



华南理工大学

South China University of Technology

《机器学习实训》期末报告

School: 华南理工大学

Major: 人工智能

Student Name : 刘彦志 王实依 刘兴琰 韦华泰 韩诚

Teacher: 徐进 刘琦

Submission Date: 2023.1.21

1. 项目简介

1.1 项目介绍

项目简介：机器学习在恒生指数日线数据和 5 分钟级数据上的应用

本项目旨在利用机器学习方法，通过对恒生指数的日线数据和 5 分钟级数据进行深入分析和建模，实现指数增强策略，从而获取超额收益。项目时间跨度为 2020 年 11 月 17 日至 2023 年 11 月 17 日，初始资金为 1000000 元。

1. 数据预处理与因子挖掘：

对提供的日线数据和 5 分钟级数据进行预处理，包括缺失值处理、数据标准化等。查阅相关资料，挖掘与恒生指数相关的因子，例如技术指标、市场情绪指标等。

2. 日线数据的机器学习模型：

构建机器学习模型，包括 LSTM 模型和 Transformer 模型，用于对日线数据进行预测。设计训练集、测试集和验证集，并通过这些数据进行模型训练和评估。针对预测周期，实现日级、周级和月级的指数预测。

3. 5 分钟级数据的模型与交易策略：

利用 5 分钟级数据构建模型，寻找日内最佳买卖时机信号。

结合日线预测结果与日内买卖信号，编写具体交易策略，以实现指数增强和获取超额收益。

4. 控制最大回撤与可视化展示：

确保最终策略实现指数增强，并将最大回撤控制在合理范围内。

利用可视化工具展示投资结果，包括策略收益曲线、交易信号与实际交易情况等。

5. 算法与模型选择：

采用 LSTM 模型和 Transformer 模型处理日线数据，以捕捉时序关系和长期依赖。

使用马尔可夫决策过程模型处理 5 分钟级数据，优化日内交易决策。

构建多因子选股模型，综合考虑多个因子对投资组合的影响。

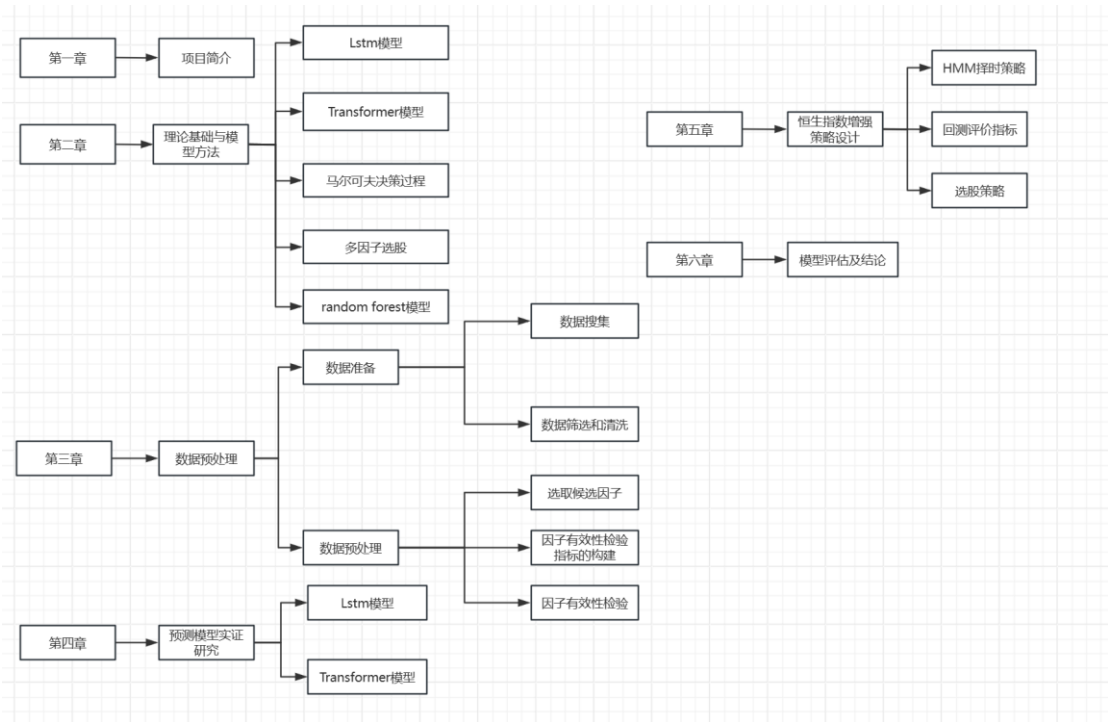
6. 课程报告与效果评估：

撰写课程报告，明确表明采用的算法、模型原理、训练过程、特征构建方法以及最终效果的验证。

对构建的模型和应用的策略进行优缺点分析，包括模型的稳定性、策略的实用性等。

通过以上步骤，我们旨在为恒生指数的投资提供一种基于机器学习的全面增强策略，同时通过对模型和策略的深入分析，为投资决策提供更多有力的支持。

1.2 流程框架图



1.3 职责分配

队员	负责任务
刘彦志	编写与训练基于 transformer 的时间滑窗股票预测模型，编写与训练基于随机森林的动态选股模型
刘兴琰	数据预处理与可视化，相关因子挖掘，编写与训练隐马尔可夫股票择时策略模型，完成回测数据评测指标的计算与策略评估可视化
王实依	编写 LSTM 代码，训练 LSTM 模型，使用 LSTM 模型预测股票和恒生指数趋势
韩诚	数据采集，编写报告
韦华泰	数据采集，编写报告

2. 理论基础与模型方法

2.1 LSTM 模型

LSTM (Long Short-Term Memory) 模型是一种特殊的循环神经网络 (RNN)，主要用于处理和预测序列数据的任务。LSTM 的关键优势在于它能够记住长期依赖关系，解决了传统 RNN 在长序列学习时出现的梯度消失问题。

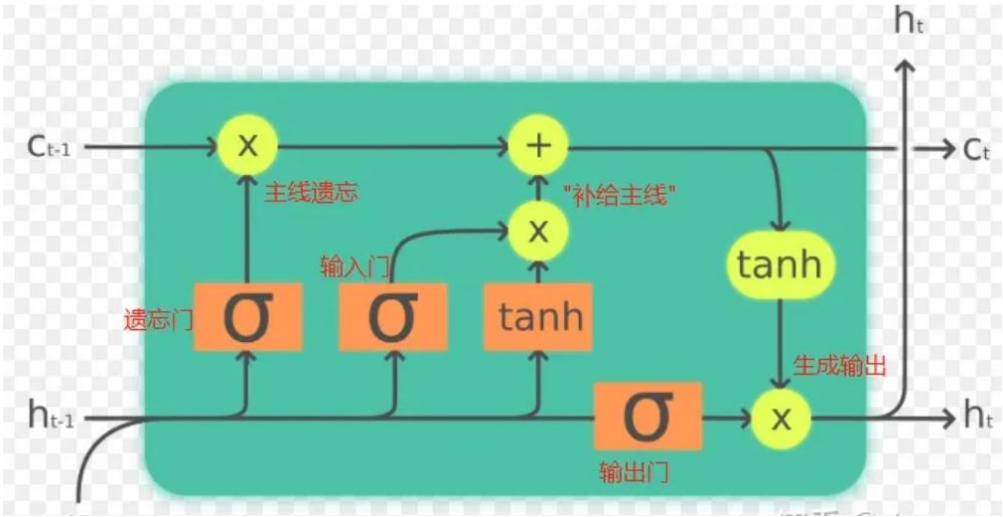


图 2.1 LSTM 核心组件

LSTM 的核心组件

遗忘门 (Forget Gate)：决定从单元状态中丢弃什么信息。它通过一个 Sigmoid 层来查看 h_t （上一时间步的输出）和 x_t （当前时间步的输入），输出一个在 0 到 1 之间的数字给每个在单元状态 $C_{(t-1)}$ 中的数。这个数字会被用来乘以单元状态，因此如果输出是 0，它就完全“忘记”了这个特征。

