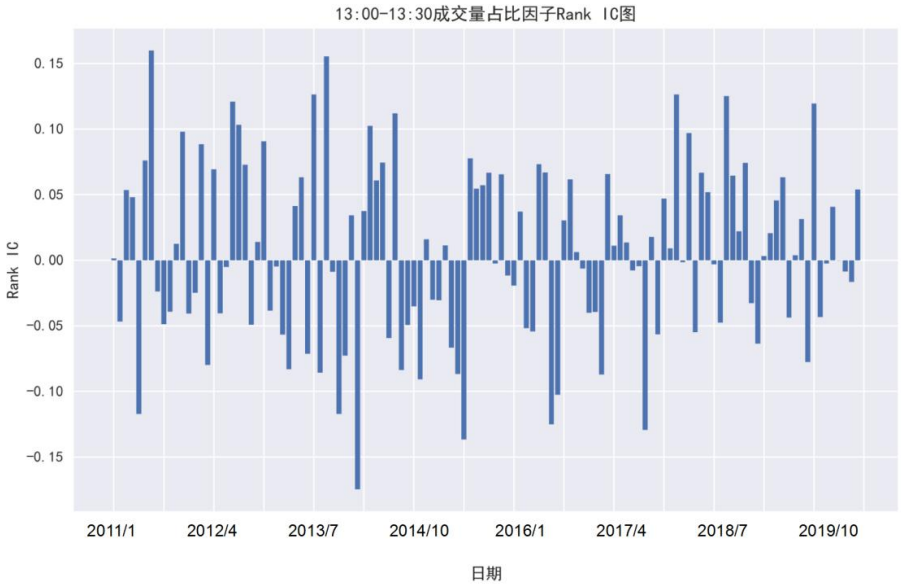

图 3-8 13:00-13:30 成交量占比因子 Rank IC 值

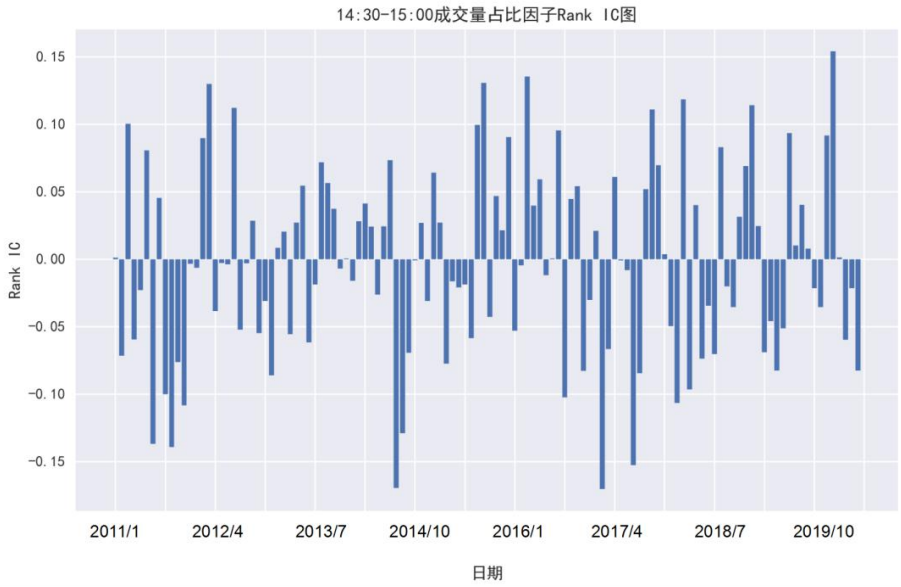

图 3-9 14:30-15:00 成交量占比因子 Rank IC 值

图 3-6、图 3-7、图 3-8 和图 3-9 分别展示了从 2010 年以来 9:30-10:00 成交量占比、10:30-11:00 成交量占比、13:00-13:30 成交量占比、14:30-15:00 成交量占比因子的月度选股效果，可以看出其中 9:30-10:00 成交量占比 13:00-13:30 成交量占比以及 14:30-15:00 成交量占比因子和股票下月收益呈现正相关关系，而 10:30-11:00 成交量占比与下月收益呈现负相关关系。

通过观察股票日内成交量走势情况可以发现，一般情况下成交量呈现“U”型或者“W”型走势。“U”型走势表示的是在开盘和收盘时段的成交量更高；

“W”型走势表示的是除开盘和收盘时段外，午间的休市也导致了下午开盘时成交量更高。即投资者的行为特征从不同时间段的成交量分布情况中也可以有所体现，成交量占比因子中蕴含了额外信息。不同时间段成交量存在一定差异的原因可能是非常复杂的，投资者在对丰富的隔夜信息进行分析之后可能导致开盘时段成交量较高，同样的午间休市也使得投资者有更多获取信息的时间，而收盘时段则体现了投资者对于下一个交易日的预期，当投资者获取到足够的信息时更容易做出正确的判断，从而在下期获得更高的收益。

3.3 本章小结

本章主要集中于对于日内高频数据的研究，首先介绍了高频因子的具体的构造方式以及原理，共构造三大类、16 个高频因子，接下来分别对这些因子进行了有效性的检验，并最终选取出 1 分钟频率下的高频偏度因子、高频峰度因子、上下行波动占比、9:30-10:00 成交量占比、10:30-11:00 成交量占比以及 13:00-13:30 成交量占比以及 14:30-15:00 成交量占比因子共 8 个因子加入后续的多因子模型研究当中，希望据此能够从日内的高频数据中获取更多的新信息。

第四章 基于 XGBoost 选股模型的实证研究

目前对于传统的多因子量化选股模型的研究主要集中于两个方面，一方面集中于对输入指标即因子的选择，另一方面则是集中于构建模型的算法的选择。本章同样也将重点放在这两方面来进实证分析。首先，将第三章中选取出来的具有良好选股效果的高频因子与传统的多因子相结合，构建新的多因子选股模型，选取行情指标、技术指标以及高频因子作为输入特征，股票收益率为输出特征；其次，总结发现机器学习算法在量化投资中有良好表现，往往能够带来稳定的超额收益，因此本文将机器学习方法与量化选股相结合，选取 XGBoost 算法运用于股票未来收益率的预测当中，同时与传统的多因子模型的预测效果进行对比，进一步体现将高频因子加入多因子模型当中的必要性，将 XGBoost 算法与传统线性回归和逻辑回归的预测效果进行对比，探究机器学习方法应用于量化选股中的必要性。本章的结构如图 4-1 所示：

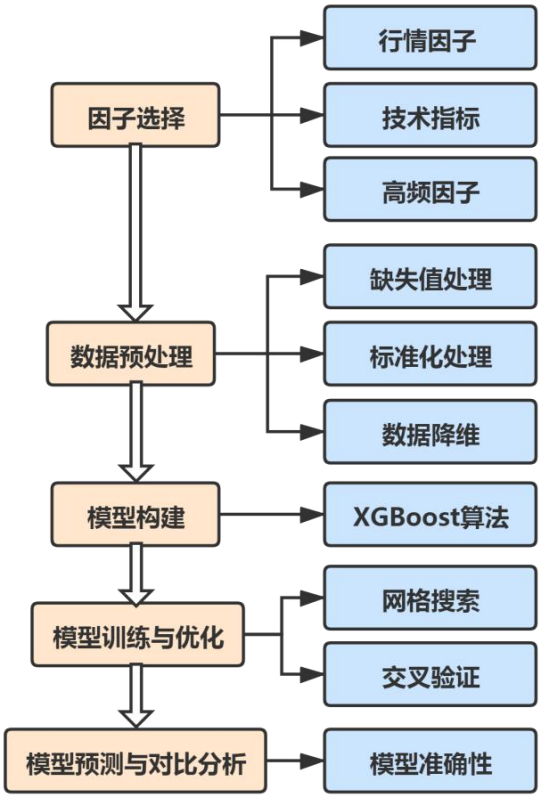

图 4-1 第四章结构图

4.1 数据获取与指标选取

4.1.1 数据获取

本章的所有股票行情数据、技术指标均是通过聚宽平台 Python API 进行调用，主要获取了从 2011 年 1 月 1 日至 2020 年 12 月 31 日共 10 年的沪深 300 指数成分股日频行情数据，行情指标包括开盘价（open）、收盘价（close）、最高价（high）、最低价（low）、成交量（volume）、成交额（money）、涨跌幅（pct_chg）以及换手率（turn），同时剔除了上市时间较短的 60 支股票，最终保留 240 支股票来进行后续模型的训练与预测，共计 583681 条数据，部分样本数据如表 4-1 所示：

表 4-1 部分样本数据

ID	time	open	high	low	close	volume	money	pct_chg	turn
000001.SZ	2011/1/4	5.1151	5.2315	5.0601	5.1798	35884061	5.73E+08	1.4566	1.1556
000001.SZ	2011/1/5	5.1701	5.2153	5.1442	5.1507	23746263	3.8E+08	-0.5618	0.7647
000001.SZ	2011/1/6	5.1507	5.1992	5.0763	5.1119	20653127	3.27E+08	-0.7533	0.6651
002120.SZ	2015/8/4	6.2623	6.7412	6.1886	6.7412	5635146	99304548	9.976	5.7662
002120.SZ	2015/8/5	6.7559	7.0801	6.6601	6.7043	6767598	1.26E+08	-0.5464	6.925
002120.SZ	2015/8/6	6.4501	6.9917	6.4317	6.789	5669669	1.04E+08	1.2637	5.8015
000063.SZ	2020/12/29	30.65	33.63	30.65	33.63	1.61E+08	5.3E+09	10.0098	4.6173
000063.SZ	2020/12/30	33.59	33.59	32.6	33.44	1.14E+08	3.78E+09	-0.565	3.2807
000063.SZ	2020/12/31	33.43	34.06	32.82	33.65	1.05E+08	3.51E+09	0.628	3.0238

4.1.2 指标选择

通过研读的文献[23-25]可以发现，部分学者把开盘价、收盘价、最低价、最高价、成交量、成交价格等行情指标作为输入变量，同时随着近几年来对于多因子模型研究的不断深入，大量优秀的量化投资模型相继出现，一些研究结果显示技术指标[11-13]通常具有良好的预测效果，主要是通过对原始行情数据的构造形成新的特征应用于股票价格预测当中，本文也同样将技术指标作为输入特征，将行情指标、技术指标与高频因子相结合，应用于后续的股票收益率预测模型当中。

本章选取行情因子、技术指标以及高频因子共计 27 个，其中行情因子 8 个、技术指标 11 个、高频因子 8 个。对于高频因子的选取在前文已经进行了详细说明，这部分不再赘叙，下面主要对选取的部分技术指标含义以及计算方式进行说明，其他指标在表 4-2、4-3 和 4-4 中给出。

（1）超买超卖型

（i）RSI

相对强弱指标（Relative Strength Index），该指标由著名的技术派分析家威尔斯.威尔德（Welles Wilder）在 1978 年提出，旨在通过对一段时间内的平均收盘价上涨数情况以及平均收盘价下跌数情况来进行比较，进而分析整个市场对该股票的意向和实力对比，从而对未来市场的走势做出判断，可以理解为通过数学方法来计算出买卖双方的力量对比，如一件商品有 x 人同时面对，超过半数的人想要获得该商品，则会使得商品价格上涨，若超过半数的人想要卖出该商品，则使得商品价格下跌。该指标的取值范围为（0，100），当 RSI 大于 70 时为超买区域，小于 30 时为超买区域，具体计算公式如（4-1）所示：

$$RSI(n)=\frac{100\times RS}{1+RS} \tag{4-1}$$

式（4-1）中 n 表示计算的具体周期，RS 表示过去 n 天内收盘价上涨数之和的平均值除以过去 n 天内收盘价下跌数之和的平均值。

（ii）CCI

顺势指标（Commodity Channel Index），该指标由美股技术分析家唐纳德·蓝伯特（Donald Lambert）于上世纪 80 年代提出，是指导股市投资的一种中短线指标。该指标专门测量股价是否已超出常态分布范围，波动范围为正无穷到负无穷之间，在超买超卖类指标中较特殊。其运行区间分为三类，当 CCI 值超过 100 为超买区，在-100 以下为超卖区，在（-100,100）范围内时为震荡区，属于正常交易，具体计算公式如（4-2）所示：

$$\begin{aligned} CCI(N)&=\frac{TP-ATP}{0.015\times MD}\\ TP&=\frac{High_n+Low_n+Close}{3} \end{aligned} \tag{4-2}$$

其中 TP 表示 Typical Price，$High_n$ 表示过去 n 天的最高价，Low_n 表示过去 n 天最低价，$Close$ 表示收盘价，ATP 为 TP 过去 N 天移动平均值，MD 为平均绝对偏差。

（iii）ROC

变动率指标（Rate of change），是以当日的收盘价 $Close_t$ 和 n 天前的收盘价 $Close_{t-n}$ 相比较，该指标属于反趋势指标，其本质是通过过去 n 天的收盘价涨跌幅变动情况来衡量股价买卖供需力量的强弱，进而分析股价的趋势及其是否有转势的意愿。计算公式如式（4-3）所示:

$$ROC = 100 \times \frac{Close_t - Close_{t-n}}{Close_{t-n}} \tag{4-3}$$

（iv）Williams %R

威廉指标，1973 年由 Larry Williams 提出，该指标为一个振荡指标，主要通过股价的摆动点来对其是否处于超买或超卖的情况进行判断，以提供股市趋势反转的信号。其取值范围为（0,100），通常情况下该指标超过 80 表示处于超卖状态，低于 20 表示处于超买状态，其计算公式如式（4-4）：

$$W\%R = \frac{High_n - Close}{High_n - Low_n} \tag{4-4}$$

其中 n 表示计算周期，$High_n$ 表示过去 n 天最高价，Low_n 表示过去 n 天最低价，$Close$ 表示收盘价。

（2）趋势型

指数移动平均线（Exponential Moving Average）是技术分析的主要内容。该指标的计算是一个累积的过程，包括所有的数据，历史数据对平均值的贡献程度越来越小，而近期的数据对平均值的贡献程度更大，即指数移动平均线对数据的变化更加敏感。计算公式如式（4-5）所示：

$$EMA(n)_t = \alpha \times Close_t + (1-\alpha) \times EMA(n)_{t-1} \tag{4-5}$$

其中 $EMA(n)_t$ 表示第 t 天指数移动平均值，n 表示计算该值用到的天数，其中 α 表示平滑指数，一般取 $\alpha = \frac{2}{n+1}$。

（3）均线型

MA（Moving Average）为算术移动平均线，其计算公式如式（4-6）所示：

$$MA(n)_t = \frac{Close_t + \cdots + Close_{t-n}}{n} \tag{4-6}$$

其中 n 表示计算移动平均线用到的天数，$Close_t$ 为第 t 天收盘价，$Close_{t-n}$ 为过去第 n 天收盘价。

（4）成交量型

OBV（On Balance Volume）为能量潮指标，该指标思想为通过成交量的变动情况来预测未来的股票价格运动趋势。其计算公式如式（4-7）所示:

$$OBV(n)=\sum_{i=1}^{n} volume_i \cdot sign(i)$$
$$sign(i)=\begin{cases}1, close_i > close_{i-1}\\ -1, close_i \le close_{i-1}\end{cases} \quad (4\text{-}7)$$

选取的全部因子如表 4-2、4-3、4-4 所示：

表 4-2 选取的全部行情因子

行情因子名称	行情因子定义
open	开盘价
close	收盘价
high	最高价
low	最低价
volume	成交量
money	成交金额
pct_chg	涨跌幅
turn	换手率，指在一定时间内市场中股票转手买卖的频率，反映股票流通性强弱

