2023 年第四届“大湾区杯”粤港澳金融数学建模竞赛

基于多元策略的跨境 *ETF* 套利策略设计

摘　　要：

本文主要对跨市场交易策略进行设计，提出给予长短期记忆神经网络，网格交易策略和 *Dual Thrust* 策略的创新型模型，以实现市场投资的目标筛选以及超额收益，通过对建立最优化模型和策略的具体回测，检验其实用性和有效性，利用 *Matlab* 和 *Python* 给出跨境 *ETF* 套利策略。

针对问题一，在遵循基金市场规则下，解决一二级市场套利问题，数通过据预处理，进行 *Pearson* **相关性检验**和**最小二乘法**调整，结合使用一系列参数和指标确定套利模型参数，最终确定 *ETF* 的波动率、β 值、折溢率、收益率、夏普指数共 5 个指标，进行**评价模型**建立，合理地对目标函数和数据进行**归一化处理**，并对 *ETF* 基金进行评估和排序，确定前十支最有潜力 *ETF* 基金，满足套利需求，提高了整体资金使用效率，促进一二级市场平衡发展。

针对问题二，在不同的交易制度下，进行策略改进和跨境数据，并合理建立**时间序列**以确认最佳**预测模型**，结合使用代表性观察总体趋势，采用一阶差分或二阶差分处理，将数据转为平稳时间序列。比较不同预**测模型**的各性能指标以选择最佳模型，建立网格交易策略，合理设置止盈止损条件可较好规避风险。根据**预测模型**和**网格交易策略**，实施套利策略，记录套利的效果和绩效。沿用任务一的等级排名制度，并从**投资者角度**出发以提高收益为目标，并综合各个指标得到最适合套利且收益率较高的排名前十的跨境 *ETF* 基金，实现更好的投资策略

针对任务三，由于跨境 *ETF* 与国际股票市场的联动性，进行**相关性分析**，设置**相关性阈值为 0.9**，并从中筛选出相关性大于此阈值的股票列表，同时对列表中股票进行**线性回归拟合**得到相应价格波动趋势，针对各趋势进行**评分指标计算**，结合数据拟合结果的真实率，获得对应的得分情况。根据**大类资产轮动理论**，选取在当前经济周期下最适合的资产类型组合，并结合所得**网格交易策略**，得到最终的总收益。

通过对本题三个任务中各种情况的具体分析，进行多维度问题分析，建立数学模型以实现市场投资的目标筛选以及超额收益，验证所建立的模型的可行性。

关键词： *Dual Thrust* **策略**　**网格交易策略**　*LSTM* **预测**　**大类资产轮动理论**

一、问题重述

1.1 引言

随着金融市场的不断发展，跟踪“标的指数”变化的基金，即交易型开放式指数基金（*Exchange Traded Fund*，称 *ETF*）已逐渐成为投资者和机构的首选，并以其高流动性、低费用和多样性而备受欢迎。跨境 *ETF* 作为一种特殊类型，将国内市场与国际市场联系起来，利用跨境 *ETF* 独特的交易机制，在不同市场间构建多空组合，获取低风险稳健收益的过程，为投资者提供了独特的套利机会。由于不同市场之间存在价格差异，投资者通过同时在不同市场执行交易来获得差价利润。通过价差来获得盈利，投资者在跨境 *ETF* 套利交易中实现低风险稳健收益。

基于复杂而潜在收益可观的跨境 *ETF* 套利投资策略，用于规避市场波动和获取稳定的回报。同时，由于跨境 *ETF* 套利交易需要涉及两个市场之间的交易和转换。因此，投资者具备较高的投资技巧和经验，以及对市场的深刻理解和判断能力。本任务将聚焦在跨境 *ETF* 套利，以实现更好的投资回报，通过将建立套利模型，以筛选出最适合进行套利和跨境的 *ETF* 优化组合，并针对不同市场的交易制度，调整模型以实现 $T+0$ 交易提高交易效率。通过研究跨境 *ETF* 与股指期货的跨市场套利策略，以充分利用国际市场与 *A* 股之间的关联性。结合实际测试和数据分析所生成实测报告，为投资者提供有关跨境 *ETF* 和股指期货套利策略，进而帮助投资者们利用跨境 *ETF* 和股指期货之间的价格差异，更好地了解和利用跨境 *ETF* 套利的潜力，给出实际操作合理可靠且行之有效的组合策略。

1.2 问题提出

结合赛题给出的相关 *ETF* 基金交易数据（2023 年 8 月 1 日-10 月 31 日），讨论 *ETF* 基金的套利策略设计，旨在探索跨境 *ETF* 套利策略的有效性和可行性，运用数学建模方法，研究以下任务：

任务一：分析相关交易数据，建立一二级市场间套利模型，以选择最适合进行套利的前 10 只 *ETF* 。

任务二：通过调整模型实现 $T+0$ 交易，建立跨境 *ETF* 套利模型，并选出前 10 只最适合的跨境 *ETF* 。

任务三：关注跨境 *ETF* 与股指期货的跨市场套利策略，建模并选择最佳组合。

任务四：结合实际测试任务二和任务三中的模型，生成实测报告，提供有关套利操作的反馈、收益率分析、风险评估和建议。

通过这些任务，进而帮助投资者们利用跨境 *ETF* 和股指期货之间的价格差异，更好地了解和利用跨境 *ETF* 套利的潜力，以提高投资决策水平。

二、问题分析

本题主要是探索跨境*ETF*和股指期货套利策略的有效性和可行性。针对任务一，建立套利模型以选择最适合进行套利的前10只*ETF*；针对任务二，将调整模型实现$T+0$交易，建立跨境ETF套利模型，选出前10只最适合的跨境*ETF*；针对任务三，关注跨境*ETF*与股指期货的跨市场套利策略，建立模型并选择最佳组合。结合以上任务，分别作如下分析。

2.1 任务一分析

在任务一中，对应其设计目标是求解一二级市场之间的套利模型，以满足基金市场的规则。根据提供的具体规则，在基金市场内进行套利，以实现在一级市场和二级市场之间的价格差异，通过有效的交易策略来获取中间价差的利润，以下是对任务一的分析：

（1）明确基于基金单位净值和交易价格之间的差异，当差值大于中间手续费时，可以进行套利以获取中间价差，通过股市走势的预测，以减少股票交易风险，并获取短期趋势带来的利润[1]。同时，考虑基金规模对流动性的影响，红利分配的特殊情况，以及分红对套利模型的影响。

（2）数据预处理，筛选有效数据和处理异常数据，以确保模型的准确性，使用日均成交额来筛选投资备选池，同时排除异常数据。从处理后的数据中，确定套利模型参数，任务中涉及了多个参数，如折溢率、*ETF*成分股利率、波动性等。结合参数用于计算套利机会和风险，需要确定它们的值和如何使用它们来支持套利决策。

（3）投资风险指标的考量，不同的投资风险指标[2]：波动性、股票与市场相关性、夏普指数等，这些指标有助于评估套利策略的风险和回报。

（4）通过将各个参数纳入考虑评价模型，合理地对目标函数和数据进行归一化处理，对*ETF*基金进行评级和排序，以便确定前十支*ETF*基金。

总的来说，任务一旨在利用规则和各参数来寻找潜在的一二级市场套利机会，并使用建立的评价模型对不同的*ETF*基金进行排序，以确定最有利可图的前十支*ETF*基金。通过数据分析、模型建立和风险管理的综合应用，以确保套利活动的成功和合规性。

2.2 任务二分析

在任务二中，对应其设计目标是修改套利模型，以适应跨境*ETF*基金，并允许$T+0$交易，即在同一交易日内买入和卖出，根据不同的交易制度，改进任务一的策略，建立相应的跨境套利模型，便于投资者更快速地响应市场变化，提高套利的效率，以下是对任务二的分析：

（1）任务二着眼于建立跨境*ETF*套利模型，基于任务一对跨境数据进行分析和预测建立模型，先进行时间序列数据分析，确定数据是否适合进行时间序列建模[3]。由于时间序列通常包含趋势和季节性，将其转化为平稳时间序列，以确保模型的有效性。通过整理数据，从预测模型的结果中选择效果最好的模型，比较不同预测模型的性能确定套利模型最适合。

（2）为了进一步优化策略，引入网格交易策略被引入[4]。根据每个网格的中心价格，计算买入和卖出的数量，同时考虑市场波动和模型预测的趋势并设置止盈和止损条件，可以更好地管理风险。

（3）实施建立的套利策略并对跨境*ETF*基金进行排名。根据任务一的等级排名制度和任务二的策略绩效，得出对应等级排名前十的跨境*ETF*基金。基于投资者的角度出发，以提高收益为目标，主要侧重收益率数据进行排序，从中选择排名前十的跨境*ETF*基金。

综合来看，任务二通过建立时间序列模型、比较不同预测模型、引入网格交易策略，最终在排名制度中选择符合投资人的套利前十只跨境*ETF*基金，以优化投资策略。

2.3 任务三分析

在任务三中，对应其设计目标是基于股票相关性和价格趋势的分析，建立一个基于跨境*ETF*和股指期货的套利策略模型来实现盈利[5]。

以下是任务三的分析：

（1）基于任务一二中跨境ETF、股指期货和相关股票的历史市场数据，包括开盘价、收盘价等，对跨境*ETF*和股指期货之间的相关性进行分析，以确定它们之间的关联程度。

（2）根据相关性阈值，筛选出与跨境*ETF*关性高于阈值的股票为潜在的投资对象。通过对相关性高的股票执行线性回归分析，以确定价格趋势和波动性。

（3）通过收盘价和开盘价的表现来评估它们的潜在价值并为每只*ETF*计算得分，选择排名前五或最高得分的*ETF*作为投资组合的一部分，计算每只*ETF*在投资组合中的权重[6]。

（4）针对每个交易日，根据前一天的数据和预测价格趋势，决定如何分配资本和进行买卖交易，并计算每日的收益率，并将其累积以获得最终的总收益。

总的来说，任务三通过像相关性分析进行阈值设置，从中筛选出相关性大于此阈值的股票列表，行线性回归拟合得到相应价格波动趋势，通过评分指标计算，获得对应的得分情况。根据大类资产轮动理论，选取在当前经济周期下最适合的资产类型组合，并结合所得网格交易策略，得到最终的总收益。

三、符号说明

符号	描述说明
$close_{idm}$	第i只ETF在第d天的第m分钟的收盘价
C_i	第i支ETF基金的波动性
R_{id}	第i只ETF在第d天的收益率
NAV_{id}	第i只ETF对应第d天的单位净值
R_{it}	第i只ETF中第t个时间段的收益率
W_f、b_f	分别对应tanh函数的权重向量和偏置向量
K_1、K_2	网格交易策略相应的触发参数
$close_d$、$open_d$	第d天的收盘价和开盘价
h_{t-1}	$t-1$时间的输入价格
$P_{Now\,(i-1)}$	当前所在位置下一格的价格
W_i	给予第i个基金或股票的权重大小
$score_i$	第i个基金或股票的得分

四、模型假设

1、考虑ETF折溢价机制可出现折价或溢价的情况。

2、假设市场流动性良好支持套利交易，可执行大规模交易。

3、假设跨境ETF与股指期货市场之间存在关联性，即价格变动会相互影响。

4、假设跨市场交易可行性，投资者可以在不同市场之间无障碍地进行交易，包括买入和卖出跨境ETF以及股指期货。

5、市场具有有效性，即ETF和相关股指的价格反映了市场信息，但存在一定程度的价格差异，可以被利用进行套利。

6、考虑有效市场假说，所有公开消息和全部有价值的讯息，被股票的市场价格充分、及时地反映，市场价格就代表着股票的内在价值。

7、假设初始投资为150万元，期货为100万元，现货为50万元，期货保证金比率为10%，交易费率为0.005%，股票为0.01%，价格滑点为0.01。

五、模型与建立

在满足收益可观的跨境*ETF*套利投资策略的前提下，规避市场波动和获取稳定的回报，以跨境*ETF*套利的有效性和可行性为目标，建立最优化模型，设计对应的算法，帮助投资者们利用跨境*ETF*和股指期货之间的价格差异，更好地了解和利用跨境*ETF*套利的潜力，提高投资决策水平。

5.1 任务一：一二级市场间套利模型

根据基金市场的规则，任务一涉及一二级市场套利，确保套利活动在基金市场内遵循法律法规，并以一定的规范方式进行。

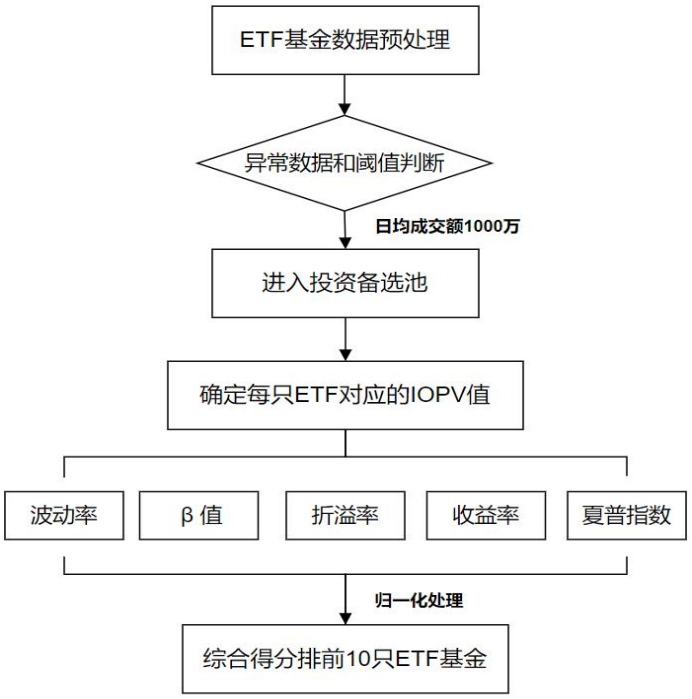

图 1　任务一分析处理流程图

在执行一二级市场套利策略时需要谨慎考虑风险和监管要求，以保障自身权益和市场秩序，具体规则如下：

① 在基金单位净价（*Net Asset Value*，*NAV*）与交易价差值大于中间手续费时，可进行套利赚取中间差价。

② 通过预测股市走势和*ETF*赎回需要的股票（申赎费），将其余不看好的股票抛售（需印花税），以这种方法投资的股票可实现$T+0$交易，减少股票交易风险，可及时止损或者赚取短期趋势带来的利益。

③ 基于基金特殊的交易机制，不论是对基金的申购还是赎回，都涉及到一篮子权重股票的交易。因此当基金的申购与赎回数目很大时，导致权重股票和基金的交易成本和对市场的冲击成本的增加，这会引起加重基金与目标指数之间的差异，使得跟踪误差扩大。

④ 由于 *ETF* 存在利润红利分配，在计算 *IOPV* 时，需注意红利对 *IOPV* 的影响，不能直接进行计算，需对分红后基金波动公式进行特殊变化。

通常来讲，在一二级市场上，价格不会有很大差别。但由于若在一二级市场中有巨大折、溢价的出现 ，套利人势必会根据二级市场的交易机制与申赎机制，完成套利在一二级市场中，这样就造成价格趋于一致的情况在一二级市场上出现。

如上所述交易机制，详见下图：

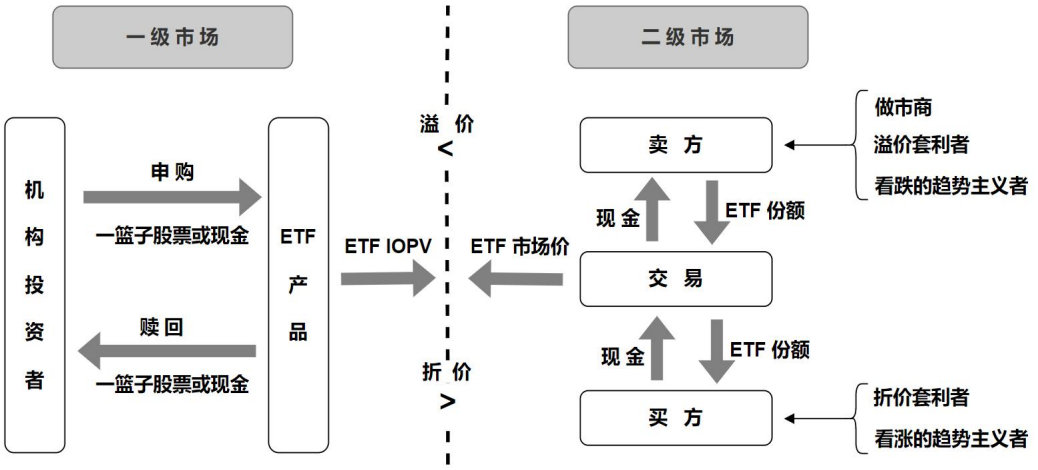

图 2　一二级市场 *ETF* 套利交易机制

5.1.1 数据预处理

ETF 规模越大，流动性也越好，因此越能应对大额资金赎回所带来的冲击。*ETF* 基金的运营会更加平稳，投资者在二级市场上的大金额买卖就不会因流动性不足而受到影响。同时，当基金运作的规模越大时，也会产生一定的规模效应，基金的管理费、托管费等固定费用对其收益率的影响就会越小。