输入门 (Input Gate)：更新单元状态。首先，一个 Sigmoid 层决定哪些值将要更新，然后一个 \tanh 层创建一个新的候选值向量 C_t ，这可能会加到状态中。我们将这两个信息相乘，决定更新状态的程度。

单元状态 (Cell State)：LSTM 的核心，通过线路在整个链上运行，仅有一些线性交互，信息流动几乎不受阻碍。单元状态有能力删除或添加信息，由遗忘门和输入门共同控制。

输出门 (Output Gate)：决定下一个隐藏状态。隐藏状态包含了前一个时间步的信息，也用于预测。一个 Sigmoid 层决定单元状态的哪个部分将输出，然后将单元状态通过 \tanh （获取一个值在 -1 到 1 之间）并将其乘以 Sigmoid 门的输出，以决定最终输出。

LSTM 的工作原理

数据流：输入进入系统后，会经过这些门结构的处理。每个门对数据进行某种形式的变换，如遗忘、更新或输出。

时间步迭代：这个过程在每个时间步重复进行，每一步都会更新单元状态和隐藏状态。

2.2 Transformer 模型

Transformer 模型是一种深度学习模型，主要用于处理序列到序列的任务，如机器翻译、文本生成等。其核心特点是完全依赖于注意力机制（attention mechanisms），不像之前的模型那样依赖于循环神经网络（RNN）或卷积神经网络（CNN）。

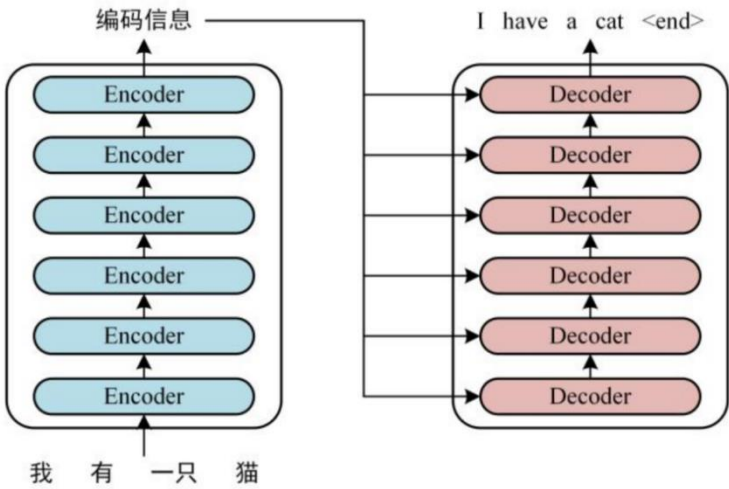


图 2.2-1 Transformer 核心组件

Transformer 的主要组件

编码器（Encoder）：Transformer 模型包含多个编码器层。每个编码器层包括两个主要子层：多头注意力（multi-head self-attention）机制和简单的、位置全连接的前馈网络。编码器处理输入序列，并将信息传递到解码器。

解码器（Decoder）：解码器也由多个层组成，每层包含三个子层：解码器自身的多头注意力机制、用于读取编码器输出的多头注意力机制，以及一个前馈网络。解码器根据编码器的输出和自身的前一输出来生成下一个输出。

多头注意力机制：通过将注意力分散到序列的不同位置，多头注意力机制能够同时学习来自不同位置的信息。这对于理解和表示序列中的复杂关系非常有用。

位置编码 (Positional Encoding)：由于 Transformer 模型不使用循环神经网络，因此需要另一种方法来理解序列中词汇的顺序。位置编码通过给每个词汇增加位置信息来解决这个问题。

Transformer 工作原理

数据流：输入数据首先进入编码器，编码器对数据进行处理并传递给解码器。解码器根据编码器的输出和自己之前的输出生成最终的输出序列。

注意力机制：注意力机制允许模型在生成输出时“关注”输入序列中的不同部分，这是 Transformer 的核心。

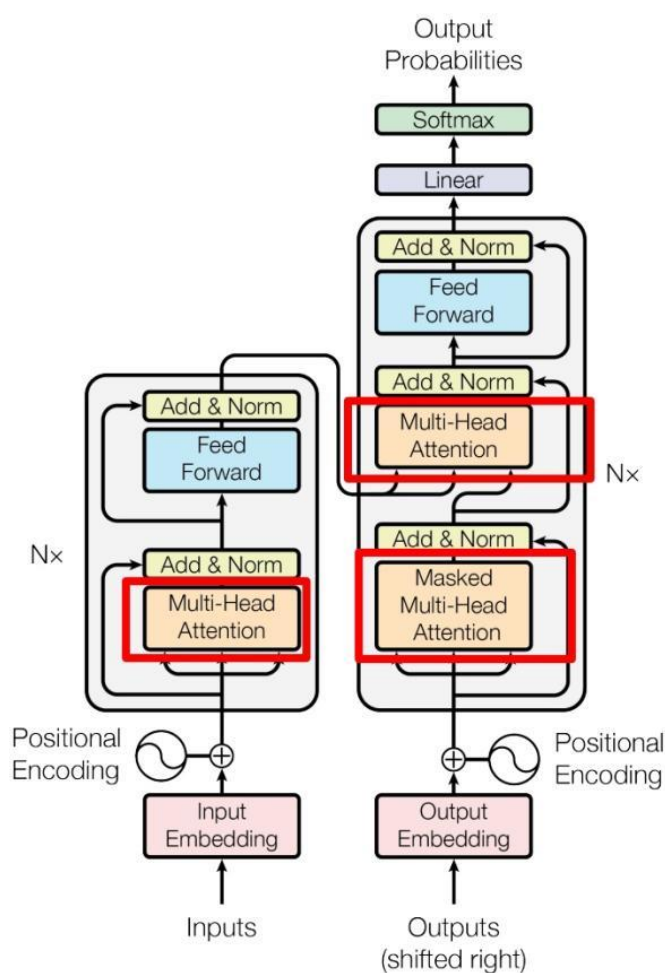


图 2.2-2 Transformer 工作流程图

2.3 随机森林

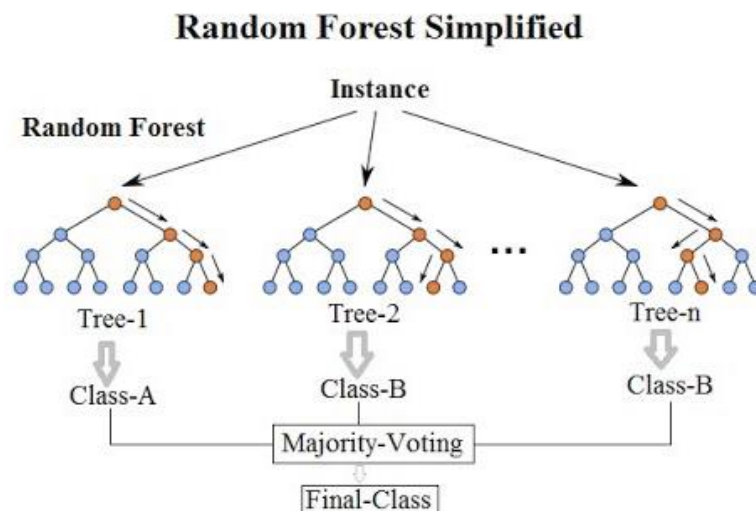


图 2.3 随机森林工作原理

随机森林的核心组件

决策树 (Decision Trees)： 随机森林由多个决策树组成，是其基本组件。每棵决策树都是一个独立的分类器，负责对数据进行划分和预测。决策树通过对特征进行逐步的二分来制定决策规则。