表 4-3 选取的全部高频因子

高频因子类型	高频因子名称
收益分布因子	高频偏度因子
	高频峰度因子
	上行波动因子
	下行波动因子
成交量分布因子	9:30-10:00 成交量占比
	10:30-11:00 成交量占比
	13:00-13:30 成交量占比
	14:30-15:00 成交量占比

表 4-4 选取的全部技术指标

技术指标类型	技术指标简称	技术指标全称	技术指标参数
超买超卖型	RSI	相对强弱指标	n=12
	CCI	顺势指标	n=14
	ROC	变动率指标	n=15
	Williams %R	威廉指标	n=14
	BIAS	乖离率	n=12
	MTM	动量指标	n=6
趋势型	EMA	指数移动平均值	n=5
	DMI	动向指数	n=14
均线型	MA	算术移动平均值	n=5
成交量型	OBV	能量潮指标	n=20

4.2 数据预处理

当我们在实际情况下进行量化选股模型构建之前，首先要对原始数据进行一定的预处理，一方面股票数据一般都数据量非常大，另一方面是数据的来源多样化，这就造成获取到的原始数据存在着大量的缺失数据以及极端数据，同时存在数据量纲不一致的问题，如若直接进行模型的构建，可能导致蕴藏在数据中的信息无法被有效的挖掘和利用，甚至预测结果与真实结果大相径庭，因此，对于数据的预处理工作是十分重要的。在整个机器学习的过程中，数据预处理过程往往占据整个流程百分之六十以上的时间，干净的数据是建模过程成功的一半，规范的数据预处理有利于算法发掘出更多有效信息，有利于构建一个预测结果良好的量化选股模型。

4.2.1 缺失值处理

对数据进行缺失值处理是数据分析中的重要一步，对于数据量较小的情况，可以采用手工处理的方式直接进行操作，但在实际的建模过程中，所涉及的数据量往往较大，其中的缺失值比重也较大，手工处理显然是非常低效的处理方式。对于缺失值的处理主要有不处理、数据补齐处理以及删除处理三种方式。

不处理指的是对缺失数据不进行处理而直接建模的方式，对于缺失值的处理往往依靠于我们的主观估计，若处理不恰当很可能会引入新的噪声，对建模的结果产生负面的影响，因此为了保护原始的信息系统，可以采取不处理缺失值的方法直接进行训练；数据补齐处理是指用一定的值来填充空值，在数据挖掘的过程中常用的数据补齐方法有人工填写、特殊值填充、平均值填充、热卡填充（就近

补齐）、最近邻法以及回归法等；删除处理是指通过直接将含有缺失值的对象进行删除的方式来实现数据完备的目的，这种方法操作起来十分简单，对于数据集中存在某条数据有多个指标存在缺失值，并且含缺失值的对象与原始数据集的数据量相比非常小的情况是非常适用的，但是通过直接删除数据来实现数据集的完备可能会导致其中隐藏的有效信息也被删除，因此该方法还是有一定的局限性的。本文中数据集的数据缺失情况如表 4-5 所示：

表 4-5 数据缺失情况表

因子名称	缺失值总数	缺失值占比
BIAS	67856	0.116255
tech_EMA5	67811	0.116178
tech_vema5	67803	0.116165
DMI_2	61058	0.104609
WR	60383	0.103452
CCI	52744	0.090365
RSI	50303	0.086182
ROC	49940	0.085561
volume	49708	0.085163
low	49708	0.085163
pct_chg	49708	0.085163
high	49708	0.085163
amt	49708	0.085163
open	49708	0.085163
turn	49708	0.085163
tech_MA5	49660	0.085081
MTM	49622	0.085016
close	49304	0.084471
tech_obv	49227	0.084339

从表 4-5 中可以观察到，每个指标都有不同程度的缺失现象，总体上缺失值占比较小。本文中股票数据出现缺失值的原因主要可以归结为两点，第一点是股票停牌导致的数据缺失，第二点是公司退市导致的数据缺失。本文中选取的股票数据周期较长，时间跨度从 2011 年至 2020 年共十年，其中难免会出现停牌和退市的股票。由于含缺失值的对象与原始数据集的数据量相比占比非常小，同时，当某支股票停牌或退市时，该条数据将含有多个属性缺失值。因此，本文选择操作方便的直接删除法来处理缺失值，既提高了数据预处理的效率，又不会对模型

训练结果产生太大的影响。

4.2.2 数据标准化

数据标准化常用的方法有以下三种。

（1）Min-Max 标准化：通过对原始数据进行线性变换使其落在[0,1]之间，其具体公式如式（4-8）所示：

$$x^* = \frac{x - \min}{\max - \min} \tag{4-8}$$

其中 max 表示该特征里的最大值，min 表示该特征里的最小值，max- min 表示该特征的极差。

通过 Min-Max 标准化方法来做数据处理，能够较好的保留原始数据中存在的关系，这是数据归一化中最简单却很实用的方法。

（2）Z-score 标准化：又称为标准差标准化，是数据标准化中最常用的方法，在样本量较大的情况下，计算公式如式（4-9）所示：

$$y_i = \frac{x_i - \mu}{\sigma} \tag{4-9}$$

其中 μ 为原始数据的均值，σ 为原始数据的标准差。

（3）归一化：主要适用于对正项数列进行标准化，对正项序列 $x_1, x_2, \cdots, x_n$ 进行变换，其公式如式（4-10）所示：

$$y_i = \frac{x_i}{\sum_{i=1}^{n} x_i} \tag{4-10}$$

则新序列 $y_1, y_2, \cdots, y_n \in [0,1]$ 且无量纲，并且有 $\sum_{i=1}^{n} y_i = 1$。

数据标准化主要是用来解决不同数据之间量纲不同的问题，如本文中获取到的股票行情数据中，开盘价一般都为几元到几十元，而像成交量就会出现上千万甚至上亿的金额，两者之间的量纲差距非常大，若不进行处理直接建模的话，会对预测结果产生较大的影响。因此需要对行情数据进行标准化的处理，将不同的量纲和单位消除，使其成为纯数值。

通过对文献[11][26]总结发现，在进行量化投资建模过程中，多数学者都采用了 Min-Max 标准化，同样本文也采用该方法来进行数据的标准化。表 4-6 为标准化后相关信息：

表 4-6 数据描述表

因子名称	count	mean	std	min	25%	50%	75%	max
open	514798	0.009455	0.024614	0	0.002708	0.005154	0.009889	1
high	514798	0.009357	0.024254	0	0.002675	0.005101	0.009795	1
low	514798	0.009306	0.02434	0	0.002671	0.005069	0.009721	1
close	514798	0.0092	0.023964	0	0.002634	0.005013	0.009619	1
volume	514798	0.006868	0.016579	0	0.001015	0.002736	0.006806	1
amt	514798	0.007725	0.015029	0	0.001367	0.003323	0.00791	1
pct_chg	514798	0.040321	0.00552	0	0.037707	0.040147	0.042699	1
turn	514798	0.021525	0.03121	0	0.005918	0.011828	0.024476	1
MA5	514798	0.009649	0.025091	0	0.002764	0.005266	0.010095	1
EMA5	514798	0.009547	0.024826	0	0.002734	0.00521	0.009989	1
VEMA5	514798	0.010781	0.024498	0	0.001723	0.004538	0.011041	1
RSI	514798	0.50833	0.131678	0	0.416603	0.505162	0.596902	1
CCI	514798	0.505553	0.117837	0	0.413748	0.505941	0.595665	1
ROC	514798	0.124698	0.018858	0	0.114907	0.123371	0.13275	1
WR	514798	0.503639	0.28936	0	0.245474	0.5	0.762887	1
OBV	514798	0.070566	0.087692	0	0.024612	0.040077	0.077536	1
DMI	514798	0.237068	0.107669	0	0.158926	0.230425	0.306319	1
BIAS	514798	0.307343	0.024274	0	0.296743	0.307012	0.318256	1
MTM	514798	0.583316	0.006645	0	0.582385	0.583235	0.584189	1
var	514798	0.044574	0.056089	0	0.020709	0.03153	0.047761	1
skew	514798	0.527803	0.072894	0	0.515005	0.528643	0.545691	1
kurt	514798	0.049767	0.096227	0	0.011348	0.018901	0.040095	1
down ratio	514798	0.475204	0.118433	0	0.409218	0.481477	0.537917	1
vol_1	514798	0.200185	0.08085	0	0.142872	0.192234	0.24779	1
vol_3	514798	0.24331	0.100861	0	0.176939	0.224842	0.28855	1
vol_5	514798	0.252201	0.111078	0	0.178199	0.231153	0.302456	1
vol_8	514798	0.218264	0.085753	0	0.161931	0.201781	0.259231	1
earn_rate_log	514798	0.692118	0.005509	0	0.690644	0.692117	0.693635	1

4.3 数据降维

本文共选取各类指标 27 个，其中行情因子 8 个、技术指标 11 个、高频因子 8 个，在构建 XGBoost 模型的时候，并不是输入越多的因子就会得到越好的效果，相反可能因子数量过多导致数据量过于庞大，使得在建模以及后续参数调优的过程中不仅浪费大量的时间，还无法得到良好的模型，因此有必要先进行降维处理。数据降维是机器学习过程当中处理高维度特征数据的方法，通过保留对结果贡献大的特征，去除不重要的特征以及数据中的噪声来实现降维的目的。在实际的建模过程中，通过降维既保留了数据中大部分的有效信息，使得模型预测的准确度得到了保证，又能够节省大量的时间和运行成本。本文主要借鉴了其他学者的研究[27-29]，通过主成分分析的方法对数据进行降维处理。

4.3.1 主成分分析降维

主成分分析方法（PCA）是数据降维领域中应用最为广泛的方法，其主要思想是将 n 维特征映射到 k 维上，即在原有的 n 维特征基础上重新构造出 k 维新的特征，这 k 维新特征被称为主成分。该方法的主要任务就是从原始空间中不断的寻找相互正交的坐标轴，每一次找到的新坐标均与数据本身有着密切的联系。通过 n 次旋转之后，大部分解释方差都包含在前 k 个坐标轴中，后面的坐标轴所含的解释方差几乎为零。因此，可以将含有解释方差较少的坐标轴删除，只保留前 k 个包含绝大部分方差的坐标轴，即保留能够解释大部分原始数据信息的主成分，从而实现对原始数据的降维处理。

主成分分析法的具体降维步骤如下:

（1）数据标准化

原始数据集中很多特征往往有着不同的数据量纲，不同特征之间的单位差距较大，因此在进行主成分分析前需对数据进行标准化处理。设经过标准化的矩阵为：$X^{*}=\left(X_{1}^{*},X_{2}^{*},\cdots,X_{m}^{*}\right)$。

（2）求特征值和特征向量

首先求 X^{*} 的协方差矩阵 Σ，具体计算公式如式（4-11）所示:

$$\Sigma=\frac{1}{m-1}\left(X^{*}\right)^{T}X^{*} \tag{4-11}$$

接下来根据 $\left|R-\lambda_{j}E\right|=0$ 求特征值 λ_{j} 及特征向量 $\mu_{ij}\left(j=1,2,\cdots,m\right)$。

具体表示如式（4-12）所示：

$$\lambda_1 \geq \lambda_2 \cdots \geq \lambda_m \geq 0$$
$$U_i = (\mu_{i1}, \mu_{i2}, \cdots, \mu_{im}) \quad (4\text{-}12)$$

将主成分记为$F_1, F_2, \cdots, F_m$，是经过标准化的特征向量$(X_1^*, X_2^*, \cdots, X_m^*)$的线性组合，具体计算方式如式（4-13）所示：

$$\begin{aligned} F_1 &= \mu_{11}X_1^* + \mu_{12}X_2^* + \cdots + \mu_{1m}X_m^* \\ F_2 &= \mu_{21}X_1^* + \mu_{22}X_2^* + \cdots + \mu_{2m}X_m^* \\ F_3 &= \mu_{31}X_1^* + \mu_{32}X_2^* + \cdots + \mu_{3m}X_m^* \\ &\vdots \\ F_m &= \mu_{m1}X_1^* + \mu_{m2}X_2^* + \cdots + \mu_{mm}X_m^* \end{aligned} \quad (4\text{-}13)$$

其中$(1)\mu_{i1}^2 + \mu_{i2}^2 + \cdots + \mu_{im}^2 = 1$，$(2)Cov(F_i, F_j) = 0(i, j = 1,2,\cdots,m, 且 i \neq j)$即各主成分是相互独立的，$(3)Var(F_1) \geq Var(F_2) \geq \cdots \geq Var(F_m)$即第一主成分携带最多信息，第二主成分其次，依次递减。

（3）计算各主成分解释方差，并根据设定的累计解释方差决定主成分个数。

（4）计算各主成分权重。

4.3.2 降维结果展示

图 4-2 为含有高频因子的累积方差贡献率曲线图：

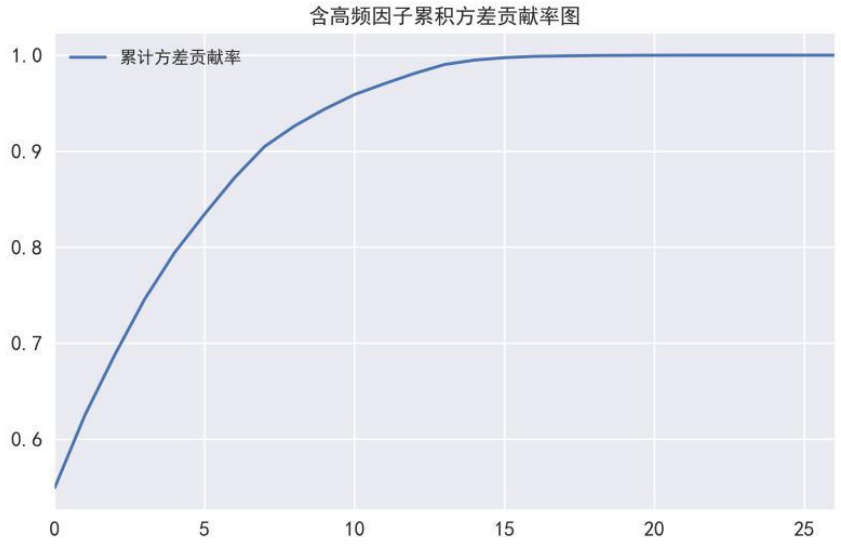

图 4-2 含高频因子模型累积方差贡献图

可以观察到因子数量在 15 个左右时几乎就可以解释所有的变量，最终选取

了 13 个主成分，其携带的累积可解释方差达 98%，表 4-7 展示了选取的 13 个主成分携带的可解释方差值：

表 4-7 主成分可解释方差表

主成分	F1	F2	F3	F4	F5	F6	F7
可解释方差值	0.1149	0.0157	0.0132	0.0121	0.0102	0.0084	0.0079
主成分	F8	F9	F10	F11	F12	F13	
可解释方差值	0.0068	0.0045	0.00378	0.0032	0.0024	0.0022	