根据一般情况，将日均成交额 1000 万以上的 *ETF* 纳入投资备选池，同时筛选处理部分异常数据。

5.1.2 *ETF* 套利模型参数确定

基于投资的组合理论，一般会查看折溢率、*ETF* 成分股利率、波动性、基金净值收益率、投资风险 β 指标、夏普指数来检验基金险，综合检验是否具有合理的股票的回报，具体分析法如下：

（1）折溢率

ETF 市场价由市场供求决定了其在二级市场的交易价格，基金的 *NAV* （即真实的单位净值）是实际所拥有的股票总价值与基金份额所决定：

$$NAV = \frac{Value}{part}$$

其中，*Value* 为所拥有的股票总价值， *part* 基金份额。

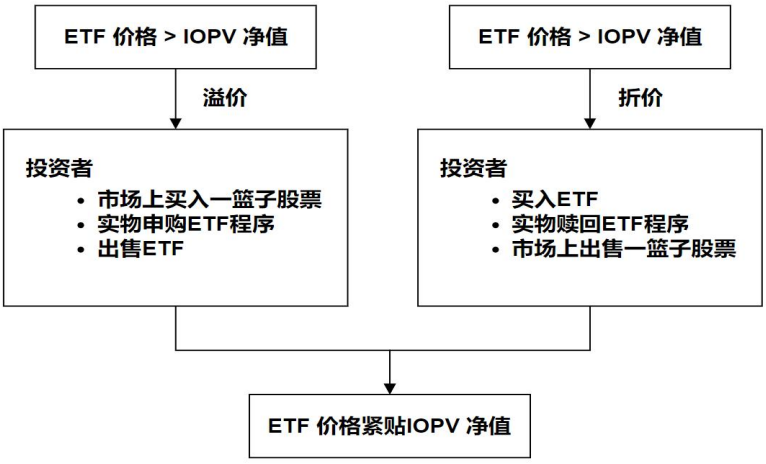

图 3　　*ETF* 折溢价套利流程图

折价，即市场价格低于 NAV 时，即 $P < NAV$ 时，对应其折价率为：

$$ETF_{id}\text{折价率} = \frac{P_{id} - NAV_{id}}{NAV_{id}} \times 100\% \quad (d = 1,2...n, i = 1,2...886)$$

当 $P_d + C_{d1} < NAV_d$ ， C_{d1} 时才能进行套现操作，否则不进行套现。溢价则是当市场价格高于 NAV 时，即 $P > NAV$ 时：

$$ETF_{id}\text{溢价率} = \frac{NAV_{id} - P_{id}}{NAV_{id}} \times 100\% \quad (d = 1,2...n, i = 1,2...886)$$

由于当进行大量套利操作时， NAV 会由于套利操作继而影响市场价格，市场价格会迅速趋近于 NAV 值，很快的回归到 NAV 的水平面上，即套利的机会会迫使 ETF 迅速的靠近 NAV 。

而当 $P_d + C_{d2} > NAV_d$ ， C_{d2} 时才能进行套现操作，否则不进行套现。

最终套利结束的标志为：

$$P_d = NAV_d \quad (d = 1,2...n)$$

故 ETF 二级市场无套利区间为：

$$P \in [NAV_d - C_{d1}, NAV_d - C_{d2}]$$

（2） *ETF* 成分股利率

为计算 ETF 可赎回的股票的利率可先计算股票对应日期内的趋势，表示为：

$$beta_i = \sum_{m=1}^{241} \sum_{d=1}^{n} \left(\frac{close_{id(m+1)}}{close_{idm}} - 1 \right) \quad (i = 1,2...463)$$

由于前十只股票占比基本已经足够计算其对应的净价，其中，$close_{idm}$ 为第 i 只 ETF 在第 d 天的第 m 分钟的收盘价，$close_{id(m+1)}$ 为第 i 只 ETF 第 $m+1$ 分钟与 m 分钟之比可得到分钟内的一个趋势，求平均之后可得到第 i 只 ETF 的总体趋势，正向趋势越好的 $beta_i$ 的值越大。

（3）波动性

波动性是用于衡量价格波动水平的指标，能够反映出基金价格偏离平均值的幅度，波动性越大，意味着价格波动幅度就越大，在一定程度下可以反映 ETF 的风险大小，因此可以根据波动性来调整我们的 ETF 交易策略，从而优化收益。

在金融数学中，波动性通常被定义为收益率的标准差。收益率假定为正态分布，尽管实际分布可能不同。在正态分布中，68%的观测值在一个标准差内，95%的观测值在两个标准差内。波动性对于预测最低价和最高价的范围尤为重要。如果没有重大事件，资产将在其平均波动性范围内移动，因此选定波动性作为评定套利收益的一个重要指标，其具体计算公式如下：

$$C_i = \frac{\sum_{j=1}^{59}\sum_{k=1}^{241}\frac{\max_k - \min_k}{\min_k}}{14219} \qquad (i = 1...886)$$

其中，C_i 表示第 i 支 ETF 基金的波动性，$\max_k$ 表示第 k 分钟交易的最高价，$\min_k$ 表示第 k 分钟交易的最低价。

（4）基金净值收益率

基金的投资价值通常使用基金净值收益率来衡量，这意味着随着基金的净值增长率的增加，基金的投资价值也随之增加。基金净值收益率是一个衡量基金投资绩效的指标，通常用于评估投资的盈利能力。基金净值收益率的计算方式可以根据不同的需求和背景略有不同，但通常可以表示为：

$$R_{id} = \frac{NAV_{id} - NAV_{it-1}}{NAV_{it-1}}$$

R_{id} 为第 i 只 ETF 在第 d 天的收益率，NAV_{id} 为第 i 只 ETF 对应第 d 天的单位净值其中，期望净值是投资期望结束时的净值，初始净值是投资开始时的净值。这个指标可以告诉投资者他们的投资在一定时期内实现了多少回报，以及投资的盈利能力。

每分钟的基金单位净值可通过对收盘进行拟合得到参考 $IOPV$ 估计值，由于日收盘价与单位净价基本相似，并通过单位净价计算式：

$$NAV = \frac{allmoney - deletmoney}{allpart}$$

其中，*allmoney* 为基金总资产，*deletmoney* 基金总负债，*allpart* 基金总份额。由于数据只获取到了每只 *ETF* 前十大持仓股票，故结果值应该满足：

$$IOPV_{估计} \leq NAV$$

如图所示：

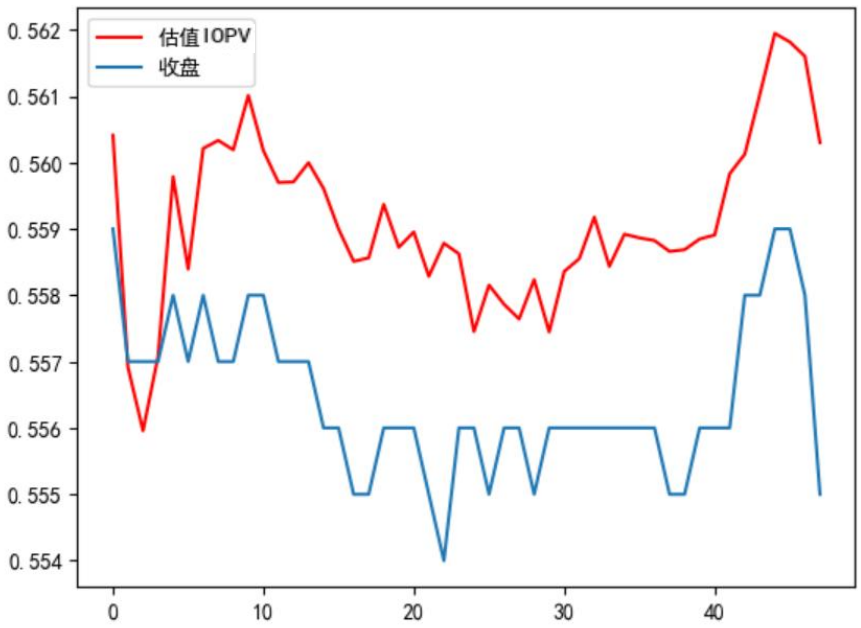

图 4　部分股票 *IOPV* 持仓图

原假设：两数据之间不具有相关性；备择假设：量数据具有相关性。

通过皮尔逊 *Pearson* 相关性检验得到：

表 1　皮尔逊相关参数表

相关参数名称	具体参数值
Pearson 相关系数	0.5829772100309214
P 值	1.3740785217699507e-05

由于 *P* 值小于 0.05，故拒绝原价设，接受备择假设，两数据具有相关性，而由于 *IOPV* 为每只基金的前十个持仓最高值进行计算故，与真实值之间具有一定偏差，且 *P* 值小于 0.05 故拒绝原假设，两数据之间具有相关性。

由于数据量较小，如使用复杂的模型，会导致数据过拟合，不符合真实情况，故选用最小二乘法调整序列整体的位置即可，最小二乘法公式如下：

$$ax + b = IOPV_{真实}$$

其中，x为计算出的参考估计值$IOPV$，通过数据带入可得到如下图所示的拟合结果：

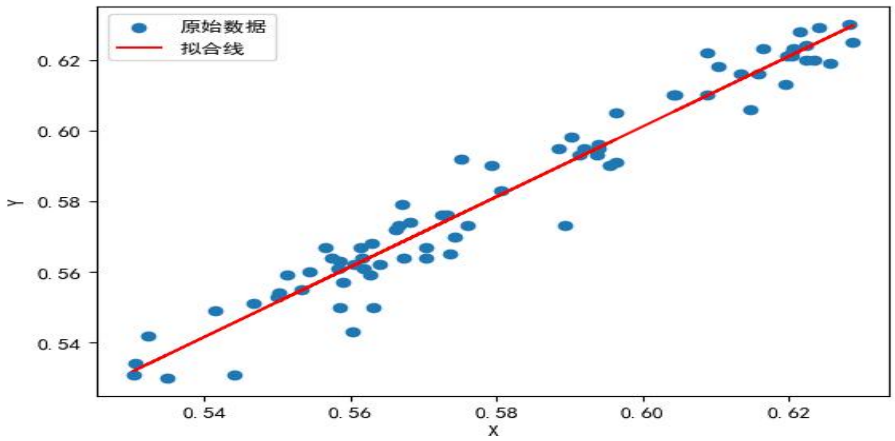

图 5　$IOPV$ 拟合图

在进行最小二乘法线性回归时，通过带入数据得到了最小二乘法线性回归的斜率和截距参数，用于描述数据之间的线性关系，具体参数详见下表：

表 2　最小二乘法参数表

最小二乘法数据	具体数值
估计的斜率（a）	0.9945767761050496
估计的截距 （b）	0.0028690711629792

对参数进行保留 6 位小数处理，代入最小二乘法获得如下公式：

$$0.994577\,x + 0.002869 = IOPV_{真实}$$

通过$IOPV$公式带入分钟级数据，可计算出对应分钟级的$IOPV$，对应详细结果如图所示：

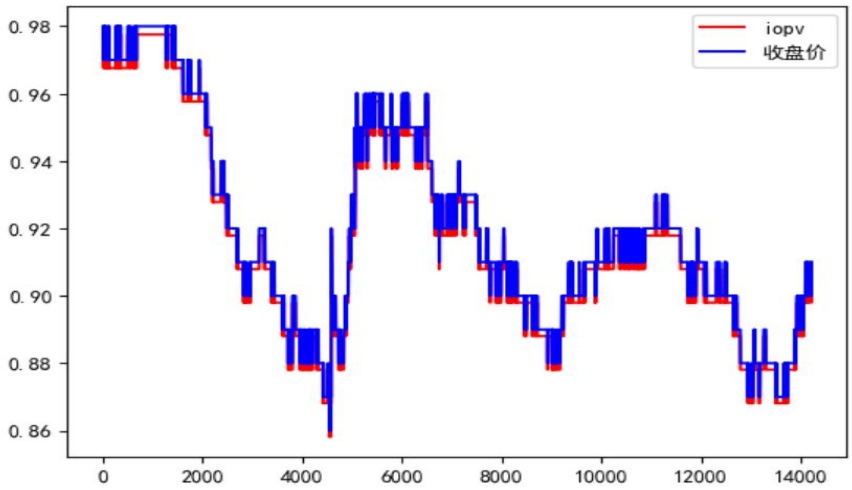

图 6　对应分钟级的$IOPV$图

通过上图，可以获得其相对误差均值：1.4951003398285442e-16，故认为该模型具有较好的拟合效果。

（5）投资风险的β指标

根据净值收益率进行计算式，可计算得出基金平均收益率如下：

$$\overline{R_i}=\frac{\sum_{t=1}^{241}R_{it}}{241}\qquad (i=1,2...463)$$

其中，R_{it}为第i只ETF中第t个时间段的收益率，$\overline{R_i}$为每只ETF分别的收益率。为了反应投资风险的大小量，可使用收益率偏离程度来表示，通过收益率方差来表示这种分散情况：

$$\sigma_i^2=\frac{\sum_{t=1}^{241}\left(R_{it}-\overline{R_i}\right)^2}{240}\qquad (i=1,2...463)$$

根据资本定价模型可得，股票对市场贡献度大小可用方差比值来衡量，具体公式如下所示：

$$\beta_i=\frac{\sigma_{iM}}{\sigma_{M^2}}$$

协方差是用来衡量基金的回报与市场整体回报之间的共同波动性的统计指标。协方差的正值表示基金和市场之间的回报变化是正相关的，而负值表示它们之间的回报变化是负相关的。协方差的绝对值越大，表示两者之间的波动性关系越强烈。协方差的计算可以用于分析基金的风险暴露和与市场相关性。

故可得出β指标结论，如下：

当$\beta<1$时，市场风险大于股票风险；当$\beta>1$时，股票风险大于市场风险；当$\beta=1$时，风险等价。

（6）夏普指数

夏普指数作为一种标准，代表风险与回报，是进行比较资本市场线和基金所处点的相对地位，表示一单位的风险所能获取的回报，因此对应夏普指数越高越好，夏普指数可用下述公式表达 ：

$$S_i=\frac{\overline{R_i}-r_f}{\sigma_i}$$

其中，S_i夏普指数，r_f为市场无风险利率

5.1.3 评价模型建立

结合上述参数可建立评价指标，考虑风险与利益以及*ETF*对应的成分股正向趋势性，可得到整体较优的*ETF*，并通过以上参数对数据进行量化，最终得到前十支整体评分较高的*ETF*，对应本评价指标进行参数设置，详见下表：

表 3　设置默认参数表

参数设置	数值	参数设置	数值
初始投资	150 万元	期货交易费率	0.005 %
期货投资额	100 万元	股票交易费率	0.01 %
现货投资额	50 万元	价格滑点	0.01
期货保证比率	10 %	手续费	0.05