Bagging (自助聚集)： 随机森林使用 Bagging 技术，即自助聚集，通过有放回地从训练数据集中抽取多个不同的子集。每个决策树都是在一个独立的子集上训练的，这有助于增加模型的多样性。

随机特征选择： 在构建每个决策树的过程中，随机森林引入了随机性，即在每个节点的分裂过程中，只考虑特征的随机子集而不是全部特征。这确保了每棵树都在不同的特征子集上做出决策，增加了整体模型的多样性。

投票或平均机制： 随机森林的预测结果是由所有决策树的结果进行投票（分类问题）或平均（回归问题）得到的。这种投票或平均机制有助于提高模型的准确性和鲁棒性。

随机性： 随机森林通过引入随机性，包括随机选择训练集样本和随机选择特征子集，减少了单个模型的过拟合风险，提高了整体模型的泛化能力。

随机森林的工作原理

- **Bagging (自助聚集)：** 对于给定的训练数据集，随机森林通过有放回地从数据集中随机抽样生成多个不同的子集，这些子集将用于独立地训练每棵决策树。

- 并行构建决策树： 随机森林中的每棵决策树都在一个子集上进行构建。这意味着每个树都是在不同的数据集子集上训练的，以增加模型的多样性。
- 随机特征选择： 在构建每个决策树的过程中，对于每个节点的分裂，只考虑特征的随机子集，而不是全部特征。这样可以确保每棵树都在不同的特征子集上做出决策，增加了模型的多样性。
- 投票或平均： 在进行预测时，随机森林中的每棵树都对结果进行预测。对于分类问题，通过投票选择获得最终的预测类别；对于回归问题，通过取平均值获得最终的预测结果。这种投票或平均机制有助于提高模型的鲁棒性和准确性。
- 减少过拟合： 由于随机森林中的每棵树都是在不同的数据子集上训练的，同时引入了随机特征选择，模型更加抵抗过拟合。即使某些树在训练集上表现过于复杂（过拟合），整体模型的预测结果仍然具有鲁棒性

2.4 马尔可夫决策过程

马尔可夫决策过程（Markov Decision Process，简称 MDP）是一种数学框架，用于描述在不确定性环境中的决策制定。MDP 在许多领域都有应用，特别是在强化学习中，用于模拟智能体（agent）与环境的交互过程。

MDP:

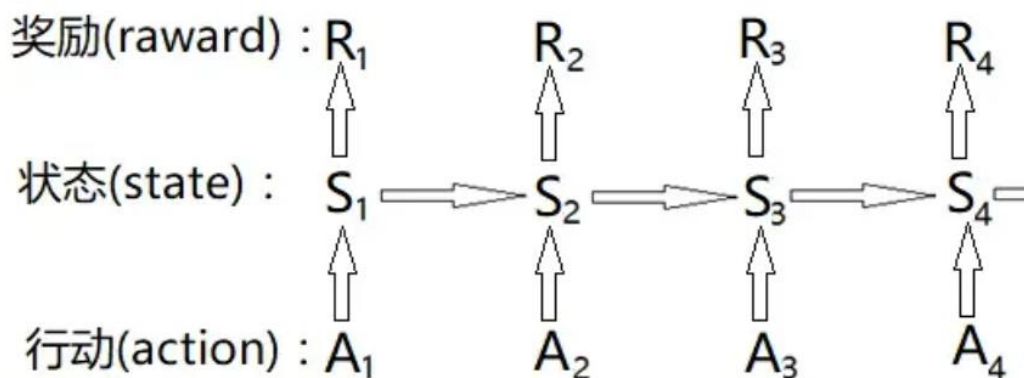


图 2.4 马尔可夫决策过程工作流程图

MDP 的基本组成

状态 (States)： 状态集合表示所有可能的环境或系统状态。在 MDP 中，每个决策点的状态都应满足马尔可夫性质，即未来的状态仅依赖于当前状态，而与过去的状态或行为无关。

行为 (Actions)：行为集合包括智能体可执行的所有行为。在给定的状态下，智能体选择某个行为来影响环境。

转移概率 (Transition Probabilities)：转移概率是从当前状态通过某个行为转移到另一个状态的概率。这些概率是 MDP 的核心，定义了环境的动态特性。

奖励函数 (Reward Function)：奖励函数定义了智能体在到达特定状态或执行特定行为后获得的即时奖励。智能体的目标通常是最大化累积奖励。

策略 (Policy)：策略是一个从状态到行为的映射，定义了智能体在每个状态下应选择哪个行为。策略可以是确定性的，也可以是随机性的。

折扣因子 (Discount Factor)：折扣因子用于调整未来奖励的当前价值。它是一个介于 0 和 1 之间的数，其中较低的值意味着更重视即时奖励，较高的值则表示更重视长期奖励。

MDP 的工作原理

- 智能体观察当前环境的状态。
- 基于策略，智能体在当前状态下选择一个行为。
- 行为影响环境，导致状态转移，环境提供给智能体一个即时奖励。
- 智能体的目标是通过选择合适的行为来最大化总奖励，这可以通过学习最优策略来实现。

2.5 多因子选股模型

多因子选股是一种投资策略，它综合考虑多种金融指标或“因子”来选择股票。这种方法基于的理念是，不同的因子可以从不同角度反映公司的财务状况、市场表现和潜在风险，通过组合这些因子可以构建一个更加稳健和多元化的投资组合。