图 4-3 为未加入高频因子的累积方差贡献率曲线图：

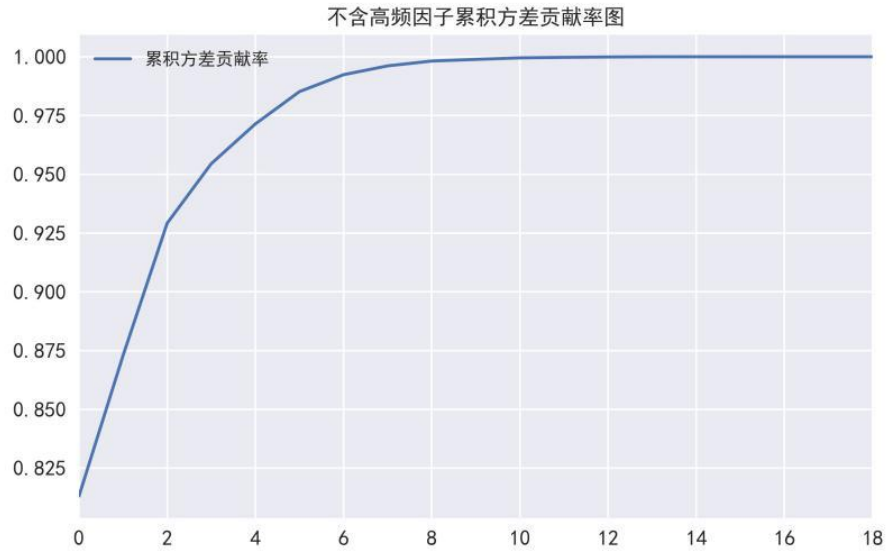

图 4-3 不含高频因子模型累积方差贡献图

可以观察到，在因子数量达到 8 个左右时，几乎可以解释全部的变量，最终选取了 6 个主成分，其携带的累积可解释方差达 98%，表 4-8 展示了选取的 6 个主成分携带的可解释方差值：

表 4-8 主成分可解释方差表

主成分	F1	F2	F3	F4	F5	F6
可解释方差	0.114603	0.00843	0.007927	0.003567	0.002384	0.001937

通过上述分析不难看出，只用少数的主成分就可以解释全部变量中的大部分信息，因此对于数据的降维处理是十分有必要的。同时，对比加入高频因子前后的降维结果，对加入高频因子的数据集进行降维，维度由 27 维降至 13 维，对未加入高频因子的数据集降维，维度由 19 维降至 6 维，因此可以推断出高频因子中携带了大量的传统因子所涉及不到的新信息，能够为后续的建模提供更多行情

数据以及技术指标无法提供的信息，一定程度上能够体现加入高频因子的必要性。

4.4 模型训练与优化

通过上述一系列的数据获取、因子选择以及数据预处理，数据中存在的问题基本已经得到了解决，接下来要实现量化选股模型的构建。在该部分将选取的各类股票因子作为输入变量，股票收益率作为输出变量，将数据集按照 2:1 的比例划分为训练集和测试集，训练集数据共计 360359 条，测试集数据共计 154439 条。

XGBoost 算法自陈天奇博士提出后，吸引了大量学者的目光，尤其是在最近几年的各类数据大赛中脱颖而出。该算法为集成算法，其训练速度较快并且预测准确度较高，能够良好的解决分类问题与回归问题；在对目标函数进行优化的同时做了预剪枝，减少了过拟合现象的发生；此外，XGBoost 的基模型一般选择 CART 分类回归树，其逻辑清晰且理论优美，适合用于金融领域。目前已有越来越多的学者[30-31]尝试将该算法应用于量化投资领域，本文也采用 XGBoost 算法来对股票进行收益率的预测，选取网格搜索方法来进行参数优化，以便获取最优模型。

4.4.1 模型参数优化

（1）参数优化步骤

在 XGBoost 算法中存在着大量的超参数，选择不同的参数值对于模型的影响是非常大的，模型的预测效果以及泛化能力是否良好在很大程度上取决于这些超参数的选择，因此参数的优化是 XGBoost 建模过程中必不可少的一个步骤。由于参数调整的顺序对于调参的效果以及模型的预测的效果有着极其重要的影响，因此在参数调整之前非常有必要了解一下 XGBoost 算法的重要参数以及其具体的作用，在优化过程中优先选择那些对模型影响较大的参数进行调整。表 4-9 展示了本文中选取的需要进行调整的具体参数地含义及其默认值。

本文首先采用默认参数进行建模，接下来再采用网格搜索与 XGBoost 中自带的 CV 函数相结合来进行参数调优。网格搜索法是一种基于交叉验证法的调参方法，它是在交叉验证法循环估计和评估方法的基础上发展而来的。在进行网格搜索调参的过程中，首先需要给定所调参数一个具体的范围，在范围内的参数值会交叉形成网格点，接下来采用交叉验证的方法对每个网格点的参数进行多次的验证以及评估，将误差数据取平均值进行评估。循环进行多次，直到找到给定范围内的参数的最优值。当数据集的体量较大时，使用网格搜索法调参往往会花费大量的时间。

表 4-9 XGBoost 常用参数含义

参数	含义及默认值
n_estnnators	弱学习器即树模型的数量，默认值为 100
max_depth	树的最大深度，可避免过拟合，值越大模型会学到更具体更局部的样本，默认值为 6
min_child_weight	决定最小叶节点样本权重和，可避免过拟合，较大时可以避免模型学习到局部的特殊样本，默认值为 1
learning rate	学习率，可防止过拟合，更新过程中用到的收缩步长，在每次提升计算后，算法会直接获得新特征权重，该参数通过缩减特征权重使提升计算过程更保守，默认值为 0.3
gamma	用于控制是否后剪枝，该参数指定了节点分裂所需的最小损失函数下降值。其值越大，算法越保守，默认值为 0
subsample	样本的采样率，用于训练模型的子样本占整个样本集合的比例。可防止过拟合，该参数值越小，算法越保守，可避免过拟合，默认值为 1
colsample bytree	每次生成树时随机抽样特征的比例，默认值为 1

下面展示了本文中 XGBoost 参数调优的具体步骤：

步骤一：确定 learning rate 和 tree_based 参数调优的估计器数目。为了确定 Boosting 参数，需要先给其他参数一个初始值，本文中参数的初始值均设置为其默认值。具体取值如表 4-10 所示：

表 4-10 默认参数值

参数名称	参数取值
max_depth	6
min_child_weigh	1
gamma	0
subsample	1
colsample_bytree	2
learning rate	0.1

步骤二：max_depth 和 min_child_weight 参数调优。这两个参数分别决定了树的最大深度和最小叶节点样本权重和，对模型的预测结果起到了至关重要的作用，因此需要优先进行调整。首先使用网格搜索进行大范围的粗略估计，接下来再将范围调小进行测试，在这个部分一般需要的测试时间较长。具体选取数值如表所 4-11 示：

表 4-11 参数优化取值

大范围		小范围	
max_depth	range（3,10,2）	max_depth	[4,5,6]
min_child_weight	range（1,6,2）	min_child_weight	[4,5,6]

步骤三：gamma 参数调优。在已经调整好上述几个参数的基础上进行 gamma 参数的调整，其取值范围较大，本文主要选取[0,0.1,1,5]几个值进行网格搜索。

步骤四：调整 subsample 和 colsample_bytree 参数。本文选取[0.6,0.7,0.8,0.9]四个值对两个参数进行网格搜索。

步骤五：正则化参数调优。gamma 函数提供了对于降低过拟合更有效的方法，取参数 reg_alpha 值为[1e-5, 1e-2, 0.1, 1, 100]进行网格搜索。

步骤六：降低学习速率，增加决策树个数。最后使用较低的学习速率，同时使用更多的决策树，learning rate 取值为 0.01 和 0.05，n_ estimators 取值为 500,1000 以及 1500，该步骤通过 Xgboost 中 CV 函数来实现。

（2）参数优化结果

通过上述一系列的参数优化过程得到了最优参数，模型经过参数优化后得到的最优值如表 4-12 所示：

表 4-12 参数优化结果

参数名称	最优值（加入高频因子）	最优值（未加入高频因子）
n_ estimators	1500	500
learning rate	0.1	0.1
max_depth	7	3
min_child_weight	1	1
gamma	0	0
subsample	0.8	1
colsample_bytree	1	1

图 4-4 为加入高频因子使用最终优化参数得到的特征重要性排名图：

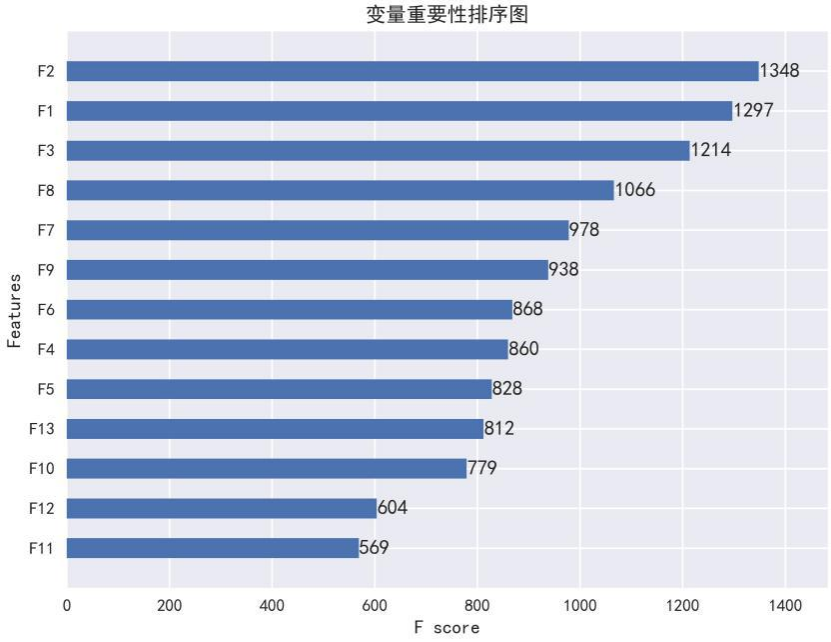

图 4-4 含高频因子模型变量重要性排序图

图 4-5 为未加入高频因子使用最终优化参数得到的特征重要性排名图：

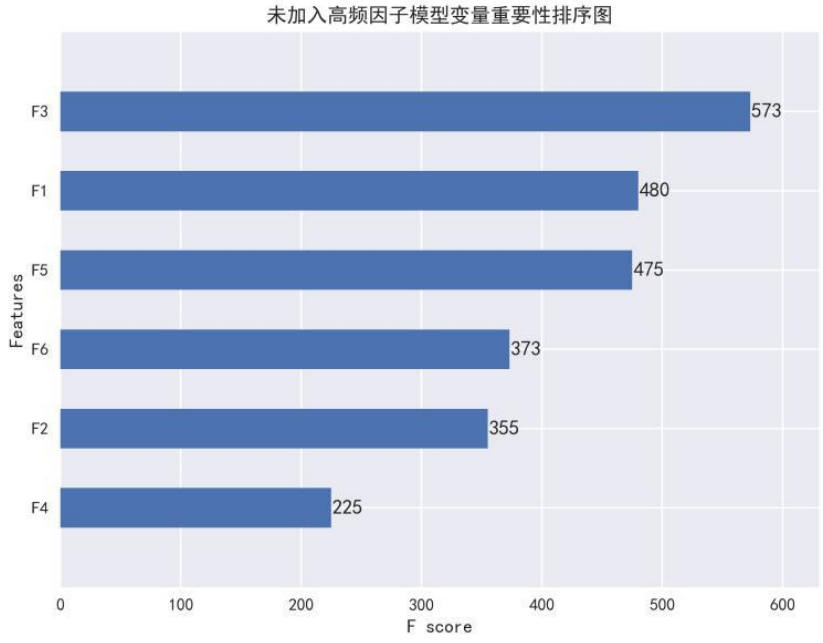

图 4-5 不含高频因子模型变量重要性排序图

根据上述调参，模型得到了一定的优化，两个模型的预测评价指标均方误差均有一定程度的减小，但是优化程度较小，可能是由于参数采用默认值时的模型已经具有了良好的预测能力，因此后续的参数优化效果不是很明显。这也同样从

另一角度提醒我们，在机器学习的过程中，是不可能完全通过参数优化使得模型的预测效果得到大幅度的提升的，因此更重要的步骤其实是前期对于特征的选择以及数据的预处理步骤，在实际的机器学习过程中，应当更注重数据的质量问题。

4.5 模型预测与结果分析

通过上述的 XGBoost 模型构建以及参数的调整，得到了具有比较好预测效果的模型，接下来主要对加入高频因子和未加入高频因子的模型进行对比分析，观察高频因子对于模型构建的影响。同时，对 XGBoost 算法与传统线性回归和逻辑回归模型的预测效果进行对比，探究 XGBoost 算法的预测能力。

4.5.1 模型评价

无论选取多么高效的模型，其预测结果总是会有一定的误差，通常将预测值与真实值之间的误差大小作为评价模型预测效果的指标，当预测值与真实值的差距越小时，模型的预测效果越好。本文选取了均方误差（MSE）、均方根误差（RMSE）以及平均绝对误差（MAE）作为模型评价指标。

均方误差是真实值与预测值差值的平方然后求和平均；均方根误差为预测值与真实值偏差的平方与观测次数比值的平方根，即均方误差的平方根；平均绝对误差为预测值和真实值之间绝对误差的平均值。当预测值与真实值越靠近时这三个评价指标的值越小，表示该模型的预测效果越好、性能越好，当模型误差越大时，这三个评价指标的值越大。表 4-13 为 XGBoost 模型评价指标汇总：

表 4-13 XGBoost 模型评价指标汇总

评价指标	参数优化情况	未加入高频因子	加入高频因子
MSE	优化前	2.777970e-05	1.889419e-05
	优化后	2.199769e-05	1.407442e-05
RMSE	优化前	0.005271	0.004347
	优化后	0.004690	0.003752
MAE	优化前	0.001932	0.001820
	优化后	0.001905	0.001635

通过观察表 4-13 可以看出，纵向对比，均方误差值、均方根误差值以及平均绝对误差值在参数优化后都有一定程度的减小，如对于加入高频因子的模型来说，在调整参数后，其均方误差值由1.889419×10^{-5}减小到1.404772×10^{-5}，即参数优化对于 XGBoost 模型预测精确度以及 XGBoost 模型性能的提升有一定的效果。横向对比，在加入高频因子之后 XGBoost 模型预测的准确度有了明显的提

升，如对于参数优化后的模型来说，在加入高频因子之后，模型均方误差值由 2.199769×10^{-5} 减小到 1.404772×10^{-5}。通过加入高频因子带来的均方误差的降低值甚至超过通过参数优化带来的降低值，即高频因子对于优化模型的效果显著。也在一定程度上表明，高频因子中携带着某些传统因子所不具备的新信息，该信息对于投资者预测下期收益有一定的帮助，即在进行量化投资的过程中，加入高频因子是有利于投资者获得更多市场信息，同时提高对未来收益率预测的准确性，因此对日内高频数据的研究是十分有必要的。

表 4-14 展示了 XGBoost 算法与传统的线性回归以及 Logistic 回归模型预测准确率的对比情况，三种模型均使用含有高频因子的数据进行建模。

表 4-14 不同算法评价指标对比

算法	MSE	RMSE	MAE
XGBoost	1.407442e-05	0.003752	0.001635
逻辑回归	0.479062	0.692143	0.692125
线性回归	2.320776e-05	0.004817	0.002091

通过观察表 4-14 不难看出，XGBoost 算法的预测精度是最高的，该模型的性能是最好的，其均方误差仅为 1.407442×10^{-5}，逻辑回归的预测精度较差，其均方误差为 0.479062。在面对复杂度极高的非线性股票数据时，与传统线性回归和逻辑回归相比，XGBoost 算法展示了其良好的性能，因此，将机器学习方法应用于量化投资领域能够更深入的探究股票数据内在的规律，为投资者提供更多有效信息，并获得超额收益。

4.5.2 股票收益评价

通过建立上述模型，最终还是要回到选股的根本目标上，为投资者挑选出较为优质的股票来进行投资，即选取的股票有最大的可能获得超额的收益。通过上一节的对比分析可知，加入高频因子的 XGBoost 模型更有效，表 4-14 中展示了加入了高频因子的模型预测出的收益率值排名靠在前 30 名的股票，具体股票代码及其年化收益如表 4-15 所示。