（1）目标函数

由于各参数值量纲不经相同，对数据归一化处理能更好地将各数据量纲统一处理，最终通过对各特征值分割划分得分等级，最终进行求和可得到*ETF*的最终得分：

$$Score_i = S\_diff_i + S\_beta_i + S\_S_i - S\_C_i + S\_\beta_i$$

$$\begin{cases} diff_i = \sum_{d=1}^{n} UpETF_{id} + DownETF_{id} & (i = 1,2 \ldots 886) \\ beta_i = \sum_{m=1}^{241}\sum_{d=1}^{n}\left(\frac{close_{id(m+1)}}{close_{idm}} - 1\right) & (i = 1,2 \ldots 463) \\ C_i = \frac{\sum_{j=1}^{59}\sum_{k=1}^{241}\frac{\max_k - \min_k}{\min_k}}{14219} & (i = 1,2 \ldots 886) \\ \beta_i = \frac{\sigma_{iM}}{\sigma_{M^2}} & (i = 1,2 \ldots 886) \\ S_i = \frac{\overline{R_i} - r_f}{\sigma_i} & (i = 1,2 \ldots 886) \\ S\_diff_i, S\_beta_i, S\_C_i, S\_\beta_i, S\_S_i = F\left(diff_i, |1 - Beta_i|, C_i, \beta_i, S\right)_i & (i = 1,2 \ldots 886) \end{cases}$$

得到具体数据后，由于各数据差值普遍大，为避免影响*ETF*后续的得分评定，因此对各参数进行归一化处理，具体公式如下：

$$F(x) = \frac{x - \max(x)}{\max(x) - \min(x)}$$

由于 *ETF* 基金的价格数据具有震荡性，因此选择 *Dual Thrust* 作为具体的交易策略，具体建立过程如下：

① 计算过程

首先，确立震荡区间同时确定上下轨，由于已知各 *ETF* 收盘价数据，因此可确定 *ETF* 的价格震荡区间，进而分别确定相应的上下轨，具体流程如下所示：

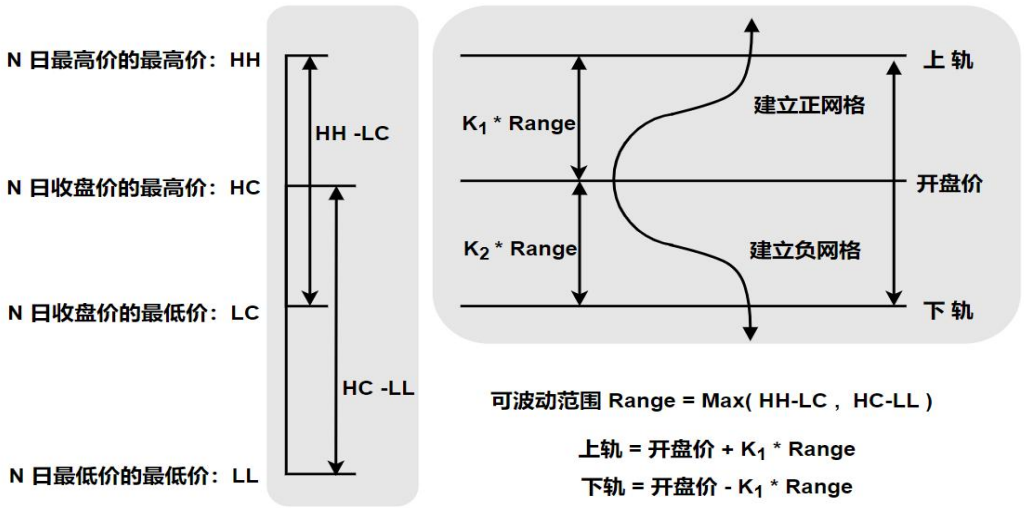

图 7　*Dual Thrust* 计算流程

根据流程图，可知其具体计算公式如下：

$$\begin{aligned} Range &= Max(HH - LC, HC - LL) \\ BuyLine &= Close + k_1 Range \\ SellLine &= Close - k_2 Range \end{aligned}$$

其中 *Range* 表示震荡区间，*Close* 表示收盘价，K_1、K_2 表示相应的触发参数。

② 买卖点确立

当价格向上突破上轨时，如果当时持有空仓，则先平仓，再开多仓；如果没有仓位，则直接开多仓；

当价格向下突破下轨时，如果当时持有多仓，先平空仓，再开空仓；如果没有仓位，则直接开空仓。

通过归一化处理削减差异值大小，平衡各参数的影响因子权重，同时确立好对应的交易策略后，即可根据已知数据进行 *ETF* 最终的得分评定。

5.1.4　前十支 *ETF*

根据上述评价模型进行具体的计算，同时综合各参数完成对每只 *ETF* 基金的等级划分，由于已知各数据具体数值，通过所知数据进行各参数因子对应的等级评定。

因此，假设已知数据为x，则参数的具体等级评定规则如下：

$$Score=\begin{cases}1 & x<0.2\\ 2 & 0.2\le x<0.4\\ 3 & 0.4\le x<0.6\\ 4 & 0.6\le x<0.8\\ 5 & x\ge 0.8\end{cases}$$

通过上式，得到各*ETF*基金参数的等级划分后，综合各参数因子对*ETF*进行最终等级排序，同时得到等级前10只*ETF*，如下表所示：

表4　归一化后前10只*ETF*参数

*ETF*基金代码	波动率	β值	折溢率	收益率	夏普指数	得分
512200	0.6172	0.9505	0.9626	0.9999	0.9495	16
159819	0.6082	0.9269	0.9621	0.3319	0.9685	14
512070	0.6262	0.9049	0.9618	0.1869	0.9413	14
159857	0.4257	0.9025	0.9620	0.6329	0.8734	13
515790	0.4317	0.9114	0.9622	0.6329	0.9582	13
516160	0.5252	0.9028	0.9624	0.5307	0.7368	12
159766	0.4275	0.8733	0.9618	0.4981	0.7176	12
512670	0.4312	0.8348	0.9623	0.4964	0.8808	12
512690	0.5697	0.9241	0.9619	0.4513	0.6888	12
515880	0.4721	0.9217	0.9624	0.4344	0.4289	11

注：参数列保留仅前4位小数；得分列保留至整数位（相同得分以波动率较小为优先排序指标）

基于一二级市场间套利模型下，从*ETF*的波动率、β值、折溢率、收益率、收益率、夏普指数共6方面展开分析，通过基金的投资价值的实证分析，筛选出上述表格中最适合进行套利的前 10 只*ETF*基金。

5.1.5 回测结果检验

利用上述所得交易模型，针对排名前十的*ETF*基金在2023年8月1日至10月31日的历史数据进行回测验证，其中**光伏*ETF*的收益率曲线**如下：

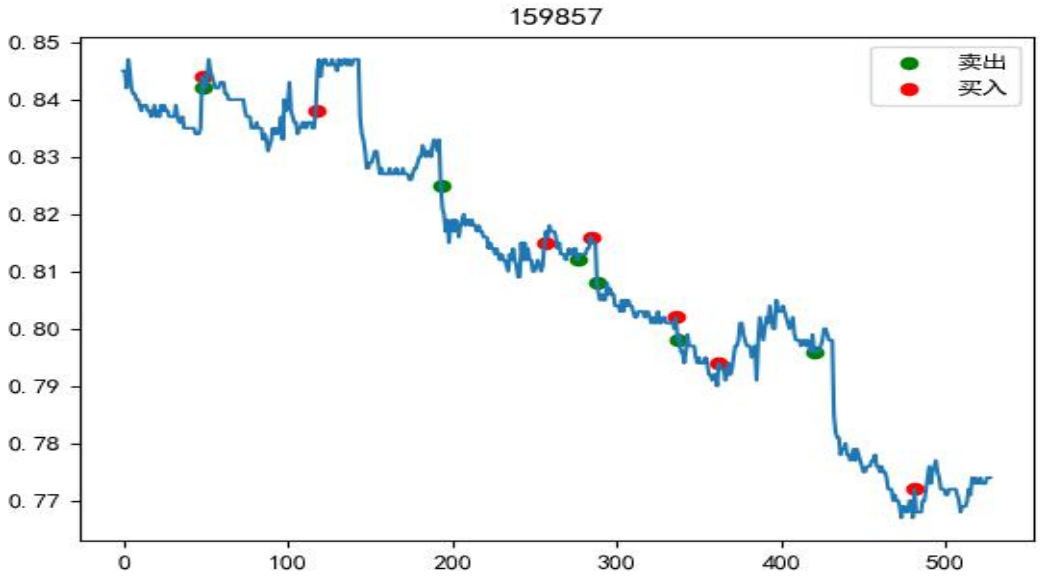

图 8　光伏 ETF 收益率曲线图

因此，我们可以得到相应**光伏** *ETF* **的收益**交易策略具体如下：

表 5　光伏 *ETF* 的收益交易策略表

方式	收盘价	时间	方式	收盘价	时间
买涨	0.844	48	卖出	0.842	49
买涨	0.838	117	卖出	0.825	193
买涨	0.815	256	卖出	0.812	276
买涨	0.816	284	卖出	0.808	288
买涨	0.802	336	卖出	0.798	337
买涨	0.794	362	卖出	0.796	420
买涨	0.772	481			

5.2 任务二：求解跨境 ETF 套利模型

由于跨境 *ETF* 可以进行 $T+0$ 的交易方式，针对交易制度的不同，需要改变任务一所得策略，建立对应跨境 *ETF* 的套利模型，因此针对所得跨境 *ETF* 数据分别进行 *ARIMA*、*LSTM* 和 *MLP* 进行时间序列预测，分别对比三种预测模型的实际效果后，结合 *Dual Thrus* 策略选出最适合进行套利的前 10 只跨境 *ETF* 。

5.2.1 时间序列模型建立

根据所得跨境 *ETF* 的收盘价数据可绘制出每支 *ETF* 相应的曲线图，**选取纳斯达克指数** *ETF* 作为代表数据，绘制其价格曲线，其总体趋势如下：

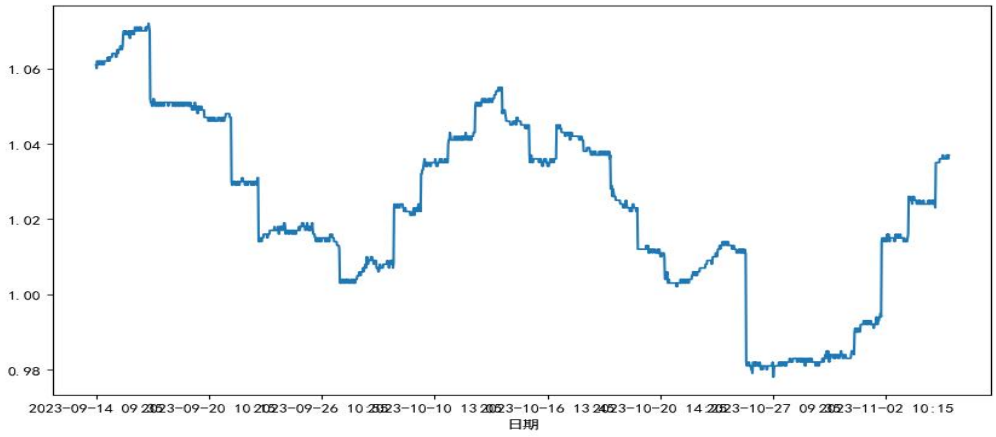

图 9　纳斯达克指数 *ETF* 价格曲线图

由上图可知，纳斯达克指数 *ETF* 的价格曲线具有显著的波动性，同时波动振幅也相对较大，具有一定普遍性，因此将其作为代表性数据进行模型策略初步的确立。

（1） *ARIMA* 时间序列

由于时间序列的大多概率理论都是关于平稳时间序列的，时间序列数据集可能包含趋势和季节性，在建模之前需要将其删除，为保证预测过程的某些性质是不随时间而变化的，因此时间序列建模第一步是将时间序列转为平稳时间序列。我们采用差分对时间序列数据集进行转换，因为原序列呈现出近似线性趋势，所以分别使用一阶差分和二阶差分进行平稳处理。

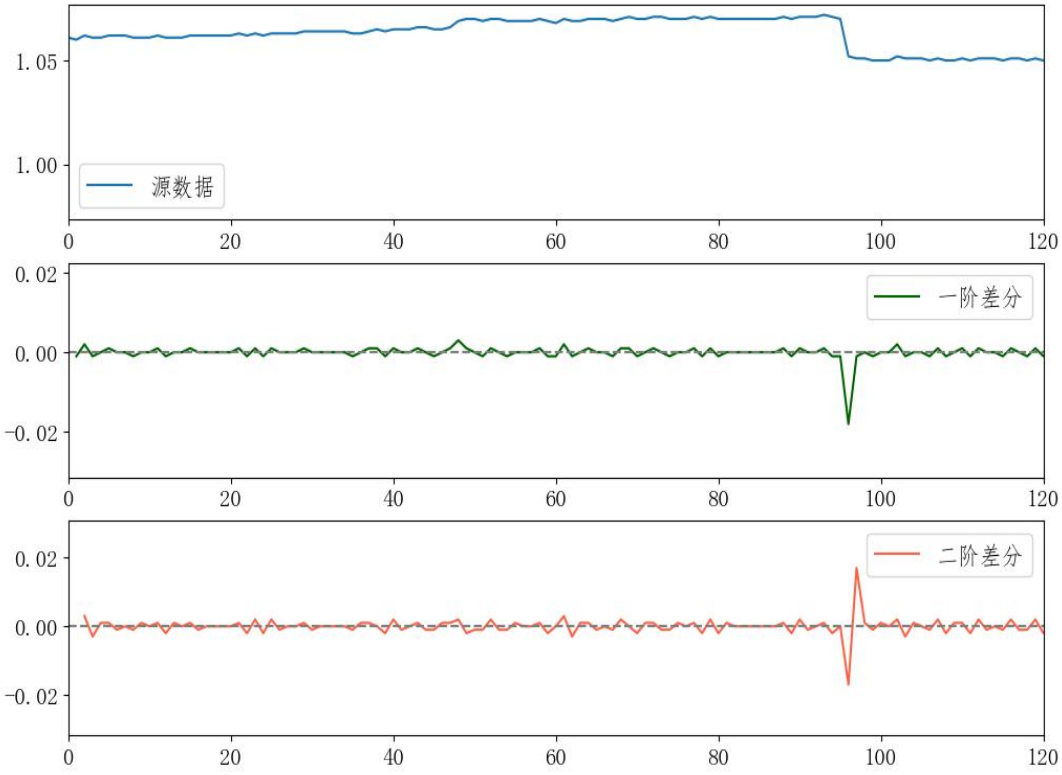

图 10　平稳序列处理图

根据上图可知，通过一阶差分处理数据集序列平稳性较好，因此选定一阶差分进行数据的平稳处理。

经过具体测试发现， *ARIMA* 时间序列预测输入数据需要按照一定的时间间隔排列，即数据集在时间段内可以呈现一定的周期性，而跨境 *ETF* 在所选时间段内普遍周期性程度小，因此模拟效果较差。

（2） *MLP* 预测处理

首先针对数据集异常问题，进行重复值数据进行清洗，并且过滤掉交易规模较小导致的异常数据。其次，进行 *MLP* 预测需要对数据集先进行归一化处理，防止参数因子差别过大，同时将数据集划分为训练集和测试集，并且两者的比例设置为 7：3，最终进行实际模拟得到的效果如下图所示：

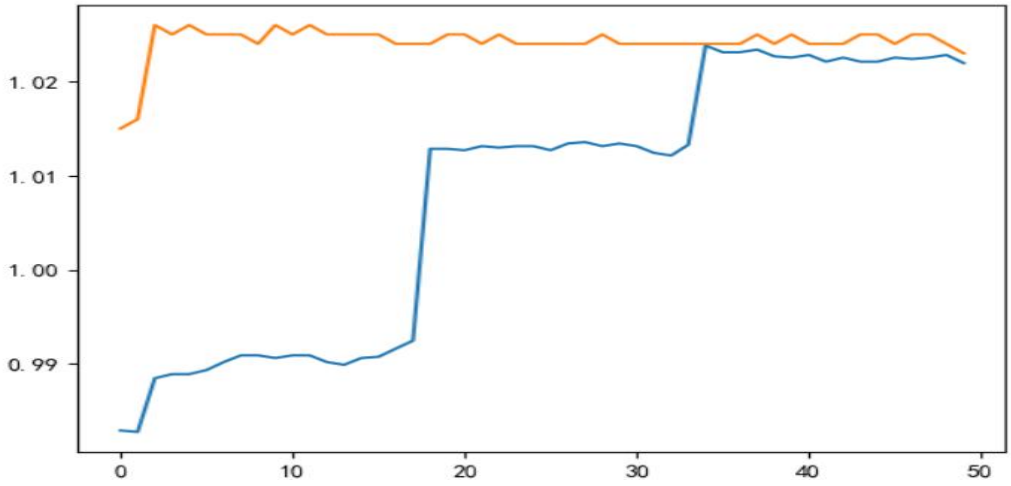

图 11　*MLP* 预测图

根据上图可知，采用 *MLP* 预测模型时， *ETF* 价格曲线具有一定的趋势，但与实际数据误差相对较大。

(3) *LSTM* 预测

由于跨境 *ETF* 数据具有一定的长期依赖性，而 *LSTM* 常被用于解决一般递归神经网络中普遍存在的长期依赖问题，因此选用 *LSTM* 作为所使用的其中一种预测模型。首先，我们将 *ETF* 的收盘价作为数据集，在进行预测的过程中，$t-1$ 时刻的价格 h_{t-1} 会被复制到 t 时刻，与当前时刻输入价格进行整合后经过一个带权重和偏置的 *tanh* 函数后形成输出，并且再次被复制到下一时刻。