主要的股票选取因子

价值因子：通常通过市盈率 (PE)、市净率 (PB)、股息收益率等指标来衡量。价值因子旨在寻找市场上被低估的股票。

动量因子：基于过去一段时间内股票价格的变动趋势。动量策略认为股价上涨的股票将继续上涨，而下跌的股票将继续下跌。

规模因子：依据公司的市值大小来分类。小盘股通常被认为具有更高的增长潜力，但也伴随更高的风险。

质量因子：关注公司的质量，包括盈利能力、经营效率、财务稳健性等。高质量的公司通常具有强大的盈利能力和稳定的财务状况。

波动率因子：基于股票历史上的价格波动性。低波动性股票通常认为风险较小。

成长因子：关注公司的收入增长、利润增长等指标。成长型公司通常能够提供更高的回报率，但也可能伴随较高的风险。

多因子选股策略的步骤



图 2.5 多因子选股模型构建工作流程图

因子选择与定义：根据投资者的风险偏好和市场状况选择适当的因子，并对因子进行明确的量化定义。

数据收集与处理：收集相关的财务数据、市场数据等，并进行清洗和处理。

因子测试与分析：对选定的因子进行历史数据测试，分析其对股票收益的预测能力。

构建模型：根据因子分析的结果，构建多因子选股模型。这通常涉及到统计和机器学习技术。

投资组合优化：根据模型输出进行股票选择，构建投资组合。在这个过程中，还需要考虑到分散化投资的原则，以降低风险。

回测与调整：通过历史数据对构建的模型进行回测，验证策略的有效性。根据市场变化和模型表现进行适时调整。

优点与局限

优点：多因子选股能够综合多个维度的信息，有助于发现被市场忽视的投资机会，同时可以分散特定因子或市场的风险。

局限：多因子选股依赖于历史数据和统计模型，可能存在过拟合的风险。此外，市场环境的快速变化可能导致某些因子的有效性下降。

3. 数据预处理

3.1 数据准备

3.1.1 数据搜集

恒生指数成分股日线数据：我们网上搜寻并收集了 1994 年之后的恒生指数成分股的日线数据。这些数据包括股票的开盘价、最高价、最低价、收盘价以及成交量和成交额等。

恒生指数成分股 5 分钟线数据：我们还搜集了 2019 年之后恒生指数成分股的 5 分钟线数据。这些数据提供了更高频率的市场信息，有助于我们进行更细致的市场动态分析。

3.1.2 数据筛选和清洗

数据筛选：在课程提供的到的恒生指数日线数据和 5 分钟级数据中，我们发现 2009 年之前的数据在成交额和成交量方面存在问题，即这些年份的数据显示为 0。考虑到这样的数据可能会影响我们模型的准确性和可靠性，我们决定将这部分数据从我们的分析中排除。

数据清洗：对于保留的数据，我们进行了彻底的清洗，包括检查数据的完整性、去除异常值、处理缺失值等，以确保数据的准确性和一致性。

3.2 数据预处理

3.2.1 选取候选因子

多因子模型的准确性和可靠性取决于如何选择最佳的因子，这些因子的选择对于构建有效的机器学习模型至关重要。由于市场上存在大量的基础因子，而因子的有效性经常会根据群体的理解的不同而有所不同，因此，要想在众多的因子中找到最适合个别投资者的交易策略，就必须根据群体的理解程度进行精心挑选，当前并未形成完全统一的因子分类。

在这篇文章里，我们通过使用 Tushare 平台收集的 85 个候选因子，这些因子主要包括六个类别，分别为估值类、质量类、成长类、风险类、情绪类和技术类。这些因素可以帮助我们更好地分析企业的财务信息，并预测它们的未来业绩。从行情数据中获取的风险、情感以及技术信息可以以明显的方式反映出市场的变化，如价格变化、成交量变化、投资者的偏好变化等。

因子库：

风险情绪类因子（21 个）：最近 3、6、12 个月收益率，最近 1、3、6、12 个月换手率乘以收益率均值，20、60、120、240 天指数加权换手率乘以收

益率，个股 60 个月收益与上证总之回归的 截距项，最近 20、60、120、240 天收益率波动率，换手率，自由流通股换手率， 涨跌幅，量比，个股 60 个月收益与上证总之回归的系数。

估值与市值类因子（6 个）：市盈率，市销率，市净率，总市值取对数，总市值，流通市值。

成长与质量类因子（11 个）：利润总额比营业收入，年化总资产净利率，营业总收入同比增长率，营业收入同比 增长率，利润总额同比增长率，营业利润同比增长率，应收账款周转率，流动资产 周转率，固定资产周转率，总资产周转率，有形资产比负债合计。

盈利因子与现金流 因子（20 个）：基本每股收益，稀释每股收益，营业总收入，营业总成本，营业利润，营业外收 入，营业外支出，利润总额，净利润，销售商品、提供劳务收到的现金，经营活动 现金流入与流出，购买商品、接受劳务支付的现金，支付给职工以及为职工支付的 现金，投资活动现金流入与流出，投资活动产生的现金流量净额，筹资活动现金流 入与流出，筹资活动产生的现金流量净额。

杠杆与规模因子（12 个）：流动比率，速动比率，经营活动产生的现金流量净额比流动负债，产权比率，流动 资产合计，非流动资产合计，资产总计，流动负债合计，非流动负债合计，总负债 合计，总股本，流通股本。

技术类指标（15 个）：平均线指标（macd、dea、dif、trix），相对强弱指标（RSI），心理线指标（psy）， 乖离率（bias），威廉指数（wr6、wr10），顺势指标（CCI），能量潮（obv）、随机指 标（KDJ），抛物线转向指标（sar），人气指标（ar、br）

因子有效性检验指标（3 个）：利用：信息系数（IC），信息比率 IR，因子间相关系数。

3.2.2 因子有效性检验指标的构建

因子可以被视为一种数学表达式，其实质是一种信息传递，它汇集了大量的投资信息，从而影响着投资决策的结果。当所选的因子与其对应股票的收益率有着很大的相关性，则我们认为该因子是有效的。一般来说，判断因子是否有效，我们主要采用以下几个指标来进行衡量：

(1) 信息系数 (IC)

IC 值反映了当前股票的因子暴露水平与未来收益率之间的关联程度，通过 IC 值的计算可以评估出该因子对于股票预测能力。IC 的计算方法如下：

$$Normal\ IC = corr(f_{t-1} - r_t)$$

其中 f_{t-1} 为 t-1 期股票的因子值； r_t 为 t 期的股票收益率。一般来说，IC 的绝对值越大，说明该因子越有效，选股能力就越强；IC 的绝对值大于 0.02 的因子具备较强的选股能力；IC 的值是 0 或者说接近于 0，则代表该因子对于股票没有任何的预测能力。除此之外，如果 IC 的值在回测期间的波动性较小，则说明该因子有效性越稳定。

(2) 信息比率 IR

信息比率 IR 是一个衡量组合超额收益的指标，其含义是单位主动风险所带来的超额收益 IR 的计算公式如下：

$$IR = \frac{P_r - P_r}{TE}$$

(3) 因子间相关系数

通过将多个因子按照相关性分组，可以有效地减少模型中的共线性，从而减少因子冗余，从而提升模型的解释能力和预测准确性。本文通过皮尔森相关系数来计算变量之间的线性相关程度，具体的数学表达式如下：

$$r(X,Y) = \frac{cov(X,Y)}{\sqrt{Var[X]Var[Y]}}$$

其中 $r(X,Y)$ 表示两个变量之间的相关系数， $Cov(X,Y)$ 表示两个变量之间的协方差， $Var[X]$ 表示的是 X 的方差， $Var[Y]$ 表示的是 Y 的方差。r 的值处在 [-1,1] 之间， $r=0$ 代表两变量之间的相关性为 0，两变量完全无关；r 的绝对值等于 1，则代表两变量之间完全相关。