表 4-15 中展示了部分在 2020 年沪深 300 指数成分股中年化对数收益率表现较好的股票码及其收益率的预测值，排名第一的股票年化收益率达到了 21.495，排名第三十的股票收益率达到 8.942，可见整体上来看模型的预测值呈现出股票收益率普遍较好并且分布均匀的景象，但是其实在现实的股市中，股票的年化收益的差距还是比较大的，同时分布也不会呈现如此均匀的情况。

表 4-15 收益率前 30 名股票一览表

排名	股票代码	股票名称	年化收益率	排名	股票代码	股票名称	年化收益率
1	600655.SH	豫园股份	21.49541	16	600436.SH	片仔癀	10.6878
2	600999.SH	招商证券	21.42551	17	601111.SH	中国国航	10.4527
3	600893.SH	航发动力	20.00878	18	002179.SZ	中航光电	10.26045
4	000786.SZ	北新建材	17.76162	19	600845.SH	宝信软件	10.14106
5	002230.SZ	科大讯飞	17.56962	20	000703.SZ	恒逸石化	9.881828
6	300059.SZ	东方财富	15.54932	21	002841.SZ	视源股份	9.835447
7	000066.SZ	中国长城	14.81362	22	002594.SZ	比亚迪	9.822754
8	300124.SZ	汇川技术	14.28656	23	002371.SZ	北方华创	9.795769
9	600760.SH	中航沈飞	13.27749	24	000069.SZ	华侨城 A	9.72774
10	601877.SH	正泰电器	13.25304	25	300033.SZ	同花顺	9.716461
11	002044.SZ	美年健康	12.45216	26	600196.SH	复星医药	9.360397
12	000768.SZ	中航西飞	11.64433	27	002739.SZ	万达电影	9.273116
13	601788.SH	光大证券	11.20874	28	600588.SH	用友网络	8.999107
14	002032.SZ	苏泊尔	10.96625	29	600998.SH	九州通	8.973388
15	300144.SZ	宋城演艺	10.927	30	600111.SH	北方稀土	8.941574

4.6 本章小结

本章为量化选股实证分析部分，选取了 2011 年 1 月 1 日至 2020 年 12 月 31 日共 10 年的沪深 300 指数成分股的日频行情数据，构建了基于 XGBoost 算法的多因子选股模型，将行情数据、技术指标以及在第三章中通过单因子检验发掘出的具有良好选股效果的高频因子作为输入变量，股票下期收益率作为输出变量来进行建模，最终挑选出下期收益率前 30 名的股票供投资者参考。同时，进行了两次对比研究，一是对加入高频因子和未加入高频因子的模型预测效果进行对比，发现加入高频因子的模型具有更准确的预测效果，即高频因子中携带了大量与股票下期收益相关的信息，值得投资者们的进一步关注与研究；二是对 XGBoost 算法与传统线性回归和逻辑回归模型的预测效果进行对比，研究发现 XGBoost 模型有着明显优于其他两个模型的预测效果，在针对复杂、非线性的股票数据建模方面有着更加优良的特性。

第五章 总结与展望

5.1 研究总结

近几年来，随着我国证券市场的不断发展以及制度的完善，量化投资逐渐受到国内的投资者和投资机构的重视，越来越多的投资机构加入到了量化投资的行列当中。与此同时，计算机科学技术作为新兴力量也在蓬勃的发展中，各行各业都在尝试着将人工智能方法融入到自己的领域当中。证券市场具有非线性、噪声大以及复杂性等特点，这就使得传统的统计预测模型在对股票价格进行预测时有不可避免的弊端，而机器学习方法能够有效的处理非线性的数据，其良好的预测效果吸引了越来越多的投资者。本文通过对机器学习方法以及量化投资理论的研究和学习，引入了对高频因子的研究，同时构建了基于 XGBoost 的量化选股模型，具体工作以及结论如下：

（1）引入对日内高频数据的研究，选取沪深 300 指数成分股从 2011 年 1 月 1 日至 2020 年 12 月 31 日共十年的 1 分钟数据，构造了收益率分布因子、成交量分布因子以及量价复合因子三类高频因子，共计十六个。分别对上述因子进行单因子有效性检验，最终选取出八个具有良好选股效果的高频因子。

（2）选取沪深 300 指数成分股从 2011 年 1 月 1 日至 2020 年 12 月 31 日 10 年的日频数据进行基于 XGBoost 算法的多因子选股，输入变量为行情因子、技术指标以及具有选股能力的高频因子，输出变量为股票收益率，最终选取具有较高收益率的股票。

（3）采用主成分分析的方法对样本数据进行降维处理，含有高频因子的数据维度从 27 维降至 13 维，不含高频因子的数据维度从 19 维降至 6 维，其携带的累计可解释性方差均达 98%。

（4）考虑到 XGBoost 算法涉及参数较多，不同参数对于模型预测效果有明显影响，采用网格搜索的方法进行参数优化，最终优化后的模型性能较优化前均有一定的提升。采用均方误差、均方根误差以及平均绝对误差作为模型评价指标，对比分析了加入高频因子前后模型的预测效果，以及不同算法得到模型的预测效果。

（5）对高频因子的作用进行了分析，其优化模型的效果显著，即高频因子中携带着某些传统因子所不具备的新信息，该信息对于投资者预测下期收益有一定的帮助，因此对日内高频数据的研究是十分有必要的。

（6）对 XGBoost 算法与传统线性回归和逻辑回归模型的预测效果进行了对比，发现 XGBoost 模型预测效果更好，更能够发掘在复杂金融时间序列中的信息，相比于传统的线性模型，在处理非线性的股票数据方面更有优势。

5.2 研究不足与展望

除了上述的工作总结之外，本文中还存在很多的不足之处，希望能够在未来进一步的改进：

（1）在本文中选取的因子数量有限，还可以再加入其他类型的因子，如上市公司财务相关的数据构造的因子等，为模型提供更多的信息，提高模型预测准确率。

（2）在股票数据的选择上有一定局限性，只选择了比较有代表性的沪深 300 指数的成分股进行实证分析。实证结果表明高频因子中携带了低频因子所不具备的新信息，具有一定的推广价值，未来可以将对高频因子的研究进一步的推广至所有的 A 股，全面的数据更有利于刻画我国的股票市场，使得得到的结果更具有实际价值。

（3）在方法的选择上，具有良好预测效果的机器学习方法有很多，只选择了 XGBoost 一种方法，未来可以尝试使用 LightGBM、神经网络等算法，对比不同机器学习方法在量化投资方面的应用效果。

（4）模型的评价指标选择较少，不能全面的衡量模型效果，可以尝试使用更多的指标来衡量模型预测的准确率。

（5）最后仅给出未来具有良好收益率的股票，后续可以进一步使用选出来的股票进行投资组合的构建，将风险考虑进去，使得得到的结果更具有参考价值。

参考文献

[1] Fama E F, French K R. Common risk factors in the returns on stocks and bonds[J]. North-Holland, 1993, 33(1):3-56.

[2] Fama E F, French K R. A five-factor asset pricing model[J]. Journal of Financial Economics, 2015, 116(1):1-22.

[3] Barth M E, Beaver W H, Landsman W R. A Structural Analysis of Pension Disclosures under SFAS 87 and Their Relation to Share Prices[J]. Financial Analysts Journal, 1993, 49(1):18-26.

[4] Piotroski J D. Value Investing: The Use of Historical Financial Statement Information to Separate Winners from Losers[J]. Journal of Accounting Research, 2000, 38:1-41.

[5] 刘毅. 因子选股模型在中国市场的实证研究[D]. 复旦大学, 2012.

[6] 高鑫. 基于投资者情绪的行为资产定价模型及其实证研究[D]. 北京科技大学, 2017.

[7] 陈志文. 基于人工智能的量化选股模型设计与应用研究[D]. 北京化工大学, 2019.

[8] Kim K, Han I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index[J]. Expert Systems With Applications, 2000, 19(2):125-132.

[9] 苏治, 傅晓媛. 核主成分遗传算法与SVR选股模型改进[A]. 中国人民大学国际货币研究所.International Monetary Institute Working Papers(2010-2014年合辑)[C]. : 中国人民大学国际货币研究所, 2014:18.

[10] 李想. 基于XGBoost算法的多因子量化选股方案策划[D]. 上海师范大学, 2017.

[11] 黄恒秋.基于高频数据的支持向量机量化择时预测模型[J]. 科技经济导刊, 2016(13):29.

[12] 杨健. 基于机器学习的量化投资策略分析系统研究与实现[D]. 上海交通大学, 2017.

[13] 王华琴. 基于机器学习的量化选股及效果评价[D]. 电子科技大学, 2020.

[14] 黄志辉. 基于卷积神经网络的量化选股模型研究[D]. 浙江大学, 2019.

[15] Chen T, Tong H, Benesty M. xgboost: Extreme Gradient Boosting[J]. 2017,1.

[16] 冯佳睿, 袁林青. 选股因子系列研究（十九）——高频因子之股票收益分布

特征[R]. 海通证券研究所, 2017.
[17] 冯佳睿, 袁林青. 选股因子系列研究（二十五）——高频因子之已实现波动分解[R]. 海通证券研究所, 2017.
[18] 冯佳睿, 袁林青, 姚石, 罗蕾, 余浩淼. 选股因子系列研究（六十九）——高频因子的现实与幻想[R]. 海通证券研究所, 2020.
[19] 冯佳睿, 姚石. 高频量价因子在股票与期货中的表现[R]. 海通证券研究所, 2018.
[20] 冯佳睿, 姚石. 选高频因子在不同周期和域下的表现及影响因素分析[R]. 海通证券研究所, 2019.
[21] 汤家正. 基于数据挖掘和 XGBoost 算法的量化多因子对冲模型研究[D]. 山东大学, 2020.
[22] 韩雨薇. 基于深度学习的多因子股票风险预测方法研究[D]. 浙江大学, 2020.
[23] 林炜. 基于机器学习的量化择时与量化选股方法研究[D]. 厦门大学, 2017.
[24] 白凯敏. 神经网络和深度学习在量化投资中的应用[D]. 山东大学, 2016.
[25] 赵群. 基于深度学习的智能投资模型与方法[D]. 哈尔滨工业大学, 2018.
[26] 田浩. 基于XGBoost的沪深300量化投资策略研究[D]. 上海师范大学, 2018.
[27] 陈盟. 基于核主成分分析和支持向量回归对每日和每分钟股票价格的预测[D]. 兰州大学, 2020.
[28] 贾秀娟.基于随机森林的支持向量机量化选股[J]. 区域金融研究, 2019(01):27-30.
[29] 王小红. 基于机器学习的量化选股研究[D]. 华中师范大学, 2019.
[30] 祝养豹. 基于 XGBoost 和 LightGBM 算法的多因子选股方案设计[D]. 南京大学, 2020.
[31] 张毅,田浩.XGBoost 在量化选股中的应用研究[J]. 金融管理研究, 2020(02):122-132.
[32] Hushani P. Using Autoregressive Modelling and Machine Learning for Stock Market Prediction and Trading[J]. Advances in Intelligent Systems and Computing, 2018:767-774.
[33] Ticknor J L. A Bayesian regularized artificial neural network for stock narket forecasting[J]. Expert Systems with Applications. 2013.40(14):5501-5506.
[34] Pastor L, Veronesi P. Stock Valuation and Learning about Profitability[J]. Journal of Finance, 2003, 58 (5): 1749-1789.
[35] Torlay L, Perrone M, Thomas E, Baciu M. Machine learning–XGBoost analysis of language networks to classify patients with epilepsy[J]. Brain Informatics, 2017,

4(3):11.

[36] Chen T, Guestrin C. XGBoost: A Scalable Tree Boosting System[J]. 2016, 3.

[37] 王霄. 基于 SVM 的沪深 300 指数量化择时策略实证研究[D]. 兰州大学, 2019.

[38] 何清, 李宁, 罗文娟, 史忠植. 大数据下的机器学习算法综述[J]. 模式识别与人工智能, 2014, 27(04):327-336.

[39] 于航. 基于支持向量机模型的股指期货高频交易策略研究[D]. 北京理工大学, 2015.

[40] 胡谦. 基于机器学习的量化选股研究[D]. 山东大学, 2016.

[41] 王淑燕, 曹正凤, 陈铭芷. 随机森林在量化选股中的应用研究[J]. 运筹与管理, 2016, 25(03):163-168+177.

[42] 肖晞晖. 基于大数据和机器学习的量化选股模型研究[D]. 华中师范大学, 2018.

[43] 费洋. A 股市场多因子选股量化模型构建及其检验[D]. 浙江大学, 2018.

[44] 李赢. 基于集成学习的量化因子选股分析[D]. 西南财经大学, 2019.

[45] 谢合亮. LSTM 在多因子量化投资模型中的改进及应用研究[D]. 中央财经大学, 2019.

[46] 李兴有. 基于人工智能的量化多因子模型的拓展及在中国股票市场上的应用[D]. 中国社会科学院研究生院, 2020.

[47] 刘佳琪, 张建. 基于机器学习的多因子选股模型[J]. 时代金融, 2020(17):99-103.

附录

高频因子的具体构建方式相关代码如下所示：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as st
import os
import glob
#高频收益因子
#步骤 1
stocks={'000001.XSHE','000002.XSHE','000063.XSHE','000066.XSHE','000069.XS
HE','000100.XSHE','000157.XSHE','000166.XSHE','000333.XSHE','000338.XSHE','
000425.XSHE','000538.XSHE','000568.XSHE','000596.XSHE','000625.XSHE','0006
27.XSHE','000651.XSHE','000656.XSHE','000661.XSHE','000671.XSHE','000703.X
SHE','000708.XSHE','000723.XSHE','000725.XSHE','000728.XSHE','000768.XSHE',
'000776.XSHE','000783.XSHE','000786.XSHE','000858.XSHE','000860.XSHE','0008
76.XSHE','000895.XSHE','000938.XSHE','000961.XSHE','000963.XSHE','000977.X
SHE','001979.XSHE','002001.XSHE','002007.XSHE','002008.XSHE','002024.XSHE',
'002027.XSHE','002032.XSHE','002044.XSHE','002049.XSHE','002050.XSHE','0021
20.XSHE','002129.XSHE','002142.XSHE','002146.XSHE','002153.XSHE','002157.X
SHE','002179.XSHE','002202.XSHE','002230.XSHE','002236.XSHE','002241.XSHE',
'002252.XSHE','002271.XSHE','002304.XSHE','002311.XSHE','002352.XSHE','0023
71.XSHE','002384.XSHE','002410.XSHE','002414.XSHE','002415.XSHE','002422.X
SHE','002456.XSHE','002460.XSHE','002463.XSHE','002475.XSHE','002493.XSHE',
'002508.XSHE','002555.XSHE','002558.XSHE','002594.XSHE','002600.XSHE','0026
01.XSHE','002602.XSHE','002607.XSHE','002624.XSHE','002673.XSHE','002714.X
SHE','002736.XSHE','002739.XSHE','002773.XSHE','002812.XSHE','002821.XSHE',
'002841.XSHE','002916.XSHE','002938.XSHE','002939.XSHE','300003.XSHE','3000
14.XSHE','300015.XSHE','300033.XSHE','300059.XSHE','300122.XSHE','300124.X
SHE','300136.XSHE','300142.XSHE','300144.XSHE','300347.XSHE','300408.XSHE',
'300413.XSHE','300433.XSHE','300498.XSHE','300529.XSHE','300601.XSHE','3006
28.XSHE','300676.XSHE','600000.XSHG','600004.XSHG','600009.XSHG','600010.
XSHG','600011.XSHG','600015.XSHG','600016.XSHG','600018.XSHG','600019.XS
HG','600025.XSHG','600027.XSHG','600028.XSHG','600029.XSHG','600030.XSHG',
'600031.XSHG','600036.XSHG','600048.XSHG','600050.XSHG','600061.XSHG','600
066.XSHG','600068.XSHG','600085.XSHG','600104.XSHG','600109.XSHG','600111.
XSHG','600115.XSHG','600118.XSHG','600150.XSHG','600161.XSHG','600176.XS
HG','600177.XSHG','600183.XSHG','600196.XSHG','600208.XSHG','600233.XSHG',
'600271.XSHG','600276.XSHG','600297.XSHG','600299.XSHG','600309.XSHG','600
332.XSHG','600340.XSHG','600346.XSHG','600352.XSHG','600362.XSHG','600369.
```