遗忘门根据记录时间减少过去一段时间的价格权重，具体表达式如下所示：

$$f_t = \sigma\left(W_f[h_{t-1},x_t] + b_f\right)$$

其中， f_t 表示神经层的输出向量， h_{t-1} 表示 $t-1$ 时间的输入价格， W_f 和 b_f 分别表示对应 tanh 函数的权重向量和偏置向量， x_t 表示当前时间输入的价格， σ 表示神经层的输出函数。

记忆门负责从当前输入提取有效信息，同时进行筛选和信息的评级，具体的实现公式如下：

$$C_t' = \tanh(W_c[h_{t-1}, x_t] + b_c)$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

其中 C_t' 表示当前输入价格的记忆等级，i_t 表示神经层的输出向量。

通过记忆门和遗忘门的相互作用，导致 *LSTM* 可以长期记忆重要信息，并且记忆可以随着输入进行动态调整，最终得到具体的预测效果如下图所示：

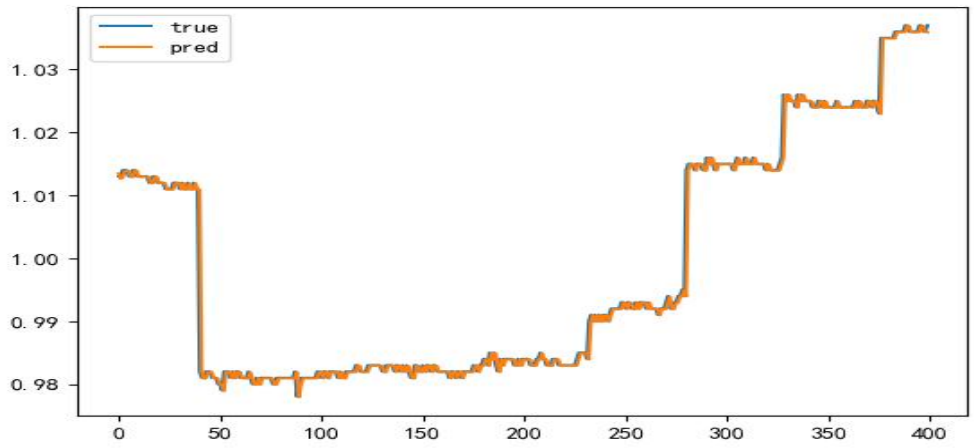

图 12　*LSTM* 预测曲线图

分析上图，可知 *LSTM* 能够有效地捕捉时间序列数据中的复杂模式和动态变化，并且在预测以上跨境 *ETF* 数据的效果比较显著，与真实数据的误差相对较小，因此大致可以确定选用 *LSTM* 作为预测模型。

5.2.2 预测模型对比

通过函数调用并进行计算，最终整理得到三种预测模型具体的预测误差数据，具体如下表所示：

表 6　预测误差数据表

预测模型	*ARIMA*	*MLP*	*LSTM*
平均绝对误差	0.00823	0.01654	0.00073
均方误差	1.27503	0.00047	4.40198
均方根误差	0.01242	0.02169	0.00209
决定系数	20.13011	-130.2182	0.98739

对比以上具体数据，显然 *ARIMA* 时间序列的预测效果高于 *MLP*，而 *LSTM* 预测模型的效果都高于其它两种预测模型，因此选定 *LSTM* 作为最终的预测模型。

5.2.3 网格交易策略建立

根据交易特点初步尝试基金定投策略，首先选定定投的周期间隔为每周的星期一进行定投，通过观察 *ETF* 收盘点位的平均值均线和市场价之间的关系来调整定投金额，在在均线上方时，我们就减少定投金额，反之则增加。

在定投策略的基础上增加止盈策略和补仓策略，设置一定量的目标收益指数为，当上证指数大于目标收益时自动止盈赎回，指数下跌一定比例，提高投入金额一定的比例，具体参数数据如下表：

表 7　基金定投参数

定投参数	参数值	定投参数	参数值
初始本金	1500000	上证指数大于	1000
定投金额	20000	持有仓位大于	70%
定投周期	每周/周一	持有收益率大于	5%
参考指数	1.000016	卖出金额	40%持有份额
补仓金额	剩余流动资金 20%	买入 *MACD* 临界点	3%

设定具体参数后，实际运行得到的策略实施效果如下：

图 13　基金定投效果

根据上图基金业绩走势，实行该策略无法带来有效收益，基金的收益率为-4.07%，因此认为基金定投策略对跨境 *ETF* 模型的运营效果相对较差，应选择其它盈利策略。

由于任务一建立的*Dual Thrust*策略模型效果可观，但由于任务一所建立策略在大波趋势行情中，只能获取到微小的次级波段，针对这一缺点对原本策略进行改进，先通过*LSTM* 模型对价格进行预测得到对应的变化趋势曲线后，再结合网格交易策略进行优化。

针对跨境*ETF* 波动性较大和交易费低等特点，进行网格交易策略优化，随着盈利水平的不同，跟踪的止盈止损线的跟进速度不一样，依靠不同的加速系数实现，具体实现思路如下：

（1）设定网格区间和大小

为扩大收益率，选择震荡频率最高区间进行运行，其中震荡行情可适当收小，需要尽量多的抓住每一个小的波动；趋势行情可适当放大，防止过早满仓或空仓。

（2）计算建仓份数

建立一个用于网格交易的底仓，建仓仓位大小需根据标的当前的价位相对设定价格区间的相对位置而定，如果当前价格相对较低，则底仓可以稍微提高；价格相对较高，则可以先轻仓，具体计算公式如下：

$$N = 1 + \frac{\left(P_{High} - P_{Now}\right)}{Big}$$

其中，N 为建仓份数，P_{High} 为上限价格，P_{Now} 为当前价格，Big 为网格大小。

（3）设定每格份额

每格份额需要根据投入资金、网格价格上下限及网格大小来推算，具体推算公式如下：

$$base = \frac{Cost}{\left(N + \frac{\left(P_{now\ (i-1)} + P_{Low}\right)}{2}\right)\left(\frac{\left(P_{now} - P_{Low}\right)}{Big}\right)}$$

其中$base$表示每格份额，$Cost$ 表示投入资金，P_{Low} 表示上限价格，$P_{Now\ (i-1)}$表示当前位置下一格的价格， Big 表示网格大小。

（4）买卖点确认

每日以上次交易价格为中心，买卖数量按照每个网格的交易量进行计算，当价格行情波动超过一个网格时，将买入 100 手，并且实际行情与模型预测趋势结果大体相同时买入 200 手，同时将当前交易价格作为新的交易中心。

同时可以确定最终的等级评分模型，具体如下：

$$Score_i = S\_benefit_i + S\_beta_i - S\_sharpe_i$$

$$\begin{cases} f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right) & \left(t = 1,2...2256\right) \\ C_t' = \tanh\left(W_c\left[h_{t-1}, x_t\right] + b_c\right) & \left(t = 1,2...2256\right) \\ i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) & \left(t = 1,2...2256\right) \\ sharpe_i = \dfrac{benefit_i - nrbenefit}{change_i} & \left(i = 1,2...99\right) \\ \beta_i = \dfrac{Cov_{benefit}}{\sigma^2_{benefit}} & \left(i = 1,2...99\right) \\ S\_benefit_i, S\_\beta_i, S\_Sharp_i = F\left(benefit_i, \beta_i, Sharp_i\right) & \left(i = 1,2...99\right) \end{cases}$$

5.2.4 策略实施与排序

通过*LSTM* 预测模型进行趋势预测，同时对所用策略进行改进后，进行实际回测并对跨境*ETF* 进行排名，沿用任务一等级排名制度综合考虑各参数，得到对应等级排名前十的跨境*ETF* 数据，具体如下表：

表 8　等级排名前十*ETF* 基金归一化处理表

代码	夏普率	*Beat*	收益率	得分
513230	0.077732996	0.96042	1	9
159513	0.108052329	0.94654	0.895951458	9
159612	0.077732996	0.87543	0.883846124	9
513360	0.077732996	0.93878	0.876758007	9
513290	0.072500801	1	0.841747237	9
159742	0.072436117	0.89321	0.821132957	9
513590	0.069237904	0.98345	0.819330956	9
159920	0.062307903	0.92789	0.81899403	9
159960	0.069116249	0.95999	0.815810455	9
159691	0.01044604	0.97111	0.80292725	9

从投资者的角度考虑，为提高*ETF* 收益，主要针对收益率数据进行排序，最终得到排名前十的跨境*ETF* 数据，具体如下表：

表 9　收益排名前十跨境 *ETF* 基金表

代码	夏普率	*Beat*	收益率
159823	0.567390999	1.00000000	0.754160032
513330	2.516624323	0.98654321	0.564824
159941	0.567390999	0.91543210	0.542796051
159866	0.567390999	1.07876543	0.529897879
513290	0.23101253	1.09987654	0.466189151
159742	0.226854046	0.93321098	0.428677564
513360	0.021240543	1.02345678	0.425398482
510900	-0.424289992	1.06789012	0.424785381
159509	0.013419344	0.99999998	0.418992264
513230	-3.7584951	1.01111111	0.39554883

综上可得，收益排名主要关注收益率，更符合投资者们的预期期望，也能更好地选择适合套利需求的 *ETF* 跨境基金。

5.2.5 回测结果检验

利用改进后模型，针对跨境 *ETF* 进行回测验证，得到各 *ETF* 的收益率曲线，其中**华夏恒生 *ETF* 的运行结果**如下：

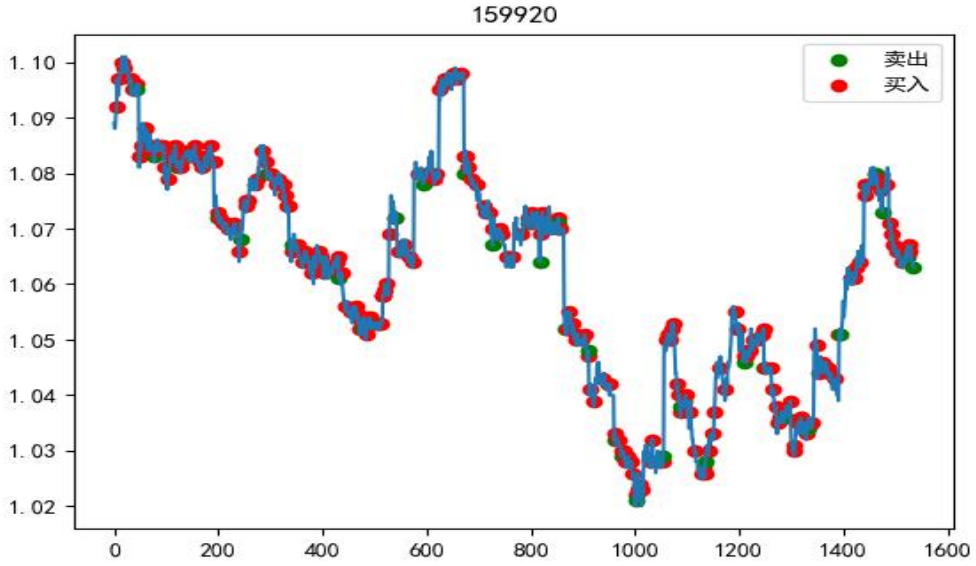

图 14　华夏恒生收益率曲线

收益率 3.7129%，最大回撤 -4.4535%，投资的收益率较高，但存在一定的市场风险，与市场波动高度相关，风险调整后的回报率为正值，因此本模型更符合投资者们的预期期望，能更好地为他们选择适合套利需求的 *ETF* 跨境基金。

5.3 任务三：跨境 ETF 和股指期货套利策略

在任务三中，对应其设计目标是基于股票相关性和价格趋势的分析，建立一个基于跨境 *ETF* 和股指期货的套利策略模型来实现盈利。

（1）相关性分析和阈值的设置

由于跨境 *ETF* 与国际市场的联动性，为保证投资组合的收益尽可能高，应保证证券组合是正相关的，收益是同向变动的，因此针对数据集先进行相关性的分析，由于各股指期货下包含多支股票，将股指期货分解为其所有成分股票，并针对跨境 *ETF* 与股指期货的关系进行相关性分析，部分结果如下：

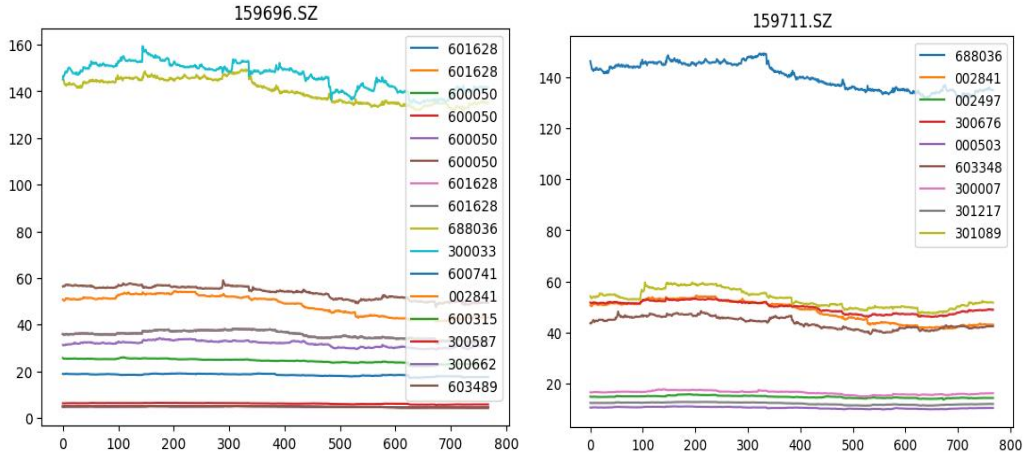

图 15　相关性分析

得到 *ETF* 与股票的相关性数据后，通过筛选其中相关性程度较高的数据，并设置相关性阈值大小为 0.9，得到与每支 *ETF* 相关性大小超过阈值的股票列表，如下表所示：

表 10　部分高相关性股票列表

跨境 *ETF* 代码	股票列表
159506.SZ	603259、600789
513030.SH	600036、002841、002497、300587
513200.SH	603259、002653、600789
513800.SH	605358、300699、002250、603348
513810.SH	000066、001227、601118

进行相关性分析后，对 *ETF* 相关性高的股票列表进行线性回归拟合得到相应波动趋势，得到一系列趋势大体相同、波动幅度大小近似的股票数据集。

已知各数据拟合结果的真实率R^2，同时以最后的收盘价与开盘价作为主要评价标准得到*ETF*和股票的相应得分情况，具体得分计算公式如下：

$$Score = R^2(close_d - open_d)$$

其中$close_d$和$open_d$分别表示第d天的收盘价和开盘价。

（2）投资组合排序

针对每只*ETF*和股票的得分计算后，进行排序，最终选取排名前五的*ETF*或者股票作为投资组合的一部分，通过计算得到在第一天排名前五的*ETF*或者股票数据具体如下：

表 11　第一天排名前五的*ETF*或者股票数据表

*ETF*基金代码	得分
513690.SH	11.50072301
513600.SH	6.457002083
513530.SH	5.035401931
513290.SH	4.471945169
159747.SZ	3.676623356