3.2.3 因子有效性检验

(1) 相关性检验

本文选择的财务因子较多，一般来说，财务因子不仅存在滞后性问题，还存在着很大的相关性。为了使操作方便，我们先进行相关性过滤，根据变量之间的相关

系数公式计算了恒生指数成分股因子之间的相关性，其中，相关性矩阵的部分截图如下：

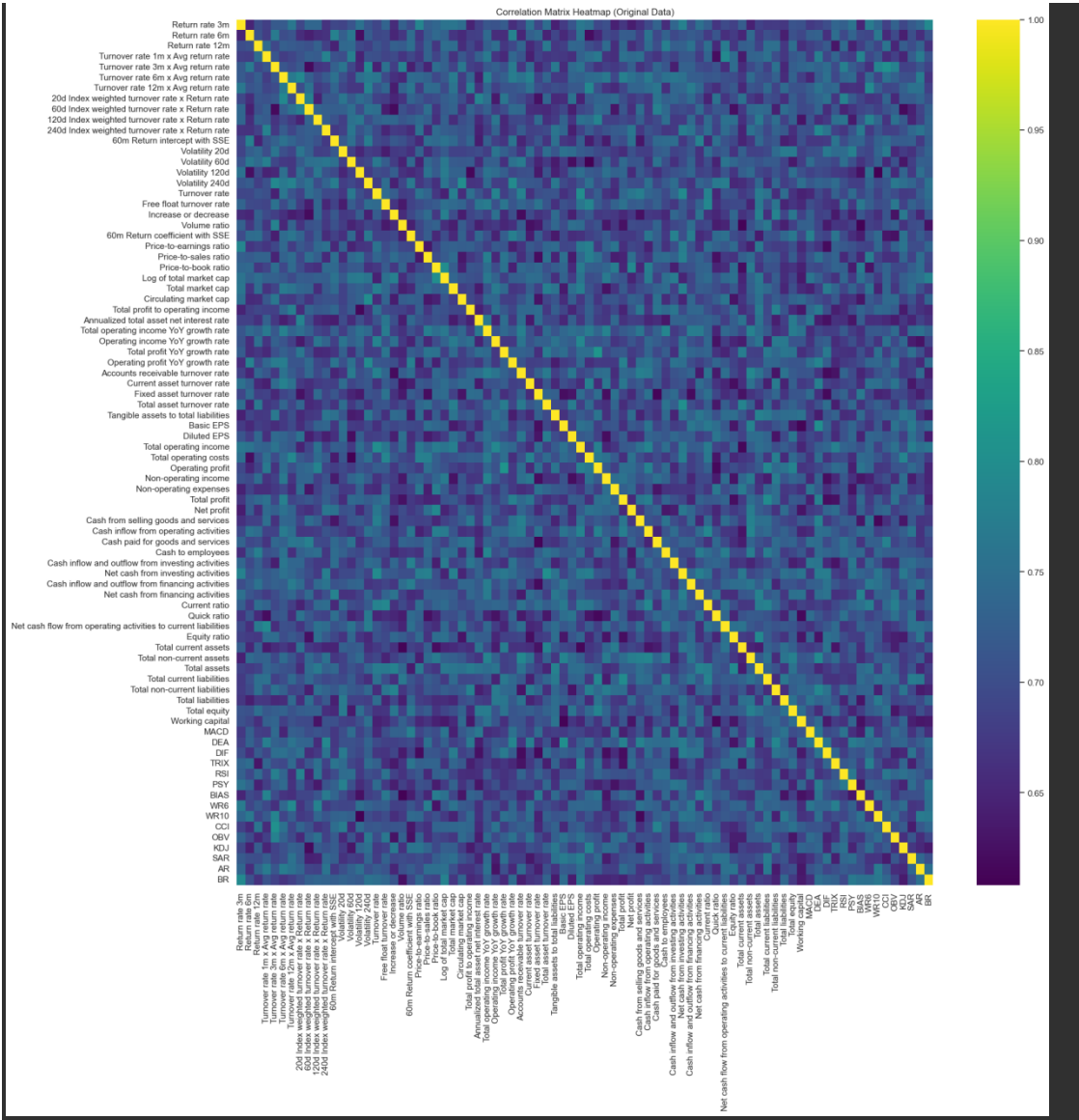


图 3.2 因子相关性（剔除因子后）

（2）IC、IR 值检验

根据 IC 和 IR 的测量，我们可以看到它们的相关系数。通常来讲，IC 的相关系数超过 0.02 就意味着这个因子是可靠的。因此，我们决定把 IC 相关系数低于 0.02 的因子排除在外。IC 绝对值大于 0.02 的因子如下表：

s	lc	lr
basic_eps	5.67	0.33
diluted_eps	3.22	0.33
pb	-2.07	-0.19
pe	-2.11	-0.19
ps	-2.14	-0.19

circ_mv	-2.13	-0.19
total_mv_log	-3.13	-0.19
tangibleasset_to_deb	2.83	0.38
n_income	2.01	0.38
assets_turn	2.23	0.25
macd	-3.1	-0.31
wr10	2.84	0.26
rsi	-2.01	-0.18
bias	-2.42	-0.21
cci	-2.48	-0.23
kdj_j	-2.84	-0.26
ar	-2.01	-0.22
br	-2.04	-0.22
sar	-3.21	-0.31
obv	-3.81	-0.36

从表中可以看到，IC 值检验之后，有 20 个因子通过因子有效性检验，其中，未通过检验的因子基本为财务因子。

4. 预测模型实证研究

4.1 Lstm 模型预测

数据准备

数据读取：从文本文件中读取预处理后的开盘价数据，清洗数据，包括去除非日期格式的条目，将日期列转换为日期类型，并按时间顺序排序。

数据转换及归一化：将文本数据转换为数值数据，使用 NumPy 数组将数据转换为 PyTorch 的张量，这些张量用于模型的输入和输出。同时进行数据归一化确保输入数据的稳定性和一致性。