```
XSHG','600383.XSHG','600390.XSHG','600406.XSHG','600436.XSHG','600438.XS
HG','600482.XSHG','600487.XSHG','600489.XSHG','600498.XSHG','600519.XSHG',
'600522.XSHG','600547.XSHG','600570.XSHG','600584.XSHG','600585.XSHG','600
588.XSHG','600600.XSHG','600606.XSHG','600637.XSHG','600655.XSHG','600660.
XSHG','600690.XSHG','600703.XSHG','600705.XSHG','600741.XSHG','600745.XS
HG','600760.XSHG','600763.XSHG','600795.XSHG','600809.XSHG','600837.XSHG',
'600845.XSHG','600848.XSHG','600872.XSHG','600886.XSHG','600887.XSHG','600
893.XSHG','600900.XSHG','600919.XSHG','600926.XSHG','600958.XSHG','600998.
XSHG','600999.XSHG','601006.XSHG','601009.XSHG','601012.XSHG','601021.XS
HG','601066.XSHG','601088.XSHG','601100.XSHG','601108.XSHG','601111.XSHG',
'601117.XSHG','601138.XSHG','601155.XSHG','601166.XSHG','601169.XSHG','601
186.XSHG','601198.XSHG','601211.XSHG','601216.XSHG','601225.XSHG','601229.
XSHG','601231.XSHG','601238.XSHG','601288.XSHG','601318.XSHG','601328.XS
HG','601336.XSHG','601360.XSHG','601377.XSHG','601390.XSHG','601398.XSHG',
'601628.XSHG','601933.XSHG','603899.XSHG','601788.XSHG','601808.XSHG','601
872.XSHG','601877.XSHG', '601881.XSHG'}

for code in stocks:
    data=pd.read_csv('D:/data_1min/'+code+'.csv')
    data.columns=['time/1min','open','close','low','high','volume','money']
#计算对数收益率
    data['earn_rate_log']=np.log((data['close']/data['close'].shift(1)))
#提取原表日期列，分成日期和时间两列
    data_new = pd.DataFrame(data['time/1min'].str.split(' ',1).tolist(),columns =
['date','time'])
    data=data.drop(axis=1,labels=['time/1min'])
    data_1=pd.concat([data_new,data],axis=1)
    data_1['earn']=data_1['close']-data_1['open']
    data_2=data_1[['date','earn_rate_log','earn']]
#高频方差因子
    data_3 = data_2.groupby(by=['date'])['earn_rate_log'].var()#方差
    data_3=pd.DataFrame(data_3)
    data_3.columns=['var']
#高频偏度因子
    data_4= data_2.groupby(by=['date'])['earn_rate_log'].skew() #偏度
    data_4=pd.DataFrame(data_4)
    data_4.columns=['skew']
#高频峰度因子
    group_by=data_2.groupby('date')
    data_5=group_by['earn_rate_log'].agg([st.kurtosis])
    data_5 = pd.DataFrame(data_5)
#print(data_5.head())
#三列因子进行合并,加入收益
    data_7= data_2.groupby(by=['date'])['earn'].mean()#收益每日均值
```

```
        data_7= pd.DataFrame(data_7)
        data_6=pd.concat([data_3,data_4,data_5,data_7],ignore_index=True,axis=1)
        data_6.columns=['var','skew','kurt','earn']
        filename='d:/data_earn/'+code+'.csv'
        data_6.to_csv(filename)
        mean_1=pd.DataFrame(data_6.mean())
        mean_1.columns=[code]
        filename='d:/data_earn_mean/'+code+'.csv'
        mean_1.to_csv(filename)

#步骤 2
for code in stocks:
        df=pd.read_csv('d:/data_earn/'+code+'.csv')
        df_new = pd.DataFrame(df['date'].str.split('-',2).tolist(),columns =
['year','month','day'])
        df_1=pd.concat([df,df_new],axis=1)
        df_2 = df_1.groupby(by=['year','month'])['var','skew','kurt','earn'].mean()
        df_2=pd.DataFrame(df_2)
        df_2.to_csv('d:/data_earn_mean_1/'+code+'.csv')

#合并上面的文件
#步骤 3
import os
import pandas as pd
import glob
data_list = glob.glob('d:/data_earn_mean_1/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
        fr = open(i,'rb').read()
        with open('d:/data_earn_mean_1/result_earn_mean_1.csv','ab') as f: #将结果保
存为 result.csv
                f.write(fr)
print('合并完毕！')

#步骤 4,进行删除空值的处理，保存为 result_earn_mean_2
result_earn_mean_1=pd.read_csv('d:/data_earn_mean_1/result_earn_mean_1.csv')
result_earn_mean_1=result_earn_mean_1.dropna()
result_earn_mean_1.to_csv('d:/data_earn_mean_1/result_earn_mean_2.csv')

#步骤 5：手动分成 10 年的数据，命名为
'2011','2012','2013','2014','2015','2016','2017','2018','2019','2020'，输出成为 10 个
CSV
```

```
#步骤 6
import xlrd
import pandas as pd
import os
def excel2csv(excel_file):
    # 打开 excel 文件
    workbook=xlrd.open_workbook(excel_file)
    # 获取所有 sheet 名字
    sheet_names=workbook.sheet_names()
    for worksheet_name in sheet_names:
        # 遍历每个 sheet 并用 Pandas 读取
        data_xls=pd.read_excel(excel_file,worksheet_name,index_col=None)
        # 获取 excel 当前目录
        dir_path=os.path.abspath(os.path.dirname(excel_file))
        # 转换成 csv 并保存到 excel 所在目录下的 csv 文件夹中
        csv_path=dir_path+'\\earn\\'
        if not os.path.exists(csv_path):
            os.mkdir(csv_path)

data_xls.to_csv(csv_path+worksheet_name+'.csv',index=None,encoding='utf-8')
excel2csv(r'd:/data_earn_mean_1/a/aa.xlsx')

# 步骤 7,10 年的数据按月分开，共 120 个 CSV
year={'2011','2012','2013','2014','2015','2016','2017','2018','2019','2020'}
for code in year:
    df=pd.read_csv('d:/data_earn_mean_1/a/earn/'+code+'.csv')
    classinformation=df['month'].unique()
    for temp in classinformation:
        temp_data=df[df['month'].isin([temp])]
        exec("df%s=temp_data"%temp)
    df1.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_1.csv')
    df2.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_2.csv')
    df3.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_3.csv')
    df4.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_4.csv')
    df5.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_5.csv')
    df6.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_6.csv')
    df7.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_7.csv')
    df8.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_8.csv')
    df9.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_9.csv')
    df10.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_10.csv')
    df11.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_11.csv')
    df12.to_csv('d:/data_earn_mean_1/a/earn/b/'+code+'_12.csv')

#步骤 8
```

```
for temp in classinformation:
    temp_data=df[df['month'].isin([temp])]
    exec("df%s=temp_data"%temp)
df1.to_csv('d:/data_earn_mean_1/a/earn/b/2011_1.csv')
df2.to_csv('d:/data_earn_mean_1/a/earn/b/2011_2.csv')
df3.to_csv('d:/data_earn_mean_1/a/earn/b/2011_3.csv')
df4.to_csv('d:/data_earn_mean_1/a/earn/b/2011_4.csv')
df5.to_csv('d:/data_earn_mean_1/a/earn/b/2011_5.csv')
df6.to_csv('d:/data_earn_mean_1/a/earn/b/2011_6.csv')
df7.to_csv('d:/data_earn_mean_1/a/earn/b/2011_7.csv')
df8.to_csv('d:/data_earn_mean_1/a/earn/b/2011_8.csv')
df9.to_csv('d:/data_earn_mean_1/a/earn/b/2011_9.csv')
df10.to_csv('d:/data_earn_mean_1/a/earn/b/2011_10.csv')
df11.to_csv('d:/data_earn_mean_1/a/earn/b/2011_11.csv')
df12.to_csv('d:/data_earn_mean_1/a/earn/b/2011_12.csv')

# 步骤 9,计算 rank_ic 值
sheet={'2011_1','2011_2','2011_3','2011_4','2011_5','2011_6','2011_7','2011_8','2011_
9','2011_10','2011_11','2011_12','2012_1','2012_2','2012_3','2012_4','2012_5','2012_6',
'2012_7','2012_8','2012_9','2012_10','2012_11','2012_12','2013_1','2013_2','2013_3','
2013_4','2013_5','2013_6','2013_7','2013_8','2013_9','2013_10','2013_11','2013_12','2
014_1','2014_2','2014_3','2014_4','2014_5','2014_6','2014_7','2014_8','2014_9','2014_
10','2014_11','2014_12','2015_1','2015_2','2015_3','2015_4','2015_5','2015_6','2015_7
','2015_8','2015_9','2015_10','2015_11','2015_12','2016_1','2016_2','2016_3','2016_4','
2016_5','2016_6','2016_7','2016_8','2016_9','2016_10','2016_11','2016_12','2017_1','2
017_2','2017_3','2017_4','2017_5','2017_6','2017_7','2017_8','2017_9','2017_10','2017
_11','2017_12','2018_1','2018_2','2018_3','2018_4','2018_5','2018_6','2018_7','2018_8
','2018_9','2018_10','2018_11','2018_12','2019_1','2019_2','2019_3','2019_4','2019_5','
2019_6','2019_7','2019_8','2019_9','2019_10','2019_11','2019_12','2020_1','2020_2','2
020_3','2020_4','2020_5','2020_6','2020_7','2020_8','2020_9','2020_10','2020_11','202
0_12'}
for code in sheet:
    df=pd.read_csv('d:/data_earn_mean_1/a/earn/'+code+'.csv')
    df['var_down']=df['var'].shift(1)
    df['skew_down']=df['skew'].shift(1)
    df['kurt_down']=df['kurt'].shift(1)
    df=df.dropna()
    df['rank_earn'] = df['earn'].rank(method="first")
    df['rank_var'] = df['var'].rank(method="first")
    df['rank_skew'] = df['skew'].rank(method="first")
    df['rank_kurt'] = df['kurt'].rank(method="first")
    a=st.spearmanr(df['rank_var'],df['rank_earn'])
    b=st.spearmanr(df['rank_skew'],df['rank_earn'])
    c=st.spearmanr(df['rank_kurt'],df['rank_earn'])
```

```
        d=pd.DataFrame([a,b,c])
        d=d.drop(axis=1,labels=['pvalue'])
        d=d.T
        d.columns=['ic_var','ic_skew','ic_kurt']
        d.to_csv('d:/data_earn_mean_1/ic/'+code+'.csv')

#合并
data_list = glob.glob('d:/data_earn_mean_1/ic/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
        fr = open(i,'rb').read()
        with open('d:/data_earn_mean_1/ic/fin_earn_IC.csv','ab') as f: #将结果保存为
result.csv
                f.write(fr)
print('合并完毕！')

# 步骤 10，计算 IC 值
for code in sheet:
        df=pd.read_csv('d:/data_earn_mean_1/a/earn/'+code+'.csv')
        df['var_down']=df['var'].shift(1)
        df['skew_down']=df['skew'].shift(1)
        df['kurt_down']=df['kurt'].shift(1)
        df=df.dropna()

        a=st.spearmanr(df['var_down'],df['earn'])
        b=st.spearmanr(df['skew_down'],df['earn'])
        c=st.spearmanr(df['kurt_down'],df['earn'])
        d=pd.DataFrame([a,b,c])
        d=d.drop(axis=1,labels=['pvalue'])
        d=d.T
        d.columns=['ic_var','ic_skew','ic_kurt']
        d.to_csv('d:/data_earn_mean_1/a/rankic/'+code+'.csv')
#合并
data_list = glob.glob('d:/data_earn_mean_1/a/rankic/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
        fr = open(i,'rb').read()
        with open('d:/data_earn_mean_1/a/rankic/fin_earn_new.csv','ab') as f: #将结果
保存为 result.csv
                f.write(fr)
print('合并完毕！')
```

```
#收益波动因子

#步骤 1
for code in stocks:
    data=pd.read_csv('D:/data_1min/'+code+'.csv')
    data.columns=['time/1min','open','close','low','high','volume','money']
#计算对数收益率
    data['earn_rate_log']=np.log((data['close']/data['close'].shift(1)))
    earn_rate_log_data=data['earn_rate_log'].dropna()
    data_new = pd.DataFrame(data['time/1min'].str.split(' ',1).tolist(),columns = ['date','time'])#提取原表日期列，分成日期和时间两列
    data=data.drop(axis=1,labels=['time/1min'])
    data_1=pd.concat([data_new,data],axis=1)
    data_1['earn']=data_1['close']-data_1['open']
    data_2=data_1[['date','earn_rate_log','earn']]

#上下行波动
    data_3=data_2[data_2.earn_rate_log>=0]
    data_4=data_2[data_2.earn_rate_log<0]
    a=pd.DataFrame(data_3['earn_rate_log']**2)
    data_3=data_3.drop(axis=1,labels=['earn_rate_log'])
    data_3=pd.concat([data_3,a],axis=1)
    b=pd.DataFrame(data_4['earn_rate_log']**2)
    data_4=data_4.drop(axis=1,labels=['earn_rate_log'])
    data_4=pd.concat([data_4,b],axis=1)
    data_5 = data_3.groupby(by=['date'])['earn_rate_log'].sum()
    data_5=pd.DataFrame(data_5)
    data_5.columns=['up']
    data_6 = data_4.groupby(by=['date'])['earn_rate_log'].sum()
    data_6=pd.DataFrame(data_6)
    data_6.columns=['down']
    data_7=pd.DataFrame(data_5['up']**0.5)
    data_8=pd.DataFrame(data_6['down']**0.5)
    data_9=pd.concat([data_7,data_8],axis=1)#data_9 为上下行波动

#上下行波动占比
    c=pd.DataFrame(data_2['earn_rate_log']**2)
    data_10=data_2.drop(axis=1,labels=['earn_rate_log'])
    data_11=pd.concat([data_10,c],axis=1)
    data_12= data_11.groupby(by=['date'])['earn_rate_log'].sum()
    data_12=pd.DataFrame(data_12)
    data_12.columns=['all**2']
    data_12=data_12.dropna()
    data_13=data_5['up']/data_12['all**2']
```

```
    data_13=pd.DataFrame(data_13)
    data_13.columns=['up ratio']
    data_14=data_6['down']/data_12['all**2']
    data_14=pd.DataFrame(data_14)
    data_14.columns=['down ratio']
    data_15=pd.concat([data_13,data_14],axis=1)#data_15 为上下行波动占比，已经进行 dropna，2421 行

#将上下行波动与上下行波动占比合并
    data_17= data_2.groupby(by=['date'])['earn'].mean()#对数收益率每日的均值
    data_17= pd.DataFrame(data_17)
    data_17=data_17.dropna()
    data_16=pd.concat([data_9,data_15,data_17],axis=1)#data_16 为收益波动因子合集+对数收益率，2421*5
    filename='d:/data_bd/'+code+'.csv'
    data_16.to_csv(filename)
    mean_1=pd.DataFrame(data_16.mean())
    mean_1.columns=[code]
    filename='d:/data_bd_mean/'+code+'.csv'
    mean_1.to_csv(filename)