确定具体的股票投资组合后，根据其相应的得分设置投入资金的权重大小，具体的权重计算公式如下：

$$W_i = \frac{socre_i}{\sum_{i=1}^{5} score_i} \qquad (i = 1,2...5)$$

W_i表示给予第i只基金或股票的权重大小，$score_i$表示第i只基金或股票的得分。根据此式，整理可得第一天的具体投资组合和各成分股投资占比，详见下表：

表 12　投资组合数据信息

*ETF*基金代码	得分	资金占比
513690.SH	11.50072301	36.9%
513600.SH	6.457002083	20.7%
513530.SH	5.035401931	16.1%
513290.SH	4.471945169	14.5%
159747.SZ	3.676623356	11.8%

设定好当天的投资组合后，沿用任务二所采取的*LSTM* 模型预测当日股票的价格波动变化趋势，同时结合*Dual Thrust*交易策略和网格交易策略进行当天的买卖交易，在第二天再次进行当天的投资组合确立，依此累加每日的收益率，得到最终的总收益，具体实测结果。

通过计算可得到最终的总收益，部分实测结果如下表所示：

表 13　*ETF* 组合投资实测表

天数	收益率
1	0.12804755
2	-0.458324248
3	0.12804755
4	0.12804755
5	0.424237034
6	0.363190494

以上组合投资主要选取景顺长城恒生消费*ETF* 、中证港股通科技*ETF* 、招商上证港股通*ETF* 和富国纳斯达克 100*ETF* ，在该组合下在高风险情况下可获取较高收益。

六、模型评价与推广

6.1 模型优点

（1）模型提供了多种分析方法和策略，包括基于时间序列分析的模型、神经网络预测模型、网格交易策略等，从不同角度探索了跨境 ETF 套利的可能性。

（2）模型的准确性高，对数据进行了清洗，包括处理重复值和异常数据，提高了数据的质量，对数据进行归一化处理，将不同参数的数据量纲统一处理，能更好地平衡各因素的影响。

（3）模型模型基于*Dual Thrust*策略建立买卖点，具体规则清晰，便于操作和执行，增加了模型的实用性和现实适应性，考虑了不同市场情况下的调整策略，如加速系数、建仓份数等，以适应跨境数据的波动性。

（4）模型基于具体的数值结果，且引入了额外数据信息，提高了决策的准确性，有助于理解各因素的重要性和相互关系。

6.2 模型缺点

（1）模型依赖大量数据，包括每只 *ETF* 的相关参数和历史数据。如果数据不准确或者历史数据不足，模型的预测和评估可能不够可靠。

（2）模型提供了套利策略，但对于风险管理的讨论相对较少。在实际操作中，风险管理对于套利策略至关重要，模型未提供详细的风险控制方案。

（3）模型主要基于历史数据和市场情况，不能充分考虑未来市场变化和突发事件所带来的影响，因此不能预测所有可能发生的情况。

6.3 模型推广

本模型在提高金融决策和套利方面具有潜力，但需在实际应用中需根据具体市场情况进行调整和优化。由于金融市场的多样性和动态性，要求模型适应不同市场和时间段的特点。此外，数据质量和可用性也是一个关键问题，需要综合考虑数据来源、数据频率、数据质量和数据处理方法，以确保模型的可行性和鲁棒性。模型的推广可应用到其他行业和领域，以解决类似决策问题。例如，这些模型的时间序列预测和机器学习方法可以用于供应链管理，股票市场分析，医疗诊断等领域。通过调整模型参数和特征工程，可以根据具体行业需求进行定制化，提高决策的准确性和效率。

总之，本模型的推广不仅有望在金融领域提供更多决策支持，还可以拓展到其他领域，为不同行业的专业人士提供更多解决方案。然而，需要谨慎处理模型的适应性和数据问题，以确保在不同领域的成功应用。

七、参考文献

[1] 岳曲.我国开放式基金市场资本资产定价模型研究[J].现代商业, 2015(29):2.DOI:CNKI:SUN:XDBY.0.2015-29-046.

[2] 马红军.开放式基金风险收益评价指标的比较与改进[J].中国流通经济, 2002, 16(4):3.DOI:CNKI:SUN:ZGLT.0.2002-04-015.

[3] 李金丹.基于"已实现"波动金融波动的符号时间序列分析[D].天津大学,2014.DOI:10.7666/d.D486669.

[4] 罗杰,胡佳垚.通过网格交易法进行趋势跟踪实现量化投资——基于掘金量化平台的程序化交易[J].中文科技期刊数据库(全文版)经济管理, 2022(1):3.

[5] 沈腾飞.中国市场条件下股指期货,ETF 期权与 ETF 的套利策略分析[J].天津财经大学, 2015.

[6] 韩广哲.股指期货在基金公司投资组合管理中的应用研究[D].中国社会科学院金融研究所(北京), 中国社会科学院, 中国社会科学院金融研究所,2010.

附录

一、支持材料的文件列表

(1) 问题一支撑材料：

1. ARIMA 预测.ipynb
2. 1 分钟级 IOPV 计算.ipynb
3. 波动率计算.m
4. 收益率计算.ipynb
5. Dual Thrust.py
6. 归一化和等级划分.m

(2) 问题二支撑材料：

1. 预测模型对比.ipynb

(3) 问题三支撑材料：

1. 拟合.ipynb
2. 回测代码.py

二、数据材料

1.回测数据文件夹

三、程序源代码

1.任务一程序代码：

任务一 *Python* 代码分析

目的： *ARIMA* 预测

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import matplotlib
import seaborn as sns
from  statsmodels.tsa.arima_model  import  ARIMA
import statsmodels as sm
from scipy import stats
import re
import efinance as ef
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf

matplotlib.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

from keras import backend as K
```

```
data_kua = pd.read_excel(r'c:\Users\Lenovo\Desktop\跨境 etf 代码表.xlsx')
code_names = data_kua['thscode']
code_data = []
print(code_names)
for i in code_names:
    if i not in code_data:
        code_data.append(i)
print(code_data)
def get_have(name:str)->str:
    name = re.findall("\d+",name)[0]
    return name
def diff(df,col):
    font = {"size":15,
       "family":"fangsong"}
    matplotlib.rc("font",**font)
    matplotlib.rcParams['axes.unicode_minus']=False

    df["diff_1"] = df[col].diff(1)  #一阶差分
    df["diff_2"] = df["diff_1"].diff(1)  #二阶差分

    #平稳数据折线图
    plt.figure(figsize=(12,8))
    plt.subplot(3,1,1)
    plt.plot(df[col].values,label="源数据")
    plt.xlim(0,120)
    plt.legend()
    plt.subplot(3,1,2)
    plt.plot(df["diff_1"].values,c="darkgreen",label="一阶差分")
    plt.plot([0,120],[0,0],"--",c = "grey")
    plt.xlim(0,120)
    plt.legend()
    plt.subplot(3,1,3)
    plt.plot(df["diff_2"].values,c="tomato",label="二阶差分")
    plt.plot([0,120],[0,0],"--",c = "grey")
    plt.xlim(0,120)
    plt.legend()
    plt.show()

    #ACF PACF
    print("-"*50,"未平稳数据 ACF 与 PACF","-"*50)
    fig = plt.figure(figsize=(12,8))
    ax1 = fig.add_subplot(211)
    fig = plot_acf(df[col], lags=40,ax = ax1)
    ax2 = fig.add_subplot(212)
    plot_pacf(df[col], lags=40,ax = ax2)
```

```
        plt.show()

        #一阶差分后的 ACF PACF
        print("-"*50,"一阶差分数据 ACF 与 PACF","-"*50)
        fig = plt.figure(figsize=(12,8))
        ax1 = fig.add_subplot(211)
        fig = plot_acf(df["diff_1"][1:].values, lags=40,ax = ax1)
        ax2 = fig.add_subplot(212)
        plot_pacf(df["diff_1"][1:], lags=40,ax = ax2)
        plt.show()

    #未差分平稳性检测（ADF 检验、单位根检验）
    from  statsmodels.tsa.stattools  import  adfuller  as  ADF
    print(u'原始序列的 ADF 检验结果为：',  ADF(close_data["收盘"]))
    #返回值依次为 adf、pvalue、usedlag、nobs、critical  values、icbest、regresults、resstore  p<0.05 时表示稳
定
    #一阶差分平稳性检测（ADF 检验、单位根检验）
    from  statsmodels.tsa.stattools  import  adfuller  as  ADF
    print(u'一阶差分序列的 ADF 检验结果为：',  ADF(close_data["diff_1"][1:]))
    #返回值依次为 adf、pvalue、usedlag、nobs、critical  values、icbest、regresults、resstore  p<0.05 时表示稳
定
    from statsmodels.tsa.arima.model import ARIMA

    pmax = 5
    qmax = 5
    bic_matrix  =  []  #bic 矩阵
    for  p  in  range(pmax+1):
        tmp  =  []
        for  q  in  range(qmax+1):  #存在部分报错，所以用 try 来跳过报错。
            try:
                tmp.append(ARIMA(close_data["收盘"],order=(p,1,q)).fit().bic)
            except:
                tmp.append(None)
        bic_matrix.append(tmp)
    bic_matrix  =  pd.DataFrame(bic_matrix)  #从中可以找出最小值
    p,q  =  bic_matrix.stack().idxmin()
    # #先用 stack 展平，然后用 idxmin 找出最小值位置。
    print(u'BIC 最小的 p 值和 q 值为：%s、%s'  %(p,q))

    pmax = 14
    qmax = 14
    aic_matrix  =  []  #bic 矩阵
    for  p  in  range(pmax+1):
        tmp  =  []
        for  q  in  range(qmax+1):  #存在部分报错，所以用 try 来跳过报错。
```

```
            try:
                tmp.append(ARIMA(close_data["收盘"],order=(p,1,q)).fit().aic)
            except:
                tmp.append(None)
        aic_matrix.append(tmp)
    aic_matrix  =  pd.DataFrame(bic_matrix)  #从中可以找出最小值
    p,q  =  bic_matrix.stack().idxmin()
    # #先用 stack 展平，然后用 idxmin 找出最小值位置。
    print(u'AIC 最小的 p 值和 q 值为：%s、%s'  %(p,q))
```

任务一 *Python* 代码分析

目的：进行 1 分钟级 *IOPV* 计算

```
import os
import numpy as np
import pandas as pd
import efinance as ef
import re
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
import akshare as ak
fund_fh_em_df = ak.fund_fh_em()
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
path = r'C:\Users\Lenovo\jupyter_notebook\金融建模大赛 A 数据'
result = os.listdir(path)
result
def get_have(name:str):
    name = re.findall("\d+",name)[0]
    return name


def linear_model(x, a, b):
    return a * x + b
end_frame= []
for code_index in range(len(result)):
    try:
        data_fund = pd.read_excel(path+'/'+result[code_index])
        up_down = []
        code_fund = get_have(result[code_index])
        data = ef.fund.get_quote_history(code_fund,pz=75)
        # public_dates = ef.fund.get_public_dates(code_fund)
        # kk = ef.fund.get_invest_position(code_fund,dates=public_dates[0])


        db = ef.stock.get_quote_history(code_fund,beg='20230714',fqt=0)
```

```
            x = np.array(db['收盘'])
            y = np.array(data['单位净值'][::-1])
            # 定义线性回归模型函数


            # 使用最小二乘法估计参数
            params, covariance = curve_fit(linear_model, x, y)
            # 提取估计的模型参数
            a_est, b_est = params
            print('avg_loss:',np.average(linear_model(x, a_est, b_est) - y))
            get_like = linear_model(data_fund['收盘价'], a_est, b_est)
            # 每只基金的折溢价
            for i,j in zip(get_like,data_fund['收盘价']):
                cha_values = abs(i-j)
                if cha_values > 0:
                    up_down.append(cha_values)
            sum_values = np.sum(up_down)
            print(f'{code_fund}',sum_values)
            end_frame.append([code_fund,sum_values])
        except:
            code_fund = get_have(result[code_index])
            print('数据异常',code_fund)
        print(end_frame)


    end_frame_sort = pd.DataFrame(end_frame,columns=['ETF 代码','折溢率']).sort_values('折溢率')
    end_frame_sort = end_frame_sort.drop(end_frame_sort[end_frame_sort['折溢率']>1000].index)
    print(end_frame_sort)
    # 获取历史净值
    code_fund = get_have(result[-3])
    data = ef.fund.get_quote_history(code_fund,pz=75).iloc[3:,:]
    print(data)
    public_dates = ef.fund.get_public_dates(code_fund)
    print(public_dates)
    kk = ef.fund.get_invest_position(code_fund,dates=public_dates[0])
    kk
    # 获取最新公开的持仓数据
    T = 1
    def get_clean_values(day):
        u_data_ = pd.DataFrame()
        try:
            for data_every in data['日期']:
                print(data_every)
                values_ = pd.DataFrame()
                one_all_before = data['单位净值'][data[data['日期']==data_every.split('
')[0]].index.to_list()[0]+1]
                data_every = data_every.replace('-','')
```

```
            for i,j in zip(kk['股票代码'],kk['持仓占比']):
                b = ef.stock.get_quote_history(i,beg=data_every,end=data_every,klt=5)['收盘']*j*0.01
                if len(b) == 0:
                    return u_data_
                values_ = pd.concat([values_,b],axis=1)
            one_data_ = np.array(np.sum(values_,axis=1)/one_all_before)[-1]
            u_data_ = pd.concat([u_data_,(np.sum(values_,axis=1) / one_data_)],axis=1)
    except:
        return u_data_
    return u_data_


clean_data_ = get_clean_values(public_dates[0])
clean_data_
import matplotlib.pyplot as plt
code_fund = '588460'
# plt.plot(range(len(clean_data_)),data['单位净值'][:31],c = 'b')
db = ef.stock.get_quote_history(code_fund,beg='20230714')
data = ef.fund.get_quote_history(code_fund,pz=75)
# db_1102 = ef.stock.get_quote_history(code_fund,beg='20231102',end='20231102',klt=5)
print(len(db['收盘']))
print(len(data['单位净值']))
plt.plot(range(len(data['单位净值'])),data['单位净值'][::-1],c = 'r',label = '单位净值')
plt.plot(range(len(data['单位净值'])),db['收盘'],label = '收盘价（日）')
plt.legend()
# plt.plot(range(len(clean_data_[0])),clean_data_[0])
# plt.plot(range(len(clean_data_[0])),db_1102['收盘'])
plt.show()
import numpy as np
from scipy.stats import pearsonr
# db = ef.stock.get_quote_history(code_fund,beg='20230801')
# 执行皮尔逊相关性分析
correlation, p_value = pearsonr(data['单位净值'][::-1], db['收盘'])

# 打印相关系数和 p 值
print(f"Pearson 相关系数: {correlation}")
print(f"p 值: {p_value}")
db = ef.stock.get_quote_history(code_fund,beg='20231031',end='20231031',klt=5)
plt.plot(range(len(clean_data_.iloc[:,3])),clean_data_.iloc[:,3],c = 'r',label = '估值 iopv')
plt.plot(range(len(db['收盘'])),db['收盘'],label = '收盘')
plt.legend()
plt.show()
#用公式计算发现具有相似的趋势性
import numpy as np
from scipy.stats import pearsonr
```

```
# 执行皮尔逊相关性分析
correlation, p_value = pearsonr(clean_data_.iloc[:,3], db['收盘'])


# 打印相关系数和 p 值
print(f"Pearson 相关系数: {correlation}")
print(f"p 值: {p_value}")
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

db = ef.stock.get_quote_history(code_fund,beg='20230714')
x = np.array(db['收盘'])
y = np.array(data['单位净值'][::-1])

# 定义线性回归模型函数
def linear_model(x, a, b):
    return a * x + b



# 使用最小二乘法估计参数
params, covariance = curve_fit(linear_model, x, y)

# 提取估计的模型参数
a_est, b_est = params

# 打印估计的参数
print(f"估计的斜率 (a):{a_est}")
print(f"估计的截距 (b):{b_est}")

# 绘制原始数据和拟合线
plt.scatter(x, y, label="原始数据")
plt.plot(x, linear_model(x, a_est, b_est), color='red', label="拟合线")
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()

print('avg_loss:',np.average(linear_model(x, a_est, b_est) - y))
get_like = linear_model(data_fund['收盘价'], a_est, b_est)
plt.plot(range(len(get_like)), get_like,c = 'r',label = 'IOPV')
plt.plot(range(len(get_like)),data_fund['收盘价'],c = 'b',label = '收盘价')
plt.legend()
```