序列生成：根据 LSTM 模型的要求，准备时序数据，其中使用前一时间点的数据来预测下一时间点的值。

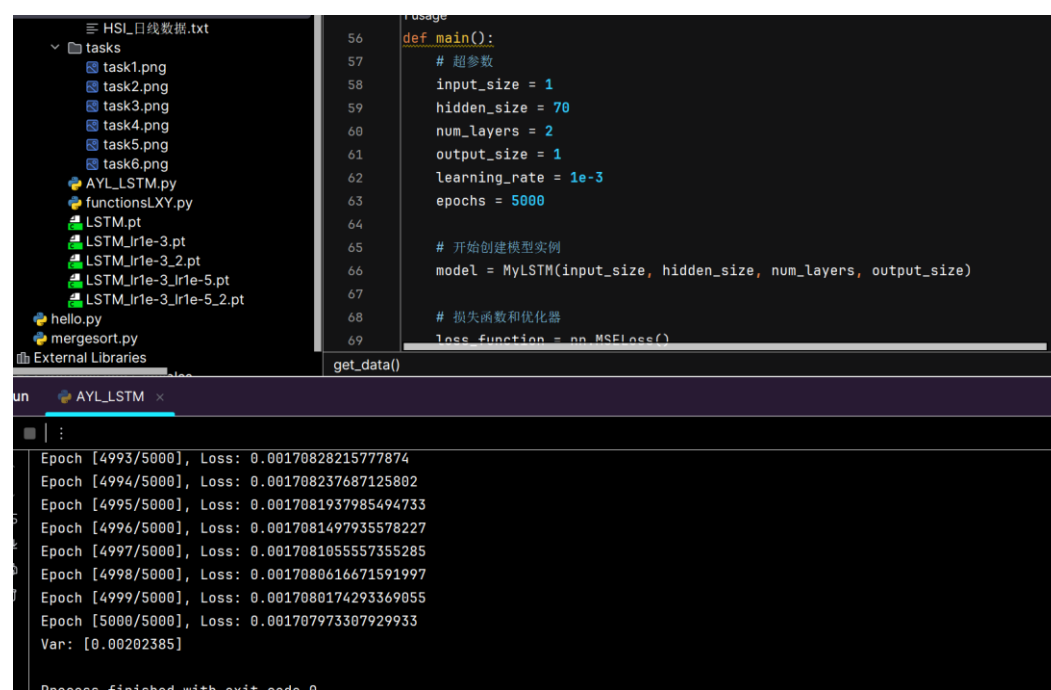
模型定义

模型结构：定义了一个包含 LSTM 层和全连接层的神经网络模型。LSTM 层用于处理时间序列数据，全连接层用于输出预测结果。

超参数设置：设定了 LSTM 模型的输入大小、隐藏层大小、层数、输出大小、学习率和训练周期数。

训练和验证

使用上述 LSTM 模型对恒生指数及恒生指数其中成分股 00001 长和 日线 开盘价的价格进行训练并预测的结果。



```
usage
56 def main():
57     # 超参数
58     input_size = 1
59     hidden_size = 70
60     num_layers = 2
61     output_size = 1
62     learning_rate = 1e-3
63     epochs = 5000
64
65     # 开始创建模型实例
66     model = MyLSTM(input_size, hidden_size, num_layers, output_size)
67
68     # 损失函数和优化器
69     loss_function = nn.MSELoss()
70
71     get_data()
72
73     # 训练
74     for epoch in range(epochs):
75         # 训练数据
76         train_data, val_data = get_data()
77         # 训练模型
78         model.train()
79         # 计算训练数据的损失
80         train_loss = loss_function(model(train_data))
81         # 计算验证数据的损失
82         val_loss = loss_function(model(val_data))
83         # 打印损失
84         print('Epoch [%d/%d], Loss: %f' % (epoch + 1, epochs, train_loss))
85
86     # 保存模型
87     torch.save(model.state_dict(), 'LSTM.pt')
88
89     # 加载模型
90     model.load_state_dict(torch.load('LSTM.pt'))
91
92     # 测试
93     test_data = get_data()
94     test_loss = loss_function(model(test_data))
95     print('Test Loss: %f' % test_loss)
96
97     # 结束
98     print('Process finished with exit code 0')
```

Epoch [4993/5000], Loss: 0.00170828215777874
Epoch [4994/5000], Loss: 0.001708237687125802
Epoch [4995/5000], Loss: 0.0017081937985494733
Epoch [4996/5000], Loss: 0.0017081497935578227
Epoch [4997/5000], Loss: 0.001708105557355285
Epoch [4998/5000], Loss: 0.0017080616671591997
Epoch [4999/5000], Loss: 0.0017080174293369055
Epoch [5000/5000], Loss: 0.001707973307929933
Var: [0.00202385]

图 4.1-1 模型超参数及训练结果

模型的超参数设置为：

- 输入大小 (input_size) : 1
- 隐藏层大小 (hidden_size) : 70
- LSTM 层数 (num_layers) : 2
- 输出大小 (output_size) : 1
- 学习率 (learning_rate) : 0.001
- 训练周期 (epochs) : 5000

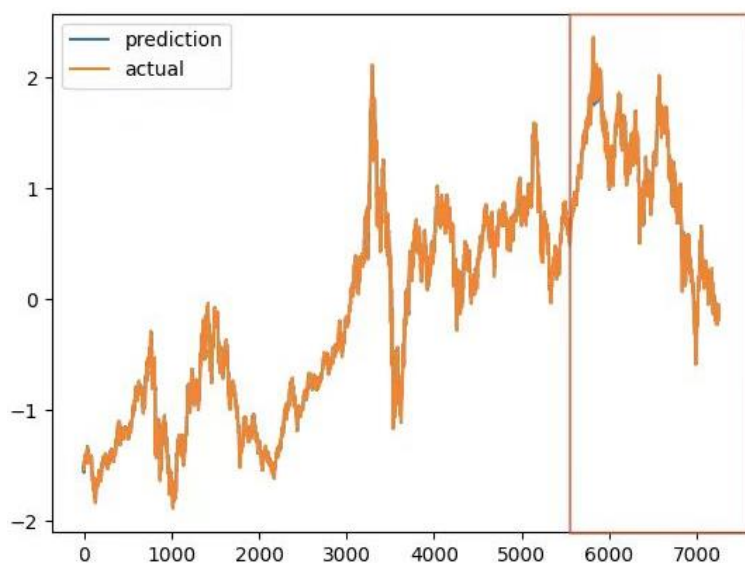


图 4.1-2 恒生指数开盘价预测与验证

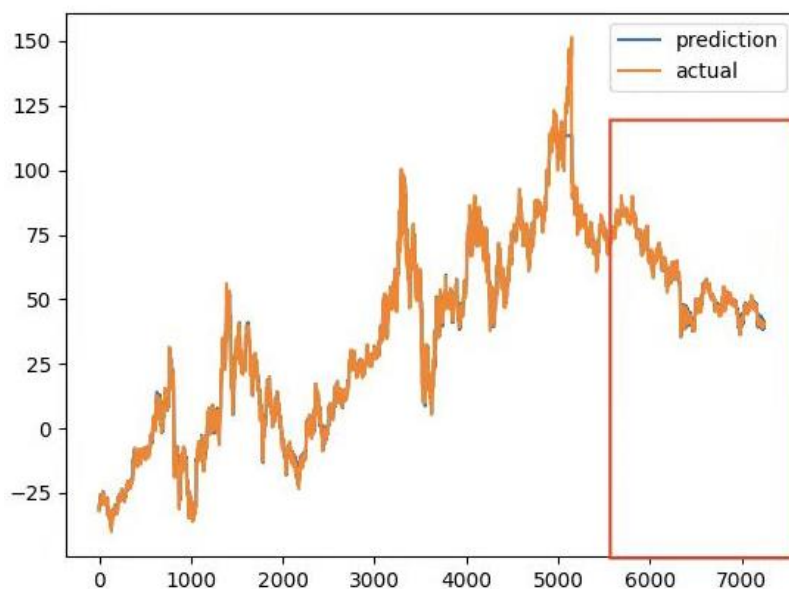


图 4.1-3 0001 长和 日线 开盘价预测与验证

图表显示了模型的预测值（蓝色线）与实际股票价格（橙色线）的对比，其中红框代表测试集，红框前代表训练集，训练数据为前 4/5。

训练集性能：在训练集区间（红框前），预测曲线与实际数据曲线非常接近，这表明模型能够很好地学习到数据集中的模式和趋势。

测试集表现：在测试集区间（红框内），模型的预测仍然紧跟实际数据的趋势。

波动性捕捉：模型能够捕捉到价格的波动性，这在金融时间序列预测中是一个重要的特性。

结论

结果：预测结果（蓝色线）与实际数据（橙色线）非常接近，尤其是在训练集上。在测试集上，尽管存在一些偏差，模型仍能跟随实际数据的趋势，在未见过的测试集上进行了合理的预测。