#步骤 2
for code in stocks:
    df=pd.read_csv('d:/data_bd/'+code+'.csv')
    df_new = pd.DataFrame(df['date'].str.split('-',2).tolist(),columns = ['year','month','day'])
    df_1=pd.concat([df,df_new],axis=1)
    df_2 = df_1.groupby(by=['year','month'])['up','down','up ratio','down ratio','earn'].mean()
    df_2=pd.DataFrame(df_2)
    df_2.to_csv('d:/data_bd_mean/'+code+'.csv')

#步骤 3
import os
import pandas as pd
import glob
data_list = glob.glob('d:/data_bd_mean/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
    fr = open(i,'rb').read()
    with open('d:/data_bd_mean/result_bd_mean_1.csv','ab') as f: #将结果保存为 result.csv
        f.write(fr)
```

```
print('合并完毕！')

#步骤 4,进行删除空值的处理，保存为 result_earn_mean_2
result_earn_mean_1=pd.read_csv('d:/data_bd_mean/result_bd_mean_1.csv')
result_earn_mean_1=result_earn_mean_1.dropna()
result_earn_mean_1.to_csv('d:/data_bd_mean/result_bd_mean_2.csv')

#步骤 5：手动分成 10 年的数据，命名为
'2011','2012','2013','2014','2015','2016','2017','2018','2019','2020'，输出成为 10 个
CSV

#步骤 6
import xlrd
import pandas as pd
import os
def excel2csv(excel_file):
    # 打开 excel 文件
    workbook=xlrd.open_workbook(excel_file)
    # 获取所有 sheet 名字
    sheet_names=workbook.sheet_names()
    for worksheet_name in sheet_names:
        # 遍历每个 sheet 并用 Pandas 读取
        data_xls=pd.read_excel(excel_file,worksheet_name,index_col=None)
        # 获取 excel 当前目录
        dir_path=os.path.abspath(os.path.dirname(excel_file))
        # 转换成 csv 并保存到 excel 所在目录下的 csv 文件夹中
        csv_path=dir_path+'\\bd\\'
        if not os.path.exists(csv_path):
            os.mkdir(csv_path)

data_xls.to_csv(csv_path+worksheet_name+'.csv',index=None,encoding='utf-8')
excel2csv(r'd:/data_bd_mean/aa_bd.xlsx')

# 步骤 7
year={'2011','2012','2013','2014','2015','2016','2017','2018','2019','2020'}
for code in year:
    df=pd.read_csv('d:/data_bd_mean/bd/'+code+'.csv')
    classinformation=df['month'].unique()
    for temp in classinformation:
        temp_data=df[df['month'].isin([temp])]
        exec("df%s=temp_data"%temp)
    df1.to_csv('d:/data_bd_mean/b/'+code+'_1.csv')
    df2.to_csv('d:/data_bd_mean/b/'+code+'_2.csv')
    df3.to_csv('d:/data_bd_mean/b/'+code+'_3.csv')
```

```
    df4.to_csv('d:/data_bd_mean/b/'+code+'_4.csv')
    df5.to_csv('d:/data_bd_mean/b/'+code+'_5.csv')
    df6.to_csv('d:/data_bd_mean/b/'+code+'_6.csv')
    df7.to_csv('d:/data_bd_mean/b/'+code+'_7.csv')
    df8.to_csv('d:/data_bd_mean/b/'+code+'_8.csv')
    df9.to_csv('d:/data_bd_mean/b/'+code+'_9.csv')
    df10.to_csv('d:/data_bd_mean/b/'+code+'_10.csv')
    df11.to_csv('d:/data_bd_mean/b/'+code+'_11.csv')
    df12.to_csv('d:/data_bd_mean/b/'+code+'_12.csv')

# 步骤 8
#计算 rank_ic
sheet={'2011_1','2011_2','2011_3','2011_4','2011_5','2011_6','2011_7','2011_8','2011_
9','2011_10','2011_11','2011_12','2012_1','2012_2','2012_3','2012_4','2012_5','2012_6',
'2012_7','2012_8','2012_9','2012_10','2012_11','2012_12','2013_1','2013_2','2013_3','
2013_4','2013_5','2013_6','2013_7','2013_8','2013_9','2013_10','2013_11','2013_12','2
014_1','2014_2','2014_3','2014_4','2014_5','2014_6','2014_7','2014_8','2014_9','2014_
10','2014_11','2014_12','2015_1','2015_2','2015_3','2015_4','2015_5','2015_6','2015_7
','2015_8','2015_9','2015_10','2015_11','2015_12','2016_1','2016_2','2016_3','2016_4','
2016_5','2016_6','2016_7','2016_8','2016_9','2016_10','2016_11','2016_12','2017_1','2
017_2','2017_3','2017_4','2017_5','2017_6','2017_7','2017_8','2017_9','2017_10','2017
_11','2017_12','2018_1','2018_2','2018_3','2018_4','2018_5','2018_6','2018_7','2018_8
','2018_9','2018_10','2018_11','2018_12','2019_1','2019_2','2019_3','2019_4','2019_5','
2019_6','2019_7','2019_8','2019_9','2019_10','2019_11','2019_12','2020_1','2020_2','2
020_3','2020_4','2020_5','2020_6','2020_7','2020_8','2020_9','2020_10','2020_11','202
0_12'}
for code in sheet:
    df=pd.read_csv('d:/data_bd_mean/b/'+code+'.csv')
    df['up_down']=df['up'].shift(1)
    df['down_down']=df['down'].shift(1)
    df['up ratio_down']=df['up ratio'].shift(1)
    df['down ratio_down']=df['down ratio'].shift(1)
    df=df.dropna()
    df['rank_earn'] = df['earn'].rank(method="first")
    df['rank_up'] = df['up_down'].rank(method="first")
    df['rank_down'] = df['down_down'].rank(method="first")
    df['rank_up ratio'] = df['up ratio_down'].rank(method="first")
    df['rank_down ratio'] = df['down ratio_down'].rank(method="first")
    a=st.spearmanr(df['rank_up'],df['rank_earn'])
    b=st.spearmanr(df['rank_down'],df['rank_earn'])
    c=st.spearmanr(df['rank_up ratio'],df['rank_earn'])
    e=st.spearmanr(df['rank_down ratio'],df['rank_earn'])
    d=pd.DataFrame([a,b,c,e])
    d=d.drop(axis=1,labels=['pvalue'])
```

```
    d=d.T
    d.columns=['ic_up','ic_down','ic_down ratio','ic_down ratio']
    d.to_csv('d:/data_bd_mean/rankic_bd/'+code+'.csv')

#合并
data_list = glob.glob('d:/data_bd_mean/rankic_bd/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
    fr = open(i,'rb').read()
    with open('d:/data_bd_mean/rankic_bd/fin.csv','ab') as f: #将结果保存为
result.csv
        f.write(fr)
print('合并完毕！')

# 步骤 8
#计算 IC 值
for code in sheet:
    df=pd.read_csv('d:/data_bd_mean/b/'+code+'.csv')
    df['up_down']=df['up'].shift(1)
    df['down_down']=df['down'].shift(1)
    df['up ratio_down']=df['up ratio'].shift(1)
    df['down ratio_down']=df['down ratio'].shift(1)
    df=df.dropna()

    a=st.spearmanr(df['up_down'],df['earn'])
    b=st.spearmanr(df['down_down'],df['earn'])
    c=st.spearmanr(df['up ratio_down'],df['earn'])
    e=st.spearmanr(df['down ratio_down'],df['earn'])

    d=pd.DataFrame([a,b,c,e])
    d=d.drop(axis=1,labels=['pvalue'])
    d=d.T
    d.columns=['ic_up','ic_down','ic_down ratio','ic_down ratio']
    d.to_csv('d:/data_bd_mean/ic_bd/'+code+'.csv')

#合并
data_list = glob.glob('d:/data_bd_mean/ic_bd/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
    fr = open(i,'rb').read()
    with open('d:/data_bd_mean/ic_bd/fin_bdic.csv','ab') as f: #将结果保存为
result.csv
```

```
            f.write(fr)
print('合并完毕！')

#成交量分布因子

#步骤 1
for code in stocks:
    data=pd.read_csv('D:/data_1min/'+code+'.csv')
    data.columns=['rank','time/1min','open','close','low','high','volume','money']
    data=data.drop(axis=1,labels=['rank'])
    data_new = pd.DataFrame(data['time/1min'].str.split(' ',1).tolist(),columns = ['date','time'])
#提取原表日期列，分成日期和时间两列
    data=data.drop(axis=1,labels=['time/1min'])
    data_1=pd.concat([data_new,data],axis=1)
    data_1['earn']=data_1['close']-data_1['open']
    data_2=data_1[['date','time','volume','earn']]
    data_3= data_2.groupby(by=['date'])['volume'].sum()
    data_3=pd.DataFrame(data_3)
    data_3.columns=['total volume']
#给数据打标签
    data_new1 = pd.DataFrame(data_2['time'].str.split(':',2).tolist(),columns = ['hour','min','sec'])
    data_new1=data_new1.drop(axis=1,labels=['sec'])
    int1 = pd.DataFrame(data_new1['min'].astype(int))#转化成 int
    data_new1['min_int']=int1
    data_new1['label']='0'
    idx1 = data_new1.query("hour== '09' and 31 <= min_int <= 59 ").index
    data_new1.loc[idx1,'label']=1
    idx2 = data_new1.query("hour== '10' and 0 <= min_int <= 30 ").index
    data_new1.loc[idx2,'label']=2
    idx3 = data_new1.query("hour== '10' and 31 <= min_int <= 59 ").index
    data_new1.loc[idx3,'label']=3
    idx4 = data_new1.query("hour== '11' and 0 <= min_int <= 30 ").index
    data_new1.loc[idx4,'label']=4
    idx5 = data_new1.query("hour== '13' and 0 <= min_int <= 30 ").index
    data_new1.loc[idx5,'label']=5
    idx6 = data_new1.query("hour== '13' and 31 <= min_int <= 59 ").index
    data_new1.loc[idx6,'label']=6
    idx7 = data_new1.query("hour== '14' and 0 <= min_int <= 30 ").index
    data_new1.loc[idx7,'label']=7
    idx8 = data_new1.query("hour== '14' and 31 <= min_int <= 59 ").index
    data_new1.loc[idx8,'label']=8
    idx9 = data_new1.query("hour== '15' and min_int==0 ").index
```

```
        data_new1.loc[idx9,'label']=8
        data_new2=pd.concat([data_2,data_new1],axis=1)

#按照标签将 8 个区间的成交量重新排列
        data_4= data_new2.groupby(by=['date','label'])['volume','earn'].sum()
        data_4=pd.DataFrame(data_4)
        data_4.columns=['group volume','earn']
        pd2=pd.DataFrame()
        for i in range(len(data_3)):
            a=data_3.iloc[i]
            d=pd.DataFrame(a).T
            pd2=pd2.append([d]*8)   #每行复制 8 倍
        pd2=pd.DataFrame(pd2)
#两个 dataframe 需要将索引去掉，并保留作为列
        pd3= pd2.reset_index(drop=False)
        pd4= data_4.reset_index(drop=False)
        data_5=pd.concat([pd4,pd3],axis=1)
#计算每天各时段成交量因子
        data_5['volume ratio']=data_5['group volume']/data_5['total volume']
        data_5=data_5.drop(['total volume','index','group volume'],axis=1)
#data_5 为成交量因子，但还未进行根据标签分类，将每个时段的因子分开
        filename='d:/data_vol/'+code+'.csv'
        data_5.to_csv(filename)

#步骤 2
for code in stocks:
        df=pd.read_csv('d:/data_vol/'+code+'.csv')
        df_new        =        pd.DataFrame(df['date'].str.split('-',2).tolist(),columns        =
['year','month','day'])
        df_1=pd.concat([df,df_new],axis=1)
        df_2 = df_1.groupby(by=['year','month','label'])['volume ratio','earn'].mean()
        df_2=pd.DataFrame(df_2)
        df_2['code']=code
        df_2.to_csv('d:/data_vol_mean/'+code+'.csv')

#步骤 3
import os
import pandas as pd
import glob
data_list = glob.glob('d:/data_vol_mean/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
        fr = open(i,'rb').read()
```

```
    with open('d:/data_vol_mean/vol_1.csv','ab') as f: #将结果保存为 result.csv
        f.write(fr)
print('合并完毕！')

#步骤 4,进行删除空值的处理，保存为 result_earn_mean_2
result_earn_mean_1=pd.read_csv('d:/data_vol_mean/vol_1.csv')
result_earn_mean_1=result_earn_mean_1.dropna()
result_earn_mean_1.to_csv('d:/data_vol_mean/vol_2.csv')

#贴标签分成 10 年的数据
data_new1=pd.read_csv('d:/data_vol_mean/vol_2.csv' ,low_memory=False)
data_new1['lab']='0'
idx1 = data_new1.query("year== '2011' ").index
data_new1.loc[idx1,'lab']=1
idx2 = data_new1.query("year== '2012' ").index
data_new1.loc[idx2,'lab']=2
idx3 = data_new1.query("year== '2013' ").index
data_new1.loc[idx3,'lab']=3
idx4 = data_new1.query("year== '2014' ").index
data_new1.loc[idx4,'lab']=4
idx5 = data_new1.query("year== '2015' ").index
data_new1.loc[idx5,'lab']=5
idx6 = data_new1.query("year== '2016' ").index
data_new1.loc[idx6,'lab']=6
idx7 = data_new1.query("year== '2017' ").index
data_new1.loc[idx7,'lab']=7
idx8 = data_new1.query("year== '2018' ").index
data_new1.loc[idx8,'lab']=8
idx9 = data_new1.query("year== '2019' ").index
data_new1.loc[idx9,'lab']=9
idx10= data_new1.query("year== '2020' ").index
data_new1.loc[idx10,'lab']=10
data_new1.to_csv('d:/data_vol_mean/vol_3.csv')
df=pd.read_csv('d:/data_vol_mean/vol_3.csv', low_memory=False)
classinformation=df['lab'].unique()
for temp in classinformation:
    temp_data=df[df['lab'].isin([temp])]
    exec("df%s=temp_data"%temp)
df1.to_csv('d:/data_vol_mean/a/2011.csv')
df2.to_csv('d:/data_vol_mean/a/2012.csv')
df3.to_csv('d:/data_vol_mean/a/2013.csv')
df4.to_csv('d:/data_vol_mean/a/2014.csv')
df5.to_csv('d:/data_vol_mean/a/2015.csv')
df6.to_csv('d:/data_vol_mean/a/2016.csv')
```

```
df7.to_csv('d:/data_vol_mean/a/2017.csv')
df8.to_csv('d:/data_vol_mean/a/2018.csv')
df9.to_csv('d:/data_vol_mean/a/2019.csv')
df10.to_csv('d:/data_vol_mean/a/2020.csv')