```
plt.plot()
plt.plot(range(len(data['单位净值'])),data['单位净值'])
plt.plot(range(len(data['单位净值'])),db['收盘'][::-1])
plt.show()
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

x = np.array(data['单位净值'][::-1])
y = np.array(db['收盘'])

# 定义线性回归模型函数
def linear_model(x, a, b):
    return a * x + b



# 使用最小二乘法估计参数
params, covariance = curve_fit(linear_model, x, y)

# 提取估计的模型参数
a_est, b_est = params

# 打印估计的参数
print(f"估计的斜率 (a):{a_est}")
print(f"估计的截距 (b):{b_est}")

# 绘制原始数据和拟合线
plt.scatter(x, y, label="原始数据")
plt.plot(x, linear_model(x, a_est, b_est), color='red', label="拟合线")
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()

print('avg_loss:',np.average(linear_model(x, a_est, b_est) - y))
get_like = linear_model(data_fund['收盘价'], a_est, b_est)

plt.plot(range(len(get_like)), get_like,c = 'r',label = 'iopv')
plt.plot(range(len(get_like)),data_fund['收盘价'],c = 'b',label = '收盘价')
plt.legend()
plt.plot()
print('日 iopv 与收盘价均误差',np.average(get_like - data_fund['收盘价']))
```

```
    plt.plot(range(len(get_like)), get_like,c = 'r',label = 'iopv')
    plt.plot(range(len(get_like)),data_fund['收盘价'],c = 'b',label = '收盘价')
    plt.legend()
    plt.plot()
    print('日 iopv 与收盘价均误差',np.average(get_like - data_fund['收盘价']))
    # 每只基金的折溢价 和 次数
    up_down = []
    for i,j in zip(get_like,data_fund['收盘价']):
        cha_values = abs(i-j)
        if cha_values > 0:
            up_down.append(cha_values)
    print(np.sum(up_down))


    T = 0
    db_arr = []
    tii = [None]
    for i,j in zip(get_like,data_fund['收盘价']):#i iopv ,j 收盘
        if i < j and tii[-1] != 's': # 溢价
            db_arr.append([i,j])
            tii.append('s')
        if i > j and tii[-1] != 'b': # 折价
            db_arr.append([i,j])
            tii.append('b')
    tii = tii[1:]
    print(db_arr)
    print(tii)
    for i,j in zip(db_arr,tii):
        # 投资金额小于总交易 10%
        if j == 's':
            diff = i[1] - i[0]
            # 成本 = i[0]*手数*申赎费 + i[1]*交易费（股票/etf)
        if j == 'b':
            diff = i[0] - i[1]
            # 成本 = i[0]*手数*申赎费 + i[1]*交易费（股票/etf)









    code_index = -1
    data_fund = pd.read_excel(path+'/'+result[code_index])
    up_down = []
    code_fund = get_have(result[code_index])
    print(code_fund)
    data = ef.fund.get_quote_history(code_fund,pz=75)
```

```
print(data)
# public_dates = ef.fund.get_public_dates(code_fund)
# kk = ef.fund.get_invest_position(code_fund,dates=public_dates[0])

db = ef.stock.get_quote_history(code_fund,beg='20230714',fqt=2)
x = np.array(db['收盘'])
y = np.array(data['单位净值'][::-1])
plt.plot(range(len(x)),x,c='r',label = '收盘(日)')
plt.plot(range(len(y)),y,c='b',label='NAV')
plt.show()
print(x,'\n',y)
# 定义线性回归模型函数

# 使用最小二乘法估计参数
params, covariance = curve_fit(linear_model, x, y)
# 提取估计的模型参数
plt.scatter(x, y, label="原始数据")
plt.plot(x, linear_model(x, a_est, b_est), color='red', label="拟合线")
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()

a_est, b_est = params
print('avg_loss:',np.average(linear_model(x, a_est, b_est) - y))
get_like = linear_model(data_fund['收盘价'], a_est, b_est)
plt.plot(range(len(get_like)), get_like,c = 'r',label = 'IOPV')
plt.plot(range(len(get_like)),data_fund['收盘价'],c = 'b',label = '收盘价')
plt.legend()
plt.plot()
# 每只基金的折溢价
for i,j in zip(get_like,data_fund['收盘价']):
    cha_values = abs(i-j)
    if cha_values > 0:
        up_down.append(cha_values)
sum_values = np.sum(up_down)
print(f'{code_fund}',sum_values)
end_frame.append([code_fund,sum_values])
# print(fund_fh_em_df)
print(str(fund_fh_em_df[['分红','除息日期']].loc[fund_fh_em_df['基金代码']=='159001']))
# print(fund_fh_em_df['分红'][fund_fh_em_df[fund_fh_em_df['基金代码']=='159972'].index])
# print(fund_fh_em_df['除息日期'][fund_fh_em_df[fund_fh_em_df['基金代码']=='159972'].index])
```

任务一 *Python* 代码分析

目的：进行波动率计算

```
%筛选小规模 etf 及其波动率计算
f=zeros(886,1);
a=zeros(886,1);
Path = 'C:\Users\0\Desktop\建模\大湾区比赛\A\2023A 题-附件\';   % 设置数据存放的文件夹路径
File = dir(fullfile(Path,'*.xlsx'));  % 显示文件夹下所有符合后缀名为.txt 文件的完整信息
FileNames = {File.name}';            % 提取所有文件的文件名，转换为 n 行 1 列

for i=1:1:887
    if(i<=886)
        str = FileNames{i,1}(1:end-8);
        file_name(i) = str2num(str);
    end
end

Length = length(FileNames);    %计算文件夹里 xls 文档的个数
res = zeros(886, 1);  %etf 日均成交金额%
res_change = zeros(886, 1);  %etf 波动率%
for i = 1:886               %批量读取文件的内容并保存
    sum=zeros(59,1);  %初始化每日成交额求和数组%
    sum_change=0;
    xls_data = xlsread(strcat(Path,FileNames{i,1}));
    for j=1:1:14219
       sum(ceil(j/241))=sum(ceil(j/241))+xls_data(j,5);
       sum_change=sum_change+(xls_data(j,3)-xls_data(j,4))/xls_data(j,4);
    end
    num = 0;
    res_change(i)=sum_change/14219;
    for j=1:1:59
      num = num+sum(j);
    end
    num = num / 59;
    if num > 10000000
       f(i)=1;
    end
end

max=0.002722164;
min=0;

for i=1:1:886
    res_change(i)=res_change(i)/(max-min);
end
```

任务一 *Python* 代码分析

目的：收益率计算

```
import os
import numpy as np
import pandas as pd
import efinance as ef
import re
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
import akshare as ak
# fund_fh_em_df = ak.fund_fh_em()
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
path = r'C:\Users\Lenovo\jupyter_notebook\金融建模大赛 A 数据'
result = os.listdir(path)
result
def get_have(name:str):
    name = re.findall("\d+",name)[0]
    return name

def linear_model(x, a, b):
    return a * x + b

def add_and_before(data:list):
    bb = []
    for i in range(len(data)-1):
        bb.append(1-data[i+1]/data[i])
    return np.average(bb)
shouyilv_data = []
for code_index in range(137,len(result)):
    code_fund = get_have(result[code_index])
    data = ef.fund.get_quote_history(code_fund,pz=75)
    public_dates = ef.fund.get_public_dates(code_fund)
    a_stock = 0
    if len(public_dates) > 0:
        kk = ef.fund.get_invest_position(code_fund,dates=public_dates[0])
        for i in kk['股票代码']:
            close_data = ef.stock.get_quote_history(i,klt=5)['收盘']
            shouyilv = add_and_before(np.array(close_data))
```

```
                a_stock += shouyilv
        shouyilv_data.append([code_fund,a_stock])
        print(a_stock)


    list = [0,
    0,
    0,
    0.00011338768203858756,
    0,
    0.00012787141857046079,
    -0.0007109354545330742,
    0.0006487876877387887,
    -0.0005304820067503392,
    -0.002575425965148788,
    0.00028924977694256594,
    0.0006868543237639954,
    -0.001137150851537474,
    0.00011147939143265657,
    0.0005008582869761594,
    0.00031687953151196514,
    0.0003145865517697282,
    -0.00026607976972208046,
    0.00020036814718960225,
    -4.8457798607497515e-05,
    -0.0007137543403747398,
    0.00013094239033617573,
    0.00026861127326897595,
    1.6593448189812312e-05,
    -1.6233730333220812e-06,
    0.00014706234115766423,
    0.00012599884814122836,
    0.00047511147144039254,
    0.00028215943125058795,
    0.00028215943125058785,
    -7.436885416823655e-06,
    0,
    -0.00033694409573548897,
    0,
    0,
    0.0004091363124792197,
    0.0004091363124792197,
    -1.3436958317401787e-05,
    -8.355412130920149e-05,
    0.00045563749118594923,
    -0.00010762133383000782,
```

```
    0.0005475939022841675,
    0.0007810501343099132,
    0.0002600936947539146,
    0.0002167053954983632,
    5.225903076150513e-05,
    0.0005676949109577978,
    -0.0010402816753092487,
    0.0002107896554696986,
    0.0003781813855775476,
    0.0007810501343099132,
    0.0006549929803068365,
    0.0002140420149345404,
    -1.776432687909393e-05,
    -0.0010838926568526829,
    0.0001397285845750725,
    0.0001388441970325369,
    0.0001924291002509660,
    -0.0008462108059885307,
    0.0003794263342369722,
    0.0001924291002509660,
    0.0001924291002509660,
    4.6140575943862686e-05,
    0.0003295770617352385,
    0.0006549929803068365,
    -0.0007485466377642948,
    0.000385477112512542,
    0.0006125747508984542,
    0.0005623478692983445,
    0.0005623478692983445,
    0.0005623478692983445,
    0.0005623478692983445,
    -0.0005509432214930418,
    -0.0005509432214930419,
    -0.0005509432214930418,
    0.0002568926256494569 3,
    0,
    0,
    0,
    0.0005804668474665546,
    0.0002342822240029179 3,
    0.000136975737479328,
    0.0002361541330868795,
    -0.0005509432214930418,
    -0.0001916955863114639 8,
    0.0001029744809201334,
```

```
    0.0001029744809201334,
    0,
    0.000283656998198835,
    0.00012950599753347396,
    -0.0005115028698246538,
    0.000283656998198835,
    0.00012950599753347396,
    0.00013884419703253696,
    0.00018270490436927275,
    0.0005475939022841675,
    0.00013674297693971408,
    0.00030355777725572173,
    -1.0605071429333149e-05,
    -2.2930704901742824e-05,
    -0.0002856331947102119,
    0.00017019349166808018,
    0.0002150293586114123,
    -0.0001930672324398501,
    -7.144179475156631e-05,
    -7.144179475156631e-05,
    0.000283656998198835,
    -6.25303558996775e-06,
    0.00019242910025096607,
    0,
    -0.00011120913544039954,
    0.00013674297693971408,
    0.0005804668474665546,
    -0.00027369916631700365,
    8.042443373006902e-05,
    0,
    0.0006487876877387887,
    9.141391988556555e-05,
    0.0003032672830417881,
    0.00016909575121231476,
    7.583716066254602e-06,
    0.000178711227367606,
    0.0007428272038465101,
    7.940822636091642e-05,
    0.0012339607009587041,
    0.000411872522579792,
    0.0006681388735857933,
    7.335942732732106e-05,
    -0.00018818109328337275,
    -0.00019601466560654132,
    0.0002095098254252619,
```

```
0.0002095098254252619,
0.00025766285239059997,
0.0002342822240029179,
-0.0008638488065898283,
0.0005233200496534337,
6.70759073167143e-05]
gb = []
for i,j in zip(list,result[:138]):
    gb.append([get_have(j),i])
gb = pd.DataFrame(gb)


shouyilv_frame = pd.DataFrame(shouyilv_data)


shouyilv_frame = pd.concat([gb,shouyilv_frame],axis=0)
shouyilv_frame = shouyilv_frame.reset_index(drop=True)
# shouyilv_frame = shouyilv_frame.sort_values(1)
shouyilv_frame.columns = ['ETF 代码','股票利率']
shouyilv_frame.to_excel(r'C:\Users\Lenovo\Desktop\ETF 成分股票利率(已处理).xlsx')
ak.fund_etf_hist_em('516001')
```

任务一 *Python* 代码分析

目的：进行 *Dual Thrust* 交易策略

```
import efinance as ef
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import re


plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号


data_kua = pd.read_excel(r'c:\Users\Lenovo\Desktop\排名代码表.xlsx')
code_names = data_kua['问题 1']
code_data = []
print(code_names)
for i in code_names:
    if i not in code_data:
        code_data.append(str(i))
print(code_data)


db_all_ = []
```

```
    neam_ids = ef.futures.get_futures_base_info()
    hua = 0.01
    def get_name_index(name):
        return neam_ids['行情 ID'][neam_ids[neam_ids['期货名称'] == f'{name}'].index.tolist()[0]]


    def split_to_data(data,name,wheel):
        data_wheel = []
        for i in range(5,len(data)):
            data_wheel.append(np.average(data[name][i-wheel:i-1]))
        return data_wheel
    def find_h_l(data,wheel):
        data_h = []
        data_l = []
        close_h = []
        close_l = []
        for i in range(5, len(data)):
            data_h.append(np.max(data['最高'][i - wheel:i - 1]))
            data_l.append(np.min(data['最低'][i - wheel:i - 1]))
            close_h.append(np.max(data['收盘'][i - wheel:i - 1]))
            close_l.append(np.min(data['收盘'][i - wheel:i - 1]))
        return data_h,data_l,close_h,close_l


    def one_to(data_to_one):
        one_to_data = []
        pif = np.max(data_to_one) - np.min(data_to_one)
        for i in data_to_one:
            one_to_data.append((i - np.min(data_to_one))/pif)
        return one_to_data


    def get_have(name:str)->str:
        name = re.findall("\d+",name)[0]
        return name


    def main(index):
        """
        k1 = 0.5
        k2 = 0.1
        """
        k1 = 1
        k2 = 1
        wheel = 5
        m_time = 30
        id_futures = get_have(code_data[index])
        len_data = 0
```

```
        line_values = 0
        data_do = [[-3]]
        data_base = ef.stock.get_quote_history(id_futures, klt=5,beg='20231010',end='20231024')
        # print(data_base)
        if len(data_base) == 0:
            return
        try:
            data_base.to_excel(f"{str(data_base['日期'][0])[:-6]}.xlsx")
        except:
            pass
        data_day = np.array(data_base['日期'])
        data_base_open = np.array(data_base['开盘'])
        data_base_close = np.array(data_base['收盘'])
        data_base_zdf = np.array(data_base['涨跌幅'])
        benefit = []
        for i in range(len(data_base_close)-1):
            benefit.append(data_base_close[i+1]/data_base_close[i])
        beta = np.cov(benefit)/np.var(benefit)
        yuzhi = (data_base_zdf.max()-data_base_zdf.min())/2
        # print(data_base_zdf)
        # print(data_base.columns)
        # print(data_base)
        data_h, data_l, close_h, close_l = find_h_l(data_base,wheel)
        public_dates = ef.fund.get_public_dates(id_futures)
        kk = ef.fund.get_invest_position(id_futures, dates=public_dates[0])
        try:
            bodonglv = ((np.sum(kk['持仓占比']*kk['较上期变化'],axis=0))/10)
        except:
            bodonglv = None
        guo = 0.02
        for i in range(5,len(data_h)+5):
            range_k = max(data_h[i-5]-close_l[i-5],close_h[i-5]-data_l[i-5])
            buy_line = data_base_open[i] + k1*range_k
            sell_line = data_base_open[i] - k2*range_k
            if data_do[-1][0]:
                pass
            if data_base_close[i] > buy_line:
                if data_do[-1][0] != '买涨':
                    # print('买涨',data_base_close[i])
                    data_do.append(['买涨',data_base_close[i],i])
                else:
                    pass
            if data_base_close[i] < sell_line:
                if data_do[-1][0] == '买涨':
                    # print('卖出',data_base_close[i])
```