潜在改进：根据预测与实际曲线之间的偏差，还有进一步优化模型的空间，例如通过调整超参数或使用更复杂的模型结构。

4.2 Transformer 模型预测

数据准备

读取和清洗数据：代码从文件中读取股票市场数据，然后清洗数据，包括去除非日期格式的条目，将日期列转换为日期类型，并按时间顺序排序。

划分数据集：将数据集划分为训练集、验证集和测试集，这是机器学习项目的标准做法，有助于模型的训练和评估。

创建序列：通过 `create_sequences` 函数，数据被处理成序列，这对于时间序列预测是重要的，我们可以通过捕捉时间序列数据中的模式来进行未来值的预测。序列的长度（`seq_length`）设为 5，意味着使用过去 5 天的数据来预测下一天。（即时间序列 X 为 5 天，预测数据 Y 为 5 天后的一天）

```
# 定义窗口长度
seq_length = 5 # 使用过去 5 天的数据来预测下一天

# 划分数据集
train_data = df[:train_size]
val_data = df[train_size:train_size+val_size]
test_data = df[train_size+val_size:]

# 创建训练集、验证集和测试集的 X 和 Y 序列
X_train, y_train = create_sequences(train_data[['Opening', 'Highest', 'Low', 'Closing', 'Volume', 'Trading Value']], seq_length)
X_val, y_val = create_sequences(val_data[['Opening', 'Highest', 'Low', 'Closing', 'Volume', 'Trading Value']], seq_length)
X_test, y_test = create_sequences(test_data[['Opening', 'Highest', 'Low', 'Closing', 'Volume', 'Trading Value']], seq_length)
```

图 4.2-1 时间序列划分

标准化处理：使用 `MinMaxScaler` 对数据进行标准化处理，这有助于加快模型训练的收敛速度，并提高模型性能。

Transformer 模型定义

模型结构：定义了一个 `StockTransformer` 类，它继承自 `torch.nn.Module`。这个类定义了 Transformer 模型的结构，包括编码器层

(nn.TransformerEncoderLayer) 和 Transformer 编码器 (nn.TransformerEncoder)。

输入尺寸和参数：模型接受的输入尺寸 (input_size) 设为 6，对应于 6 个特征（开盘价、最高价、最低价、收盘价、成交量和成交额）。num_layers 和 heads 分别定义了 Transformer 编码器层的数量和多头注意力的头数。

前向传播：forward 方法定义了数据通过模型的流程。数据首先通过编码器层，然后通过一个线性层 (self.decoder) 来生成预测输出。

训练和验证

损失函数和优化器：使用均方误差损失 (nn.MSELoss) 和 Adam 优化器。这是回归任务中常用的损失函数和优化器。

训练循环：进行了多个周期 (epochs) 的训练，每个周期都包括对训练集的遍历和对验证集的评估。在每个批次中，模型的输出与目标值进行比较，计算损失，并进行反向传播以更新模型权重。

验证性能：在每个训练周期后，使用验证集评估模型性能，以监控模型是否过拟合或者是否还在学习。

结果可视化：周期性地将模型的预测结果与实际值进行可视化比较，这有助于直观地评估模型的预测能力。

```
Epoch [966/1000], Loss: 1.2210, Train Accuracy: 0.83, Train Sensitivity: 0.78, Train Specificity: 0.78, Train AUC: 0.96
Epoch [967/1000], Loss: 1.2117, Train Accuracy: 0.82, Train Sensitivity: 0.77, Train Specificity: 0.77, Train AUC: 0.95
Epoch [968/1000], Loss: 1.2134, Train Accuracy: 0.82, Train Sensitivity: 0.77, Train Specificity: 0.77, Train AUC: 0.96
Epoch [969/1000], Loss: 1.1988, Train Accuracy: 0.85, Train Sensitivity: 0.82, Train Specificity: 0.82, Train AUC: 0.96
Epoch [970/1000], Loss: 1.1963, Train Accuracy: 0.81, Train Sensitivity: 0.76, Train Specificity: 0.76, Train AUC: 0.93
Epoch [970/1000], Loss: 1.1963
Epoch [971/1000], Loss: 1.3085, Train Accuracy: 0.83, Train Sensitivity: 0.78, Train Specificity: 0.78, Train AUC: 0.97
Epoch [972/1000], Loss: 1.2184, Train Accuracy: 0.70, Train Sensitivity: 0.66, Train Specificity: 0.66, Train AUC: 0.94
Epoch [973/1000], Loss: 1.4223, Train Accuracy: 0.81, Train Sensitivity: 0.76, Train Specificity: 0.76, Train AUC: 0.93
Epoch [974/1000], Loss: 1.2796, Train Accuracy: 0.81, Train Sensitivity: 0.75, Train Specificity: 0.75, Train AUC: 0.92
Epoch [975/1000], Loss: 1.3158, Train Accuracy: 0.81, Train Sensitivity: 0.76, Train Specificity: 0.76, Train AUC: 0.94
Epoch [976/1000], Loss: 1.2671, Train Accuracy: 0.82, Train Sensitivity: 0.77, Train Specificity: 0.77, Train AUC: 0.95
Epoch [977/1000], Loss: 1.2783, Train Accuracy: 0.81, Train Sensitivity: 0.76, Train Specificity: 0.76, Train AUC: 0.94
Epoch [978/1000], Loss: 1.2308, Train Accuracy: 0.72, Train Sensitivity: 0.68, Train Specificity: 0.68, Train AUC: 0.94
Epoch [979/1000], Loss: 1.2713, Train Accuracy: 0.73, Train Sensitivity: 0.69, Train Specificity: 0.69, Train AUC: 0.94
Epoch [980/1000], Loss: 1.2366, Train Accuracy: 0.81, Train Sensitivity: 0.77, Train Specificity: 0.77, Train AUC: 0.94
Epoch [980/1000], Loss: 1.2366
Epoch [981/1000], Loss: 1.2402, Train Accuracy: 0.82, Train Sensitivity: 0.77, Train Specificity: 0.77, Train AUC: 0.94
Epoch [982/1000], Loss: 1.2322, Train Accuracy: 0.81, Train Sensitivity: 0.76, Train Specificity: 0.76, Train AUC: 0.95
Epoch [983/1000], Loss: 1.2146, Train Accuracy: 0.82, Train Sensitivity: 0.77, Train Specificity: 0.77, Train AUC: 0.92
Epoch [984/1000], Loss: 1.2519, Train Accuracy: 0.82, Train Sensitivity: 0.77, Train Specificity: 0.77, Train AUC: 0.93
```

图 4.2-2 训练过程图

结论

模型在某些训练周期 (epoch) 中显示出相对较高的损失值 (Loss)，并且在不同的训练周期中损失值有波动，意味着模型的收敛效果不够理想。我们可以从以下几个方面来分析这个情况：