# 步骤 7
year={'2011','2012','2013','2014','2015','2016','2017','2018','2019','2020'}
for code in year:
    df=pd.read_csv('d:/data_vol_mean/a/'+code+'.csv')
    classinformation=df['label'].unique()
    for temp in classinformation:
        temp_data=df[df['label'].isin([temp])]
        exec("df%s=temp_data"%temp)
    df1.to_csv('d:/data_vol_mean/b/'+code+'_1.csv')
    df2.to_csv('d:/data_vol_mean/b/'+code+'_2.csv')
    df3.to_csv('d:/data_vol_mean/b/'+code+'_3.csv')
    df4.to_csv('d:/data_vol_mean/b/'+code+'_4.csv')
    df5.to_csv('d:/data_vol_mean/b/'+code+'_5.csv')
    df6.to_csv('d:/data_vol_mean/b/'+code+'_6.csv')
    df7.to_csv('d:/data_vol_mean/b/'+code+'_7.csv')
    df8.to_csv('d:/data_vol_mean/b/'+code+'_8.csv')
month={'2011_1','2011_2','2011_3','2011_4','2011_5','2011_6','2011_7','2011_8',
       '2012_1','2012_2','2012_3','2012_4','2012_5','2012_6','2012_7','2012_8',
       '2013_1','2013_2','2013_3','2013_4','2013_5','2013_6','2013_7','2013_8',
       '2014_1','2014_2','2014_3','2014_4','2014_5','2014_6','2014_7','2014_8',
       '2015_1','2015_2','2015_3','2015_4','2015_5','2015_6','2015_7','2015_8',
       '2016_1','2016_2','2016_3','2016_4','2016_5','2016_6','2016_7','2016_8',
       '2017_1','2017_2','2017_3','2017_4','2017_5','2017_6','2017_7','2017_8',
       '2018_1','2018_2','2018_3','2018_4','2018_5','2018_6','2018_7','2018_8',
       '2019_1','2019_2','2019_3','2019_4','2019_5','2019_6','2019_7','2019_8',
       '2020_1','2020_2','2020_3','2020_4','2020_5','2020_6','2020_7','2020_8',}
for code in month:
    df=pd.read_csv('d:/data_vol_mean/b/'+code+'.csv')
    classinformation=df['month'].unique()
    for temp in classinformation:
        temp_data=df[df['month'].isin([temp])]
        exec("df%s=temp_data"%temp)
    df1.to_csv('d:/data_vol_mean/c/'+code+'_1.csv')
    df2.to_csv('d:/data_vol_mean/c/'+code+'_2.csv')
    df3.to_csv('d:/data_vol_mean/c/'+code+'_3.csv')
    df4.to_csv('d:/data_vol_mean/c/'+code+'_4.csv')
    df5.to_csv('d:/data_vol_mean/c/'+code+'_5.csv')
    df6.to_csv('d:/data_vol_mean/c/'+code+'_6.csv')
    df7.to_csv('d:/data_vol_mean/c/'+code+'_7.csv')
```

```
    df8.to_csv('d:/data_vol_mean/c/'+code+'_8.csv')
    df9.to_csv('d:/data_vol_mean/c/'+code+'_9.csv')
    df10.to_csv('d:/data_vol_mean/c/'+code+'_10.csv')
    df11.to_csv('d:/data_vol_mean/c/'+code+'_11.csv')
    df12.to_csv('d:/data_vol_mean/c/'+code+'_12.csv')

# 步骤 8
#计算 rank_ic
sheet={'2011_1_1','2011_2_1','2011_3_1','2011_4_1','2011_5_1','2011_6_1','2011_7_
1','2011_8_1','2011_1_2','2011_2_2','2011_3_2','2011_4_2','2011_5_2','2011_6_2','201
1_7_2','2011_8_2','2011_1_3','2011_2_3','2011_3_3','2011_4_3','2011_5_3','2011_6_3
','2011_7_3','2011_8_3','2011_1_4','2011_2_4','2011_3_4','2011_4_4','2011_5_4','2011
_6_4','2011_7_4','2011_8_4','2011_1_5','2011_2_5','2011_3_5','2011_4_5','2011_5_5',
'2011_6_5','2011_7_5','2011_8_5','2011_1_6','2011_2_6','2011_3_6','2011_4_6','2011
_5_6','2011_6_6','2011_7_6','2011_8_6','2011_1_7','2011_2_7','2011_3_7','2011_4_7',
'2011_5_7','2011_6_7','2011_7_7','2011_8_7','2011_1_8','2011_2_8','2011_3_8','2011
_4_8','2011_5_8','2011_6_8','2011_7_8','2011_8_8','2011_1_9','2011_2_9','2011_3_9',
'2011_4_9','2011_5_9','2011_6_9','2011_7_9','2011_8_9','2011_1_10','2011_2_10','20
11_3_10','2011_4_10','2011_5_10','2011_6_10','2011_7_10','2011_8_10','2011_1_11','
2011_2_11','2011_3_11','2011_4_11','2011_5_11','2011_6_11','2011_7_11','2011_8_1
1','2011_1_12','2011_2_12','2011_3_12','2011_4_12','2011_5_12','2011_6_12','2011_7
_12','2011_8_12','2012_1_1','2012_2_1','2012_3_1','2012_4_1','2012_5_1','2012_6_1',
'2012_7_1','2012_8_1','2012_1_2','2012_2_2','2012_3_2','2012_4_2','2012_5_2','2012
_6_2','2012_7_2','2012_8_2','2012_1_3','2012_2_3','2012_3_3','2012_4_3','2012_5_3',
'2012_6_3','2012_7_3','2012_8_3','2012_1_4','2012_2_4','2012_3_4','2012_4_4','2012
_5_4','2012_6_4','2012_7_4','2012_8_4','2012_1_5','2012_2_5','2012_3_5','2012_4_5',
'2012_5_5','2012_6_5','2012_7_5','2012_8_5','2012_1_6','2012_2_6','2012_3_6','2012
_4_6','2012_5_6','2012_6_6','2012_7_6','2012_8_6','2012_1_7','2012_2_7','2012_3_7',
'2012_4_7','2012_5_7','2012_6_7','2012_7_7','2012_8_7','2012_1_8','2012_2_8','2012
_3_8','2012_4_8','2012_5_8','2012_6_8','2012_7_8','2012_8_8','2012_1_9','2012_2_9',
'2012_3_9','2012_4_9','2012_5_9','2012_6_9','2012_7_9','2012_8_9','2012_1_10','201
2_2_10','2012_3_10','2012_4_10','2012_5_10','2012_6_10','2012_7_10','2012_8_10','
2012_1_11','2012_2_11','2012_3_11','2012_4_11','2012_5_11','2012_6_11','2012_7_1
1','2012_8_11','2012_1_12','2012_2_12','2012_3_12','2012_4_12','2012_5_12','2012_
6_12','2012_7_12','2012_8_12','2013_1_1','2013_2_1','2013_3_1','2013_4_1','2013_5
_1','2013_6_1','2013_7_1','2013_8_1','2013_1_2','2013_2_2','2013_3_2','2013_4_2','2
013_5_2','2013_6_2','2013_7_2','2013_8_2','2013_1_3','2013_2_3','2013_3_3','2013_
4_3','2013_5_3','2013_6_3','2013_7_3','2013_8_3','2013_1_4','2013_2_4','2013_3_4','
2013_4_4','2013_5_4','2013_6_4','2013_7_4','2013_8_4','2013_1_5','2013_2_5','2013
_3_5','2013_4_5','2013_5_5','2013_6_5','2013_7_5','2013_8_5','2013_1_6','2013_2_6',
'2013_3_6','2013_4_6','2013_5_6','2013_6_6','2013_7_6','2013_8_6','2013_1_7','2013
_2_7','2013_3_7','2013_4_7','2013_5_7','2013_6_7','2013_7_7','2013_8_7','2013_1_8',
'2013_2_8','2013_3_8','2013_4_8','2013_5_8','2013_6_8','2013_7_8','2013_8_8','2013
_1_9','2013_2_9','2013_3_9','2013_4_9','2013_5_9','2013_6_9','2013_7_9','2013_8_9',
```

'2013_1_10','2013_2_10','2013_3_10','2013_4_10','2013_5_10','2013_6_10','2013_7_10','2013_8_10','2013_1_11','2013_2_11','2013_3_11','2013_4_11','2013_5_11','2013_6_11','2013_7_11','2013_8_11','2013_1_12','2013_2_12','2013_3_12','2013_4_12','2013_5_12','2013_6_12','2013_7_12','2013_8_12','2014_1_1','2014_2_1','2014_3_1','2014_4_1','2014_5_1','2014_6_1','2014_7_1','2014_8_1','2014_1_2','2014_2_2','2014_3_2','2014_4_2','2014_5_2','2014_6_2','2014_7_2','2014_8_2','2014_1_3','2014_2_3','2014_3_3','2014_4_3','2014_5_3','2014_6_3','2014_7_3','2014_8_3','2014_1_4','2014_2_4','2014_3_4','2014_4_4','2014_5_4','2014_6_4','2014_7_4','2014_8_4','2014_1_5','2014_2_5','2014_3_5','2014_4_5','2014_5_5','2014_6_5','2014_7_5','2014_8_5','2014_1_6','2014_2_6','2014_3_6','2014_4_6','2014_5_6','2014_6_6','2014_7_6','2014_8_6','2014_1_7','2014_2_7','2014_3_7','2014_4_7','2014_5_7','2014_6_7','2014_7_7','2014_8_7','2014_1_8','2014_2_8','2014_3_8','2014_4_8','2014_5_8','2014_6_8','2014_7_8','2014_8_8','2014_1_9','2014_2_9','2014_3_9','2014_4_9','2014_5_9','2014_6_9','2014_7_9','2014_8_9','2014_1_10','2014_2_10','2014_3_10','2014_4_10','2014_5_10','2014_6_10','2014_7_10','2014_8_10','2014_1_11','2014_2_11','2014_3_11','2014_4_11','2014_5_11','2014_6_11','2014_7_11','2014_8_11','2014_1_12','2014_2_12','2014_3_12','2014_4_12','2014_5_12','2014_6_12','2014_7_12','2014_8_12','2015_1_1','2015_2_1','2015_3_1','2015_4_1','2015_5_1','2015_6_1','2015_7_1','2015_8_1','2015_1_2','2015_2_2','2015_3_2','2015_4_2','2015_5_2','2015_6_2','2015_7_2','2015_8_2','2015_1_3','2015_2_3','2015_3_3','2015_4_3','2015_5_3','2015_6_3','2015_7_3','2015_8_3','2015_1_4','2015_2_4','2015_3_4','2015_4_4','2015_5_4','2015_6_4','2015_7_4','2015_8_4','2015_1_5','2015_2_5','2015_3_5','2015_4_5','2015_5_5','2015_6_5','2015_7_5','2015_8_5','2015_1_6','2015_2_6','2015_3_6','2015_4_6','2015_5_6','2015_6_6','2015_7_6','2015_8_6','2015_1_7','2015_2_7','2015_3_7','2015_4_7','2015_5_7','2015_6_7','2015_7_7','2015_8_7','2015_1_8','2015_2_8','2015_3_8','2015_4_8','2015_5_8','2015_6_8','2015_7_8','2015_8_8','2015_1_9','2015_2_9','2015_3_9','2015_4_9','2015_5_9','2015_6_9','2015_7_9','2015_8_9','2015_1_10','2015_2_10','2015_3_10','2015_4_10','2015_5_10','2015_6_10','2015_7_10','2015_8_10','2015_1_11','2015_2_11','2015_3_11','2015_4_11','2015_5_11','2015_6_11','2015_7_11','2015_8_11','2015_1_12','2015_2_12','2015_3_12','2015_4_12','2015_5_12','2015_6_12','2015_7_12','2015_8_12','2016_1_1','2016_2_1','2016_3_1','2016_4_1','2016_5_1','2016_6_1','2016_7_1','2016_8_1','2016_1_2','2016_2_2','2016_3_2','2016_4_2','2016_5_2','2016_6_2','2016_7_2','2016_8_2','2016_1_3','2016_2_3','2016_3_3','2016_4_3','2016_5_3','2016_6_3','2016_7_3','2016_8_3','2016_1_4','2016_2_4','2016_3_4','2016_4_4','2016_5_4','2016_6_4','2016_7_4','2016_8_4','2016_1_5','2016_2_5','2016_3_5','2016_4_5','2016_5_5','2016_6_5','2016_7_5','2016_8_5','2016_1_6','2016_2_6','2016_3_6','2016_4_6','2016_5_6','2016_6_6','2016_7_6','2016_8_6','2016_1_7','2016_2_7','2016_3_7','2016_4_7','2016_5_7','2016_6_7','2016_7_7','2016_8_7','2016_1_8','2016_2_8','2016_3_8','2016_4_8','2016_5_8','2016_6_8','2016_7_8','2016_8_8','2016_1_9','2016_2_9','2016_3_9','2016_4_9','2016_5_9','2016_6_9','2016_7_9','2016_8_9','2016_1_10','2016_2_10','2016_3_10','2016_4_10','2016_5_10','2016_6_10','2016_7_10','2016_8_10','2016_1_11','2016_2_11','2016_3_11','2016_4_11','2016_5_11','2016_6_11','2016_7_11','2016_8_11','2016_1_12','2016_2_12','2016_3_12','2016_4_12','2016_5_12','2016_6_12','2016_7_12','2016_8_12','2017_1_1','2017_2_1','2017_3_1','2017_4_1','2017_5_1','2017_6_1','2

017_7_1','2017_8_1','2017_1_2','2017_2_2','2017_3_2','2017_4_2','2017_5_2','2017_
6_2','2017_7_2','2017_8_2','2017_1_3','2017_2_3','2017_3_3','2017_4_3','2017_5_3','
2017_6_3','2017_7_3','2017_8_3','2017_1_4','2017_2_4','2017_3_4','2017_4_4','2017
_5_4','2017_6_4','2017_7_4','2017_8_4','2017_1_5','2017_2_5','2017_3_5','2017_4_5',
'2017_5_5','2017_6_5','2017_7_5','2017_8_5','2017_1_6','2017_2_6','2017_3_6','2017
_4_6','2017_5_6','2017_6_6','2017_7_6','2017_8_6','2017_1_7','2017_2_7','2017_3_7',
'2017_4_7','2017_5_7','2017_6_7','2017_7_7','2017_8_7','2017_1_8','2017_2_8','2017
_3_8','2017_4_8','2017_5_8','2017_6_8','2017_7_8','2017_8_8','2017_1_9','2017_2_9',
'2017_3_9','2017_4_9','2017_5_9','2017_6_9','2017_7_9','2017_8_9','2017_1_10','201
7_2_10','2017_3_10','2017_4_10','2017_5_10','2017_6_10','2017_7_10','2017_8_10','
2017_1_11','2017_2_11','2017_3_11','2017_4_11','2017_5_11','2017_6_11','2017_7_1
1','2017_8_11','2017_1_12','2017_2_12','2017_3_12','2017_4_12','2017_5_12','2017_
6_12','2017_7_12','2017_8_12','2018_1_1','2018_2_1','2018_3_1','2018_4_1','2018_5
_1','2018_6_1','2018_7_1','2018_8_1','2018_1_2','2018_2_2','2018_3_2','2018_4_2','2
018_5_2','2018_6_2','2018_7_2','2018_8_2','2018_1_3','2018_2_3','2018_3_3','2018_
4_3','2018_5_3','2018_6_3','2018_7_3','2018_8_3','2018_1_4','2018_2_4','2018_3_4','
2018_4_4','2018_5_4','2018_6_4','2018_7_4','2018_8_4','2018_1_5','2018_2_5','2018
_3_5','2018_4_5','2018_5_5','2018_6_5','2018_7_5','2018_8_5','2018_1_6','2018_2_6',
'2018_3_6','2018_4_6','2018_5_6','2018_6_6','2018_7_6','2018_8_6','2018_1_7','2018
_2_7','2018_3_7','2018_4_7','2018_5_7','2018_6_7','2018_7_7','2018_8_7','2018_1_8',
'2018_2_8','2018_3_8','2018_4_8','2018_5_8','2018_6_8','2018_7_8','2018_8_8','2018
_1_9','2018_2_9','2018_3_9','2018_4_9','2018_5_9','2018_6_9','2018_7_9','2018_8_9',
'2018_1_10','2018_2_10','2018_3_10','2018_4_10','2018_5_10','2018_6_10','2018_7_
10','2018_8_10','2018_1_11','2018_2_11','2018_3_11','2018_4_11','2018_5_11','2018_
6_11','2018_7_11','2018_8_11','2018_1_12','2018_2_12','2018_3_12','2018_4_12','20
18_5_12','2018_6_12','2018_7_12','2018_8_12','2019_1_1','2019_2_1','2019_3_1','20
19_4_1','2019_5_1','2019_6_1','2019_7_1','2019_8_1','2019_1_2','2019_2_2','2019_3
_2','2019_4_2','2019_5_2','2019_6_2','2019_7_2','2019_8_2','2019_1_3','2019_2_3','2
019_3_3','2019_4_3','2019_5_3','2019_6_3','2019_7_3','2019_8_3','2019_1_4','2019_
2_4','2019_3_4','2019_4_4','2019_5_4','2019_6_4','2019_7_4','2019_8_4','2019_1_5','
2019_2_5','2019_3_5','2019_4_5','2019_5_5','2019_6_5','2019_7_5','2019_8_5','2019
_1_6','2019_2_6','2019_3_6','2019_4_6','2019_5_6','2019_6_6','2019_7_6','2019_8_6',
'2019_1_7','2019_2_7','2019_3_7','2019_4_7','2019_5_7','2019_6_7','2019_7_7','2019
_8_7','2019_1_8','2019_2_8','2019_3_8','2019_4_8','2019_5_8','2019_6_8','2019_7_8',
'2019_8_8','2019_1_9','2019_2_9','2019_3_9','2019_4_9','2019_5_9','2019_6_9','2019
_7_9','2019_8_9','2019_1_10','2019_2_10','2019_3_10','2019_4_10','2019_5_10','201
9_6_10','2019_7_10','2019_8_10','2019_1_11','2019_2_11','2019_3_11','2019_4_11','
2019_5_11','2019_6_11','2019_7_11','2019_8_11','2019_1_12','2019_2_12','2019_3_1
2','2019_4_12','2019_5_12','2019_6_12','2019_7_12','2019_8_12','2020_1_1','2020_2
_1','2020_3_1','2020_4_1','2020_5_1','2020_6_1','2020_7_1','2020_8_1','2020_1_2','2
020_2_2','2020_3_2','2020_4_2','2020_5_2','2020_6_2','2020_7_2','2020_8_2','2020_
1_3','2020_2_3','2020_3_3','2020_4_3','2020_5_3','2020_6_3','2020_7_3','2020_8_3','
2020_1_4','2020_2_4','2020_3_4','2020_4_4','2020_5_4','2020_6_4','2020_7_4','2020
_8_4','2020_1_5','2020_2_5','2020_3_5','2020_4_5','2020_5_5','2020_6_5','2020_7_5',