```
                data_do.append(['卖出', data_base_close[i],i])
            else:
                pass
        try:
            if data_do[-1][0] == '买涨' and data_base_zdf[i] > 0:
                # if data_base_zdf[i] > 0.6:
                #     data_do.append(['卖出', data_base_close[i], i])
                pass
            elif data_do[-1][0] == '买涨' and data_base_zdf[i] < 0:
                if data_base_zdf[i] < -yuzhi:
                    data_do.append(['卖出', data_base_close[i], i])
            elif data_do[-1][0] != '买涨' and data_base_zdf[i] < 0:
                # if data_base_zdf[i] < -0.6:
                #     data_do.append(['买涨', data_base_close[i], i])
                pass
            elif data_do[-1][0] != '买涨' and data_base_zdf[i] > 0:
                if data_base_zdf[i] > yuzhi:
                    data_do.append(['买涨', data_base_close[i], i])
        except:
            pass
    data_do_frame = pd.DataFrame(data_do[1:],columns=['方式','收盘价','时间'])
    data_do_frame.to_excel(rf'C:\Users\Lenovo\Desktop\A 金融大湾区\回测数据\第一题\{id_futures}.xlsx')
    # plt.subplot(2,1,1)
    plt.plot(range(len(data_base_close)),data_base_close)
    win_number = 0
    a = []
    b = []
    for x,y,tp in zip(data_do_frame['时间'],data_do_frame['收盘价'],data_do_frame['方式']):
        color = str
        if tp != '买涨':
            color = 'g'
            a = plt.scatter(x, y, c=color)
        else:
            color = 'r'
            b = plt.scatter(x, y, c=color)


    plt.legend((a, b), ('卖出', '买入'), loc='best')
    # plt.subplot(2, 1, 2)
    # plt.plot(range(len(data_base['涨跌幅'])),one_to(data_base['涨跌幅']))
    plt.title(f'{id_futures}')
    plt.savefig(rf'C:\Users\Lenovo\Desktop\A 金融大湾区\回测数据\第一题\{id_futures}.png')
    plt.show()
    win_arr = []
    lose_arr = []
    data_do = data_do[1:]
```

```
        for i in range(len(data_do)-1):
            if data_do[i][0] == '买涨':
                income = (data_do[i + 1][1]*(1-hua) - data_do[i][1])/data_do[i][1]
                if income >= 0:
                    win_arr.append(income)
                else:
                    lose_arr.append(income)
            else:
                income = (data_do[i][1] - data_do[i + 1][1]*(1-hua))/data_do[i][1]
                if income >= 0:
                    win_arr.append(income)
                else:
                    lose_arr.append(income)
        try:
            shouyilv = np.sum(win_arr)
            max_lose = np.min(lose_arr)
        except:
            shouyilv = None
            max_lose = None
        print('收益率',shouyilv)
        print('最大回撤',max_lose)
        print('Beta',beta)
        for i in data_do:
            i[2] = data_day[i[2]]
        # print(pd.DataFrame(data_do,columns=['行为','买入价','日期']))
        # win_arr.extend(lose_arr)
        try:
            xpl = (shouyilv - guo)/bodonglv
        except:
            xpl = None
        print('夏普率:',xpl)
        print(f'k1:{k1},k2:{k2}')
        print('--------------------------------------------------')
        db_all_.append([data_do,xpl,shouyilv,beta])
    for index in range(len(code_data)):
        main(index)
    print(db_all_)
```

任务一 *Matlab* 代码分析

目的：进行归一化和等级划分

```
% 归一化和等级划分制度
%f=xlsread('C:\Users\0\Desktop\建模\大湾区比赛\A\基本数据.xlsx','D2:F887')
f=xlsread('C:\Users\0\Desktop\建模\大湾区比赛\A\收益率(网格交易法+Dual_Thrust).xlsx','C2:E101');
```

```
%归一化
% max=0.754160032;
% min=-1.065529402;
% for i=1:100
%     f(i,3)=(f(i,3)-min)/(max-min);
% end


for i=1:100
    a=abs(f(i,1));
    b=abs(f(i,2));
    c=abs(f(i,3));
    if f(i,1) < 0.2
        a=1;
    end
    if f(i,1) >= 0.2 && f(i,1) < 0.4
       a=2;
    end
    if f(i,1) >= 0.4 && f(i,1) < 0.6
       a=3;
    end
    if f(i,1) >= 0.6 && f(i,1) < 0.8
       a=4;
    end
    if f(i,1) >= 0.8
       a=5;
    end

    if f(i,2) < 0.2
        b=1;
    end
    if f(i,2) >= 0.2 && f(i,2) < 0.4
       b=2;
    end
    if f(i,2) >= 0.4 && f(i,2) < 0.6
       b=3;
    end
    if f(i,2) >= 0.6 && f(i,2) < 0.8
       b=4;
    end
    if f(i,2) >= 0.8
       b=5;
    end

    if f(i,3) < 0.2
```

```
        c=1;
    end
    if f(i,3) >= 0.2 && f(i,3) < 0.4
       c=2;
    end
    if f(i,3) >= 0.4 && f(i,3) < 0.6
       c=3;
    end
    if f(i,3) >= 0.6 && f(i,3) < 0.8
       c=4;
    end
    if f(i,3) >= 0.8 && f(i,3) <1
       c=5;
    end
    if f(i,3) >=1
       c=6;
    end
    f(i,2)=f(i,2)+f(i,3)-f(i,1);
    f(i,1)=b+c-a;

end
```

2.任务二程序代码：

任务二代码分析

目的：预测模型对比

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from keras.layers import LSTM
from keras.layers import Dense
from keras.models import Sequential
from sklearn.metrics import mean_squared_error
import numpy as np
import math
import efinance as ef
from math import sqrt
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import re

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
```

```
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
# import tensorflow
from keras import backend as K
data_kua = pd.read_excel(r'c:\Users\Lenovo\Desktop\跨境 etf 代码表.xlsx')
code_names = data_kua['thscode']
code_data = []
print(code_names)
for i in code_names:
    if i not in code_data:
        code_data.append(i)
print(code_data)
def get_have(name:str)->str:
    name = re.findall("\d+",name)[0]
    return name
import efinance as ef
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号


db_all_ = []


neam_ids = ef.futures.get_futures_base_info()
hua = 0.01
def get_name_index(name):
    return neam_ids['行情 ID'][neam_ids[neam_ids['期货名称'] == f'{name}'].index.tolist()[0]]


def split_to_data(data,name,wheel):
    data_wheel = []
    for i in range(5,len(data)):
        data_wheel.append(np.average(data[name][i-wheel:i-1]))
    return data_wheel
def find_h_l(data,wheel):
    data_h = []
    data_l = []
    close_h = []
    close_l = []
    for i in range(5, len(data)):
        data_h.append(np.max(data['最高'][i - wheel:i - 1]))
        data_l.append(np.min(data['最低'][i - wheel:i - 1]))
        close_h.append(np.max(data['收盘'][i - wheel:i - 1]))
        close_l.append(np.min(data['收盘'][i - wheel:i - 1]))
```

```
        return data_h,data_l,close_h,close_l


    def one_to(data_to_one):
        one_to_data = []
        pif = np.max(data_to_one) - np.min(data_to_one)
        for i in data_to_one:
            one_to_data.append((i - np.min(data_to_one))/pif)
        return one_to_data


    def main(index):
        """
        k1 = 0.5
        k2 = 0.1
        """
        k1 = 1
        k2 = 1
        wheel = 5
        m_time = 30
        id_futures = get_have(code_data[index])
        len_data = 0
        line_values = 0
        data_do = [[-3]]
        data_base = ef.stock.get_quote_history(id_futures, klt=5,beg='20231010',end='20231024')
        # print(data_base)
        if len(data_base) == 0:
            return
        try:
            data_base.to_excel(f"{str(data_base['日期'][0])[:-6]}.xlsx")
        except:
            pass
        data_day = np.array(data_base['日期'])
        data_base_open = np.array(data_base['开盘'])
        data_base_close = np.array(data_base['收盘'])
        data_base_zdf = np.array(data_base['涨跌幅'])
        benefit = []
        for i in range(len(data_base_close)-1):
            benefit.append(data_base_close[i+1]/data_base_close[i])
        beta = np.cov(benefit)/np.var(benefit)
        yuzhi = (data_base_zdf.max()-data_base_zdf.min())/2
        # print(data_base_zdf)
        # print(data_base.columns)
        # print(data_base)
        data_h, data_l, close_h, close_l = find_h_l(data_base,wheel)
        public_dates = ef.fund.get_public_dates(id_futures)
        kk = ef.fund.get_invest_position(id_futures, dates=public_dates[0])
```

```
        try:
            bodonglv = ((np.sum(kk['持仓占比']*kk['较上期变化'],axis=0))/10)
        except:
            bodonglv = None
        guo = 0.02
        for i in range(5,len(data_h)+5):
            range_k = max(data_h[i-5]-close_l[i-5],close_h[i-5]-data_l[i-5])
            buy_line = data_base_open[i] + k1*range_k
            sell_line = data_base_open[i] - k2*range_k
            if data_do[-1][0]:
                pass
            if data_base_close[i] > buy_line:
                if data_do[-1][0] != '买涨':
                    # print('买涨',data_base_close[i])
                    data_do.append(['买涨',data_base_close[i],i])
                else:
                    pass
            if data_base_close[i] < sell_line:
                if data_do[-1][0] == '买涨':
                    # print('卖出',data_base_close[i])
                    data_do.append(['卖出', data_base_close[i],i])
                else:
                    pass
            try:
                if data_do[-1][0] == '买涨' and data_base_zdf[i] > 0:
                    # if data_base_zdf[i] > 0.6:
                    #     data_do.append(['卖出', data_base_close[i], i])
                    pass
                elif data_do[-1][0] == '买涨' and data_base_zdf[i] < 0:
                    if data_base_zdf[i] < -yuzhi:
                        data_do.append(['卖出', data_base_close[i], i])
                elif data_do[-1][0] != '买涨' and data_base_zdf[i] < 0:
                    # if data_base_zdf[i] < -0.6:
                    #     data_do.append(['买涨', data_base_close[i], i])
                    pass
                elif data_do[-1][0] != '买涨' and data_base_zdf[i] > 0:
                    if data_base_zdf[i] > yuzhi:
                        data_do.append(['买涨', data_base_close[i], i])
            except:
                pass
        data_do_frame = pd.DataFrame(data_do[1:],columns=['方式','收盘价','时间'])
        # plt.subplot(2,1,1)
        # plt.plot(range(len(data_base_close)),data_base_close)
        win_number = 0
        # for x,y,tp in zip(data_do_frame['时间'],data_do_frame['收盘价'],data_do_frame['方式']):
```

```
#     if tp == '买涨':
#         plt.scatter(x,y,c = 'r')
#     if tp == '卖出':
#         plt.scatter(x,y,c = 'g')


# plt.subplot(2, 1, 2)
# plt.plot(range(len(data_base['涨跌幅'])),one_to(data_base['涨跌幅']))
# plt.show()
win_arr = []
lose_arr = []
data_do = data_do[1:]
for i in range(len(data_do)-1):
    if data_do[i][0] == '买涨':
        income = (data_do[i + 1][1]*(1-hua) - data_do[i][1])/data_do[i][1]
        if income >= 0:
            win_arr.append(income)
        else:
            lose_arr.append(income)
    else:
        income = (data_do[i][1] - data_do[i + 1][1]*(1-hua))/data_do[i][1]
        if income >= 0:
            win_arr.append(income)
        else:
            lose_arr.append(income)
try:
    shouyilv = np.sum(win_arr)
    max_lose = np.min(lose_arr)
except:
    shouyilv = None
    max_lose = None
print('收益率',shouyilv)
print('最大回撤',max_lose)
print('Beta',beta)
for i in data_do:
    i[2] = data_day[i[2]]
# print(pd.DataFrame(data_do,columns=['行为','买入价','日期']))
# win_arr.extend(lose_arr)
try:
    xpl = (shouyilv - guo)/bodonglv
except:
    xpl = None
print('夏普率:',xpl)
print(f'k1:{k1},k2:{k2}')
print('--------------------------------------------------')
db_all_.append([data_do,xpl,shouyilv,beta])
```

```
for index in range(len(code_data)):
    main(index)
print(db_all_)
db_fram = pd.DataFrame(db_all_,columns = ['行为','夏普率','收益率','Beta'])
db_fram = db_fram[['夏普率','收益率','Beta']]
db_fram = pd.concat([db_fram,code_names],axis=1)
db_fram.to_excel(r"c:\Users\Lenovo\Desktop\第二题.xlsx")
def difference(data_set,interval=1):
    diff=list()
    for i in range(interval,len(data_set)):
        value=data_set[i]-data_set[i-interval]
        diff.append(value)
    return pd.Series(diff)

def timeseries_to_supervised(data,lag=1):
    df=pd.DataFrame(data)
    columns=[df.shift(1)]
    columns.append(df)
    df=pd.concat(columns,axis=1)
    df.fillna(0,inplace=True)
    return df
def scale(train,test):
    # 创建一个缩放器，将数据集中的数据缩放到[-1,1]的取值范围中
    scaler=MinMaxScaler(feature_range=(-1,1))
    # 使用数据来训练缩放器
    scaler=scaler.fit(train)
    # 使用缩放器来将训练集和测试集进行缩放
    train_scaled=scaler.transform(train)
    test_scaled=scaler.transform(test)
    return scaler,train_scaled,test_scaled
def fit_lstm(train,batch_size,nb_epoch,neurons):
    # 将数据对中的 x 和 y 分开
    X,y=train[:,0:-1],train[:,-1]
    # 将 2D 数据拼接成 3D 数据，形状为[N*1*1]
    X=X.reshape(X.shape[0],1,X.shape[1])

    model=Sequential()
    model.add(LSTM(neurons,batch_input_shape=(batch_size,X.shape[1],X.shape[2]),stateful=True))
    model.add(Dense(1))

    model.compile(loss='mean_squared_error',optimizer='adam')
    for i in range(nb_epoch):
        # shuffle 是不混淆数据顺序
        his=model.fit(X,y,batch_size=batch_size,verbose=1,shuffle=False)
        # 每训练完一次就重置一次网络状态，网络状态与网络权重不同
```

```
            model.reset_states()
        return model


    def forecast_lstm(model,batch_size,X):
        # 将形状为[1:]的，包含一个元素的一维数组 X，转换形状为[1,1,1]的 3D 张量
        X = X.astype('float64')
        X=X[0].reshape(1,1,len(X))
        # 输出形状为 1 行一列的二维数组 yhat
        yhat=model.predict(X,batch_size=batch_size)
        # 将 yhat 中的结果返回
        return yhat[0,0]


    # 对预测的数据进行逆差分转换
    def invert_difference(history,yhat,interval=1):
        return yhat+history[-interval]


    # 将预测值进行逆缩放，使用之前训练好的缩放器，x 为一维数组，y 为实数
    def invert_scale(scaler,X,y):
        # 将 X,y 转换为一个 list 列表
        new_row=[x for x in X]+[y]
        # 将列表转换为数组
        array=np.array(new_row,dtype=object)
        # 将数组重构成一个形状为[1,2]的二维数组->[[10,12]]
        array=array.reshape(1,len(array))
        # 逆缩放输入的形状为[1,2]，输出形状也是如此
        invert=scaler.inverse_transform(array)
        # 只需要返回 y 值即可
        return invert[0,-1]
    from keras import backend as keras_export
    for k in range(len(code_data)):
        code_fund = get_have(code_data[k])
        close_data = ef.stock.get_quote_history(code_fund,klt=5)
        # close_data.to_excel(rf'c:\Users\Lenovo\Desktop\跨境 etf\{close_data["股票名称"][0]}.xlsx')
        close_data = close_data[['日期','收盘']]
        series = close_data.set_index(['日期'],drop=True)
        plt.figure(figsize=(10,6))
        series['收盘'].plot()
        plt.show()
        raw_value=series.values
        diff_value=difference(raw_value,1)
        print(diff_value)
        supervised=timeseries_to_supervised(diff_value,1)
        supervised_value=supervised.values
        print(supervised_value)
        testNum=400
```

```
        train,test=supervised_value[:-testNum],supervised_value[-testNum:]
        scaler,train_scaled,test_scaled=scale(train,test)
        X,y=train[:,0:-1],train[:,-1]
        X=X.reshape(X.shape[0],1,X.shape[1])
        # 构建一个 LSTM 模型并训练，样本数为 1，训练次数为 3，LSTM 层神经元个数为 4
        lstm_model=fit_lstm(train_scaled,1,3,4)
        # 取出测试集中的一条数据，并将其拆分为 X 和 y
        # X,y=test[i,0:-1],test[i,-1]
        # # 将训练好的模型、测试数据传入预测函数中
        # yhat=forecast_lstm(lstm_model,1,X)
        # # 将预测值进行逆缩放
        # yhat=invert_scale(scaler,X,yhat)
        # # 对预测的 y 值进行逆差分
        # yhat=invert_difference(raw_value,yhat,len(test_scaled)+1-i)
        predictions=list()
        for i in range(len(test_scaled)):
            # 将测试集拆分为 X 和 y
            X,y=test[i,0:-1],test[i,-1]
            # 将训练好的模型、测试数据传入预测函数中
            yhat=forecast_lstm(lstm_model,1,X)
            # 将预测值进行逆缩放
            yhat=invert_scale(scaler,X,yhat)
            # 对预测的 y 值进行逆差分
            yhat=invert_difference(raw_value,yhat,len(test_scaled)+1-i)
            # 存储正在预测的 y 值
            predictions.append(yhat)
        plt.plot(raw_value[-testNum:])
        plt.plot(predictions)
        plt.legend(['true','pred'])
        plt.show()
        y_test = raw_value[-testNum:]
        y_predict = predictions
        print("mean_absolute_error:", mean_absolute_error(y_test, y_predict))
        print("mean_squared_error:", mean_squared_error(y_test, y_predict))
        print("rmse:", sqrt(mean_squared_error(y_test, y_predict)))
        print("r2 score:", r2_score(y_test, y_predict))
        break
    from numpy import array
    from keras.models import Sequential
    from keras.layers import Dense

    # 构造监督学习型数据
    def split_sequence(sequence, n_steps):
        X, y = list(), list()
        for i in range(len(sequence)):
```

```
            # 获取待预测数据的位置
            end_ix = i + n_steps
            # 如果待预测数据超过序列长度，构造完成
            if end_ix >= len(sequence):
                break
            # 分别汇总 输入 和 输出 数据集
            seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
            X.append(seq_x)
            y.append(seq_y)
        return array(X), array(y)


    # define input sequence
    code_fund = get_have(code_data[0])
    raw_seq = ef.stock.get_quote_history(code_fund,klt=5)['收盘']
    # 定义时间步，即用几个数据作为输入，预测另一个数据
    n_steps = 3
    # 构造监督学习型数据
    X, y = split_sequence(raw_seq[:-150], n_steps)
    print(X)
    print(y)
    # 定义一个简单的 MLP 模型
    model = Sequential()
    model.add(Dense(100, activation='relu', input_dim=n_steps))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    # fit model
    model.fit(X, y, epochs=1000, verbose=0)


    # 手动定义一个输入
    x_input = array(raw_seq[-150:])
    dd = len(x_input)//n_steps
    # 将(3,)的序列[70, 80, 90]，重构成(1, 3)的序列[[70, 80, 90]]
    x_input = x_input.reshape((dd, n_steps))
    yhat = model.predict(x_input, verbose=0)


    sum = []
    for i,j in zip(yhat,raw_seq):
        sum.append(i-j)
    print(np.average(sum))
    plt.plot(range(len(yhat)),yhat)
    plt.plot(range(len(raw_seq[int(-150/n_steps):])),raw_seq[int(-150/n_steps):])
    plt.show()


    y_test = raw_seq[int(-150/n_steps):]
```

```
y_predict = yhat
print("mean_absolute_error:", mean_absolute_error(y_test, y_predict))
print("mean_squared_error:", mean_squared_error(y_test, y_predict))
print("rmse:", sqrt(mean_squared_error(y_test, y_predict)))
print("r2 score:", r2_score(y_test, y_predict))
```

3. 任务三程序代码：

任务三 *Python* 代码分析

目的：进行拟合

```
import efinance as ef
import pandas as pd
import re

def get_have(name:str):
    name = re.findall("\d+",name)[0]
    return name
guzhi_futures = ['上证 50','沪深 300','中证 500','中证 1000']
dd = pd.DataFrame()
for i in guzhi_futures:
    stock_members = ef.stock.get_members(i)
    dd = pd.concat([dd,stock_members],axis=0)

data_kua = pd.read_excel(r'c:\Users\Lenovo\Desktop\跨境 etf 代码表.xlsx')
code_names = data_kua['thscode']
code_data = []
print(code_names)
for i in code_names:
    if i not in code_data:
        code_data.append(i)
print(code_data)
futures_name = ef.futures.get_futures_base_info()
A_stock = ef.stock.get_realtime_quotes()
print(dd) #股指
# print(futures_name) #期货
print(A_stock) #A 股

common_rows = dd[dd['股票代码'].isin(A_stock['股票代码'])]
print(common_rows)
data_all = pd.DataFrame()
def have_a_dataframe(names):
    global data_all
```

```
        for id_index in names:
            get_stock_data = pd.DataFrame(ef.stock.get_quote_history(get_have(id_index), klt=5,beg =
'20231001',end='20231030')['收盘'])
            get_stock_data.rename(columns={'收盘':id_index},inplace=True)
            data_all = pd.concat([data_all,get_stock_data],axis=1)
    have_a_dataframe(code_names)
    print(data_all)
    have_a_dataframe(common_rows['股票代码'])
    print(data_all)


    print(data_all)
    import numpy as np

    data_all = data_all.loc[:, ~(data_all.isnull().all())]
    data_all.columns
    result2 = np.corrcoef(data_all, rowvar=False)
    data_like = pd.DataFrame(result2,index=data_all.columns,columns=data_all.columns).iloc[:99,99:]
    data_like_columns = data_like.columns.tolist()
    data_like_index = data_like.index.tolist()
    print(data_like_index)

    wlh = []
    for j in range(len(data_like_index)):
        kk = [[data_like_index[j]]]
        for i in range(len(data_like_columns)):
            like_ = data_like.iloc[j,i]
            if like_ >= 0.9:
                kk.append(data_like_columns[i])
        wlh.append(kk)
    print(wlh)

    import matplotlib.pyplot as plt
    for j in range(len(wlh)):
        # plt.plot(range(len(data_all[wlh[j][0]])),data_all[wlh[j][0]])
        for i in wlh[j][1:]:
            plt.title(wlh[j][0][0])
            plt.plot(range(len(data_all[i])),data_all[i],label = i)
        plt.legend()
        plt.show()
    from sklearn.linear_model import LinearRegression as LR  # 线性回归
    from sklearn.model_selection import train_test_split     # 划分训练测试集
    import pandas as pd

    index = 0
    x = np.array(range(len(csgo_main))).reshape(-1, 1)
```

```
for i in range(48,len(data_all.iloc[:,0])+1,48):
    csgo = data_all.iloc[index:i,:]
    index = i
    try:
        for name_index in range(len(wlh)):
            csgo_main = csgo[wlh[name_index][0][0]]
            reg = LR().fit(x,csgo_main)
            R_2 = reg.score(x,csgo_main)
            print(R_2)
            for g in range(len(wlh[name_index][1:])):
                csgo_js = csgo[wlh[name_index][g]]
                reg = LR().fit(x,csgo_main)
                R_2 = reg.score(x,csgo_main)
                print(R_2)
    except:
        pass
```

任务三 *Python* 代码分析

目的：回测代码

```
import efinance as ef
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import re

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

data_kua = pd.read_excel(r'c:\Users\Lenovo\Desktop\跨境 etf 代码表.xlsx')
code_names = data_kua['thscode']
code_data = []
shouyilvlist = []
print(code_names)
for i in code_names:
    if i not in code_data:
        code_data.append(i)
print(code_data)

db_all_ = []

neam_ids = ef.futures.get_futures_base_info()
hua = 0.01
```

```
def get_name_index(name):
    return neam_ids['行情 ID'][neam_ids[neam_ids['期货名称'] == f'{name}'].index.tolist()[0]]


def split_to_data(data,name,wheel):
    data_wheel = []
    for i in range(5,len(data)):
        data_wheel.append(np.average(data[name][i-wheel:i-1]))
    return data_wheel
def find_h_l(data,wheel):
    data_h = []
    data_l = []
    close_h = []
    close_l = []
    for i in range(5, len(data)):
        data_h.append(np.max(data['最高'][i - wheel:i - 1]))
        data_l.append(np.min(data['最低'][i - wheel:i - 1]))
        close_h.append(np.max(data['收盘'][i - wheel:i - 1]))
        close_l.append(np.min(data['收盘'][i - wheel:i - 1]))
    return data_h,data_l,close_h,close_l


def one_to(data_to_one):
    one_to_data = []
    pif = np.max(data_to_one) - np.min(data_to_one)
    for i in data_to_one:
        one_to_data.append((i - np.min(data_to_one))/pif)
    return one_to_data


def get_have(name:str)->str:
    name = re.findall("\d+",name)[0]
    return name


def wlh(address,grid):
    if address[0] != grid:
        return True
    else:
        return False


def get_shouyilv(data_do:pd.DataFrame):
    data = np.array(data_do[1])
    index_list = data_do[0][data_do[0]=='卖出'].index.tolist()
    index_list.insert(0,0)
    sum_ = 0
    for i in range(len(index_list)-1):
        end = data[index_list[i+1]]
```

```
            k = 1
            if i == 0:
                k = 0
            for j in range(index_list[i]+k,index_list[i+1]):
                sum_ += (data[j]-end)/data[j]
        # print(sum_)
        return sum_
    def main(index):
        """
        k1 = 0.5
        k2 = 0.1
        """
        cc = []
        k1 = 1
        k2 = 1
        wheel = 5
        id_futures = str(513130)
        guo = 0.02
        ####网格交易参数
        data_do = [[-3]]
        address = [0,0]
        get_buy_data = 0
        last_ = 0
        max_volume = 15
        data_base = ef.stock.get_quote_history(id_futures, klt=5)
        if len(data_base) == 0:
            return
        # data_day = np.array(data_base['日期'])
        data_base_open = np.array(data_base['开盘'])
        data_base_close = np.array(data_base['收盘'])
        # data_base_zdf = np.array(data_base['涨跌幅'])
        benefit = []
        for i in range(len(data_base_close)-1):
            benefit.append(data_base_close[i+1]/data_base_close[i])
        beta = np.cov(benefit)/np.var(benefit)
        # yuzhi = (data_base_zdf.max()-data_base_zdf.min())/2
        # print(data_base_zdf)
        # print(data_base.columns)
        # print(data_base)
        data_h, data_l, close_h, close_l = find_h_l(data_base,wheel)
        public_dates = ef.fund.get_public_dates(id_futures)
        kk = ef.fund.get_invest_position(id_futures, dates=public_dates[0])
        try:
            bodonglv = ((np.sum(kk['持仓占比']*kk['较上期变化'],axis=0))/10)
        except:
```

```
        bodonglv = None
    # grid = pd.cut([bar.close], context.band, labels=[1, 2, 3, 4, 5, 6, 7, 8])[0]
    for i in range(5,len(data_h)+5):
        before = i-1
        center = data_base_close[before]
        wg = np.array([0.92, 0.94, 0.96, 0.98, 1, 1.02, 1.04, 1.06, 1.08]) * center
        grid = pd.cut([data_base_close[i]], wg, labels=[1, 2, 3, 4, 5, 6, 7, 8])[0]
        range_k = max(data_h[i-5]-close_l[i-5],close_h[i-5]-data_l[i-5])
        buy_line = data_base_open[i] + k1*range_k
        sell_line = data_base_open[i] - k2*range_k
        if get_buy_data >= 8:
            data_do.append(['卖出', data_base_close[i], i])
            get_buy_data = 0
        if last_ < grid and wlh(address,grid):
            data_do.append(['买涨', data_base_close[i], i])
            get_buy_data += 1
        address = [last_, grid]
        last_ = grid
        if data_base_close[i] > buy_line:
            # if data_do[-1][0] != '买涨':
            data_do.append(['买涨',data_base_close[i],i])
            get_buy_data += 1
        if data_base_close[i] < sell_line:
            if data_do[-1][0] == '买涨':
                # print('卖出',data_base_close[i])
                data_do.append(['卖出', data_base_close[i],i])
                get_buy_data = 0


        # try:
        #     if data_do[-1][0] == '买涨' and data_base_zdf[i] > 0:
        #         # if data_base_zdf[i] > 0.6:
        #         #     data_do.append(['卖出', data_base_close[i], i])
        #         pass
        #     elif data_do[-1][0] == '买涨' and data_base_zdf[i] < 0:
        #         if data_base_zdf[i] < -yuzhi:
        #             data_do.append(['卖出', data_base_close[i], i])
        #     elif data_do[-1][0] != '买涨' and data_base_zdf[i] < 0:
        #         # if data_base_zdf[i] < -0.6:
        #         #     data_do.append(['买涨', data_base_close[i], i])
        #         pass
        #     elif data_do[-1][0] != '买涨' and data_base_zdf[i] > 0:
        #         if data_base_zdf[i] > yuzhi:
        #             data_do.append(['买涨', data_base_close[i], i])
        # except:
        #     pass
```

```
            if i == len(data_h)+4:
                data_do.append(['卖出', data_base_close[i], i])
                get_buy_data = 0
        data_do = data_do[1:]
        plt.plot(range(len(data_base_close)),data_base_close)
        a = []
        b = []
        for d_dat in data_do:
            color = str
            if d_dat[0] != '买涨':
                color = 'g'
                a = plt.scatter(d_dat[2], d_dat[1], c=color)
            else:
                color = 'r'
                b = plt.scatter(d_dat[2],d_dat[1],c = color)


        plt.legend((a, b), ('卖出', '买入'), loc='best')
        plt.show()
        data_do = pd.DataFrame(data_do)
        print(data_do)
        # data_do.to_excel(rf'C:\Users\Lenovo\Desktop\A 金融大湾区\回测数据\第二题\{id_futures}.xlsx')
        # print(data_do)
        shouyilv = get_shouyilv(data_do)
        try:
            xpl = (shouyilv - guo)/bodonglv
        except:
            xpl = None
        print(f'{id_futures}收益率:',shouyilv)


        shouyilvlist.append([id_futures,xpl,beta,shouyilv])



        # data_do_frame = pd.DataFrame(data_do[1:],columns=['方式','收盘价','时间'])
        # plt.subplot(2,1,1)
        # win_number = 0
        # for x,y,tp in zip(data_do_frame['时间'],data_do_frame['收盘价'],data_do_frame['方式']):
        #     if tp == '买涨':
        #         plt.scatter(x,y,c = 'r')
        #     if tp == '卖出':
        #         plt.scatter(x,y,c = 'g')


        # plt.subplot(2, 1, 2)
        # plt.plot(range(len(data_base['涨跌幅'])),one_to(data_base['涨跌幅']))
        # plt.show()
```

```
        # win_arr = []
        # lose_arr = []
        # data_do = data_do[1:]
        # print('夏普率:',xpl)
        # print(f'k1:{k1},k2:{k2}')
        # print('--------------------------------------------------')
        # db_all_.append([data_do,xpl,shouyilv,beta])
    for index in range(1,19):
        to_get_index = pd.read_excel(fr'C:\Users\Lenovo\Desktop\A 金融大湾区\回测数据\第三题\第{index}天得分表.xlsx')
        to_get_index_code = np.array(to_get_index.sort_values('得分')[-5:]['ETF 代码'])
        to_get_score = np.array(to_get_index.sort_values('得分')[-5:]['得分'])
        for i,j in zip(to_get_index_code,to_get_score):
            main(i)
        shouyilvlist.append(['----','----','----','----'])
        end_data = pd.DataFrame(shouyilvlist, columns=['代码', '夏普率', 'Beta', '收益率'])
        end_data.to_excel(r'C:\Users\Lenovo\Desktop\A 金融大湾区\回测数据\第三题\收益率(网格交易法+Dual_Thrust).xlsx', index_label=None)
```