```
'2020_8_5','2020_1_6','2020_2_6','2020_3_6','2020_4_6','2020_5_6','2020_6_6','2020
_7_6','2020_8_6','2020_1_7','2020_2_7','2020_3_7','2020_4_7','2020_5_7','2020_6_7',
'2020_7_7','2020_8_7','2020_1_8','2020_2_8','2020_3_8','2020_4_8','2020_5_8','2020
_6_8','2020_7_8','2020_8_8','2020_1_9','2020_2_9','2020_3_9','2020_4_9','2020_5_9',
'2020_6_9','2020_7_9','2020_8_9','2020_1_10','2020_2_10','2020_3_10','2020_4_10','
2020_5_10','2020_6_10','2020_7_10','2020_8_10','2020_1_11','2020_2_11','2020_3_1
1','2020_4_11','2020_5_11','2020_6_11','2020_7_11','2020_8_11','2020_1_12','2020_2
_12','2020_3_12','2020_4_12','2020_5_12','2020_6_12','2020_7_12','2020_8_12'}

for code in sheet:
    df=pd.read_csv('d:/data_vol_mean/c/'+code+'.csv')
    df=df.dropna()
    df['rank_earn'] = df['earn'].rank(method="first")
    df['rank_volume ratio'] = df['volume ratio'].rank(method="first")
    a=st.spearmanr(df['rank_volume ratio'],df['rank_earn'])
    d=pd.DataFrame([a])
    d=d.drop(axis=1,labels=['pvalue'])
    d=d.T
    d.columns=['ic_volume ratio']
    d.to_csv('d:/data_vol_mean/rankic_vol/'+code+'.csv')

#合并
data_list = glob.glob('d:/data_vol_mean/rankic_vol/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
    fr = open(i,'rb').read()
    with open('d:/data_vol_mean/rankic_vol/fin_vol.csv','ab') as f: #将结果保存为
result.csv
        f.write(fr)
print('合并完毕！')
sheet={'vol_1','vol_2','vol_3','vol_4','vol_5','vol_6','vol_7','vol_8',}
#合并
for code in sheet:
    data_list = glob.glob('d:/data_vol_mean/rankic_vol/'+sheet+'/*.csv')
    print(u'共发现%s 个 CSV 文件'% len(data_list))
    print(u'正在处理.....')
    for i in data_list:
        fr = open(i,'rb').read()
        with open('d:/data_vol_mean/rankic_vol/fin_'+sheet+'.csv','ab') as f: #将结
果保存为 result.csv
            f.write(fr)
    print('合并完毕！')
data_list = glob.glob('d:/data_vol_mean/ic_vol/vol_8/*.csv')
```

```
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
    fr = open(i,'rb').read()
    with open('d:/data_vol_mean/ic_vol/fin_volic_8.csv','ab') as f: #将结果保存为
result.csv
        f.write(fr)
print('合并完毕！')

#步骤 9
#计算 IC 值
for code in sheet:
    df=pd.read_csv('d:/data_vol_mean/c/'+code+'.csv')
    df=df.dropna()
    a=st.spearmanr(df['volume ratio'],df['earn'])
    d=pd.DataFrame([a])
    d=d.drop(axis=1,labels=['pvalue'])
    d=d.T
    d.columns=['ic_volume ratio']
    d.to_csv('d:/data_vol_mean/ic_vol/'+code+'.csv')

#合并
data_list = glob.glob('d:/data_vol_mean/ic_vol/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
    fr = open(i,'rb').read()
    with open('d:/data_vol_mean/ic_vol/fin_vol_IC.csv','ab') as f: #将结果保存为
result.csv
        f.write(fr)
print('合并完毕！')

#量价相关因子
#步骤 1
for code in stocks:
    data=pd.read_csv('D:/data_1min/'+code+'.csv')
    data.columns=['rank','time/1min','open','close','low','high','volume','money']
    data=data.drop(axis=1,labels=['rank'])

#data.columns=['time/1min','open','close','low','high','volume','money','factor','high_li
mit','low_limit','avg','pre_close']
    data_new = pd.DataFrame(data['time/1min'].str.split(' ',1).tolist(),columns =
['date','time'])#提取原表日期列，分成日期和时间两列
    data=data.drop(axis=1,labels=['time/1min'])
```

```
    data_1=pd.concat([data_new,data],axis=1)
    data_1['earn']=data_1['close']-data_1['open']
    data_3=data_1.groupby(by=['date'])['earn'].mean()
    data_3=pd.DataFrame(data_3)
    data_2=data_1.groupby(by=['date'])['close'].corr(data_1['volume'])
    data_2=pd.DataFrame(data_2)
    data_2.columns=['corr']#data_2 为高频量价相关系数
    data_4=pd.concat([data_2,data_3],axis=1)
    filename='d:/data_corr/'+code+'.csv'
    data_4.to_csv(filename)

#步骤 2
for code in stocks:
    df=pd.read_csv('d:/data_corr/'+code+'.csv')
    df_new = pd.DataFrame(df['date'].str.split('-',2).tolist(),columns =
['year','month','day'])
    df_1=pd.concat([df,df_new],axis=1)
    df_2 = df_1.groupby(by=['year','month'])['corr','earn'].mean()
    df_2=pd.DataFrame(df_2)
    df_2.to_csv('d:/data_corr_mean/'+code+'.csv')

#步骤 3
import os
import pandas as pd
import glob
data_list = glob.glob('d:/data_corr_mean/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
    fr = open(i,'rb').read()
    with open('d:/data_corr_mean/result_corr_mean_1.csv','ab') as f: #将结果保存为
result.csv
        f.write(fr)
print('合并完毕！')

#步骤 4,进行删除空值的处理，保存为 result_earn_mean_2
result_earn_mean_1=pd.read_csv('d:/data_corr_mean/result_corr_mean_1.csv')
result_earn_mean_1=result_earn_mean_1.dropna()
result_earn_mean_1.to_csv('d:/data_corr_mean/result_corr_mean_2.csv')

#步骤 5：手动分成 10 年的数据，命名为
'2011','2012','2013','2014','2015','2016','2017','2018','2019','2020'，输出成为 10 个
CSV
```

```
#步骤 6
import xlrd
import pandas as pd
import os
def excel2csv(excel_file):
    # 打开 excel 文件
    workbook=xlrd.open_workbook(excel_file)
    # 获取所有 sheet 名字
    sheet_names=workbook.sheet_names()
    for worksheet_name in sheet_names:
        # 遍历每个 sheet 并用 Pandas 读取
        data_xls=pd.read_excel(excel_file,worksheet_name,index_col=None)
        # 获取 excel 当前目录
        dir_path=os.path.abspath(os.path.dirname(excel_file))
        # 转换成 csv 并保存到 excel 所在目录下的 csv 文件夹中
        csv_path=dir_path+'\\corr\\'
        if not os.path.exists(csv_path):
            os.mkdir(csv_path)

data_xls.to_csv(csv_path+worksheet_name+'.csv',index=None,encoding='utf-8')
excel2csv(r'd:/data_corr_mean/aa_corr.xlsx')

# 步骤 7
#分成 120 个文件
year={'2011','2012','2013','2014','2015','2016','2017','2018','2019','2020'}
for code in year:
    df=pd.read_csv('d:/data_corr_mean/corr/'+code+'.csv')
    classinformation=df['month'].unique()
    for temp in classinformation:
        temp_data=df[df['month'].isin([temp])]
        exec("df%s=temp_data"%temp)
    df1.to_csv('d:/data_corr_mean/b/'+code+'_1.csv')
    df2.to_csv('d:/data_corr_mean/b/'+code+'_2.csv')
    df3.to_csv('d:/data_corr_mean/b/'+code+'_3.csv')
    df4.to_csv('d:/data_corr_mean/b/'+code+'_4.csv')
    df5.to_csv('d:/data_corr_mean/b/'+code+'_5.csv')
    df6.to_csv('d:/data_corr_mean/b/'+code+'_6.csv')
    df7.to_csv('d:/data_corr_mean/b/'+code+'_7.csv')
    df8.to_csv('d:/data_corr_mean/b/'+code+'_8.csv')
    df9.to_csv('d:/data_corr_mean/b/'+code+'_9.csv')
    df10.to_csv('d:/data_corr_mean/b/'+code+'_10.csv')
    df11.to_csv('d:/data_corr_mean/b/'+code+'_11.csv')
    df12.to_csv('d:/data_corr_mean/b/'+code+'_12.csv')
```

```
#步骤 8
#计算 rank_ic 值
sheet={'2011_1','2011_2','2011_3','2011_4','2011_5','2011_6','2011_7','2011_8','2011_
9','2011_10','2011_11','2011_12','2012_1','2012_2','2012_3','2012_4','2012_5','2012_6',
'2012_7','2012_8','2012_9','2012_10','2012_11','2012_12','2013_1','2013_2','2013_3','
2013_4','2013_5','2013_6','2013_7','2013_8','2013_9','2013_10','2013_11','2013_12','2
014_1','2014_2','2014_3','2014_4','2014_5','2014_6','2014_7','2014_8','2014_9','2014_
10','2014_11','2014_12','2015_1','2015_2','2015_3','2015_4','2015_5','2015_6','2015_7
','2015_8','2015_9','2015_10','2015_11','2015_12','2016_1','2016_2','2016_3','2016_4','
2016_5','2016_6','2016_7','2016_8','2016_9','2016_10','2016_11','2016_12','2017_1','2
017_2','2017_3','2017_4','2017_5','2017_6','2017_7','2017_8','2017_9','2017_10','2017
_11','2017_12','2018_1','2018_2','2018_3','2018_4','2018_5','2018_6','2018_7','2018_8
','2018_9','2018_10','2018_11','2018_12','2019_1','2019_2','2019_3','2019_4','2019_5','
2019_6','2019_7','2019_8','2019_9','2019_10','2019_11','2019_12','2020_1','2020_2','2
020_3','2020_4','2020_5','2020_6','2020_7','2020_8','2020_9','2020_10','2020_11','202
0_12'}
for code in sheet:
    df=pd.read_csv('d:/data_corr_mean/b/'+code+'.csv')
    df['corr_down']=df['corr'].shift(1)
    df=df.dropna()
    df['rank_earn'] = df['earn'].rank(method="first")
    df['rank_corr'] = df['corr_down'].rank(method="first")
    a=st.spearmanr(df['rank_corr'],df['rank_earn'])
    d=pd.DataFrame([a])
    d=d.drop(axis=1,labels=['pvalue'])
    d=d.T
    d.columns=['ic_corr']
    d.to_csv('d:/data_corr_mean/rankic_corr/'+code+'.csv')

#合并
data_list = glob.glob('d:/data_corr_mean/rankic_corr/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
    fr = open(i,'rb').read()
    with open('d:/data_corr_mean/rankic_corr/fin_corr.csv','ab') as f: #将结果保存
为 result.csv
        f.write(fr)
print('合并完毕！')

#步骤 9
#计算 IC 值
for code in sheet:
    df=pd.read_csv('d:/data_corr_mean/b/'+code+'.csv')
```

```
        df['corr_down']=df['corr'].shift(1)
        df=df.dropna()
        df['rank_earn'] = df['earn'].rank(method="first")
        df['rank_corr'] = df['corr_down'].rank(method="first")
        a=st.spearmanr(df['rank_corr'],df['rank_earn'])
        d=pd.DataFrame([a])
        d=d.drop(axis=1,labels=['pvalue'])
        d=d.T
        d.columns=['ic_corr']
        d.to_csv('d:/data_corr_mean/ic/'+code+'.csv')
#合并
data_list = glob.glob('d:/data_corr_mean/ic/*.csv')
print(u'共发现%s 个 CSV 文件'% len(data_list))
print(u'正在处理.....')
for i in data_list:
        fr = open(i,'rb').read()
        with open('d:/data_corr_mean/ic/fin_corr_ic.csv','ab') as f: #将结果保存为
result.csv
                f.write(fr)
print('合并完毕！')
```

致谢

转眼间两年的研究生生涯就要结束了，这也是我在东华大学度过的第六个年头，在论文撰写完成之际，谨向在东华大学学习生活期间给予我帮助与关怀的亲人、老师、同学以及朋友们表示衷心的感谢和诚挚的祝福。

首先，我要感谢我的父母。从小到大，父母一直努力为我创造良好的生活与学习的环境，一直以来他们都是我最坚强的后盾，无论我做怎样的决定，他们总会用自己的方式尽全力的支持我、保护我。非常感谢父母一直以来的默默支持与鼓励，愿父母身体健康、万事顺意。

其次，我要感谢我的研究生导师胡良剑教授。能够遇到胡老师这样工作态度认真、学术作风严谨的导师让我感到非常幸运。在整个论文撰写的整个过程中，从论文选题、到具体的研究过程以及最后正文的撰写，胡老师都提出了很多宝贵的意见。胡老师指出的每一个问题，指导的每一个思路，都让我有醍醐灌顶之感。同时，无论从论文的格式规范、论文内容的逻辑衔接、还是整篇文章的结构，胡老师都不厌其烦，多次给予我及时的帮助，使我能够最后顺利完成论文写作工作。

最后，我要感谢在东华遇到的同学、朋友们。东华六载，我遇到了很多可爱的同学和朋友，和大家共同度过了愉快的学生时代。感谢陪伴，感谢成长，感谢每一个为了美好未来不断努力前进的我们，愿我们都能够在各自的未来里闪闪发光。