1. 损失值 (Loss)：损失值表示模型预测和实际目标值之间的差异。损失值较高通常表示模型在训练数据上的性能不佳。理想情况下，随着训练的进行，损失值应该逐渐减小并稳定下来。
2. 训练准确率 (Train Accuracy)：虽然准确率在一些周期中达到了 0.85 左右，但并没有显示出随训练周期稳定增长的趋势，意味着模型在学习过程中没有稳定地改进。
3. 训练灵敏度 (Train Sensitivity) 和特异性 (Train Specificity)：灵敏度（也称为召回率）和特异性是分类性能的两个重要指标。灵敏度较高时意味着模型在识别正类方面表现良好，而特异性较高则意味着在识别负类方面表现良好。这两个指标也显示了波动，表明模型在不同类别上的表现不均衡。
4. 训练 AUC (Train AUC)：AUC 值衡量模型区分不同类别的能力。AUC 值在 0.92 到 0.97 之间波动，这可能表明模型在不同训练周期的性能不稳定。

总结

经过了多次调参后，我们使用该模型检验的效果不佳，故我们最终选取了 LSTM 对我们的数据进行训练和评测。

5. 恒生指数增强策略设计

5.1 选股策略设计

读取预处理数据

遍历指定文件夹中的所有 .txt 文件，每个文件代表一只股票的历史数据。读取数据后，进行数据清洗，包括转换日期格式、移除无效数据。计算每天的股价变化（当前收盘价与前一天收盘价之差）并创建目标变量 Target（涨为 1，跌为 0）。选择 '开盘'、'最高'、'最低'、'Prev_Close'（前一天的收盘价）、'成交量'、'成交额' 作为特征。

模型训练与预测

训练：使用随机森林分类器训练模型。将数据分为训练集和测试集（70% 训练，30% 测试），并处理缺失值。

预测：在测试集上进行预测，并计算每个预测的概率。计算 F1 分数、AUC 值和准确率，以验证模型效果。

选股设计

我们使用随机森林分类器预测股价上涨或下跌的概率。然后对所有股票的结果进行汇总，提供了一个基于历史数据的股票选择机制。分类器会对于输入的每一天股价数据选出 10 支上涨概率最高的股票以用来进行后面的择时策略。

结果汇总

将每个股票的预测概率和实际测试日期存储在一个字典中。汇总所有股票的平均 F1 分数、AUC 值和准确率。将概率结果合并成一个二维数组，并对每个日期的概率进行排序，找出最高的十个值。

```
[0.67 0.41 0.85 ... 0.6 0.12 0.37]
Model Accuracy: 0.7053445850914205

In [68]: print(predictions)
          print(y_test)

[1 0 1 ... 1 0 0]
4982    1
4450    1
4513    0
4836    0
3800    0
..
4560    1
1649    1
4823    1
1720    0
181     0
Name: Target, Length: 1422, dtype: int64
```

图 5.1 模型准确度及对于一天选出的 10 支上涨概率最高的股票

5.2 HMM 择时策略

数据准备：

从 CSV 文件中读取股票数据。

提取收盘价和成交量数据，计算日收益率。

将收盘价的日收益率和成交量数据进行标准化处理，使不同量级的数据可以在相同的尺度下比较。

HMM 模型训练：

使用 `hmmlearn` 库中的 `GaussianHMM` 类来创建一个高斯隐马尔可夫模型。

设置模型的状态数（本例中为 2，代表市场的两种状态，如上涨和下跌）。

使用标准化后的数据训练 HMM 模型。

状态预测与交易信号生成:

使用训练好的模型预测每个时间点的市场状态。基于预测状态的变化生成交易信号，例如，如果状态从 0 变为 1，代表市场从下跌转为上涨，生成买入信号。

策略回报计算:

通过将交易信号与日收益率相乘来计算策略的回报。

计算策略的累计回报，这可以通过将所有时间点的策略回报连乘得到。

可视化:

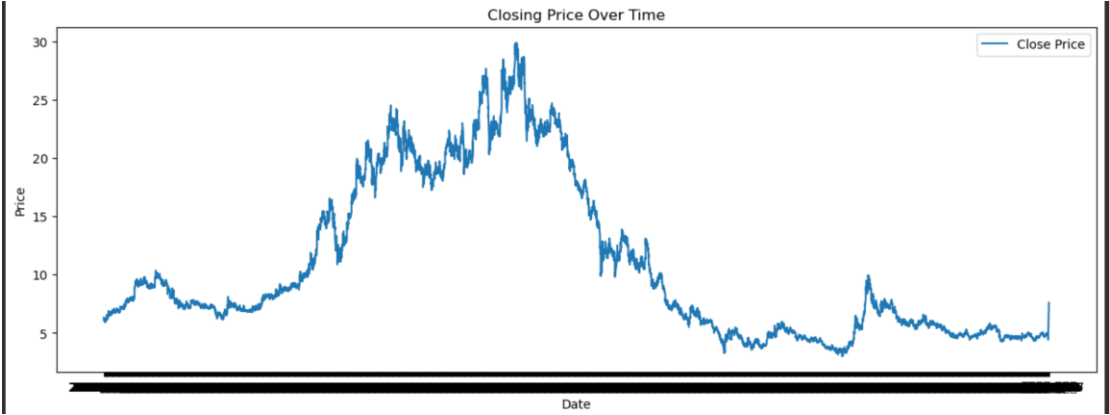


图 5. 2-1 股票的收盘价时间序列图

图 5. 2-1 股票的收盘价时间序列图展示了股票的收盘价随时间的变化。从股票市场的角度来看，这是股票表现的直观展示。在择时策略中，可以作为基础数据的展示，帮助投资者了解市场的整体趋势。

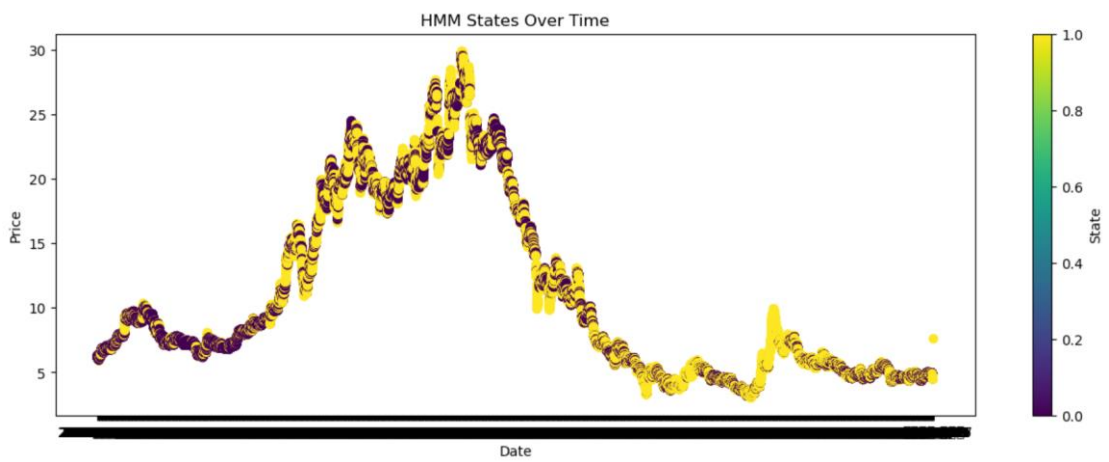


图 5. 2-2 状态随时间变化散点图

图 5.2-2 状态随时间变化散点图通过不同颜色标记了不同的市场状态，这些状态是由 HMM 模型预测的。可以帮助投资者识别市场可能的转折点，例如，从上涨转为下跌，或者相反。每个颜色代表一个特定的市场状态，可能与市场的上涨或下跌相对应。

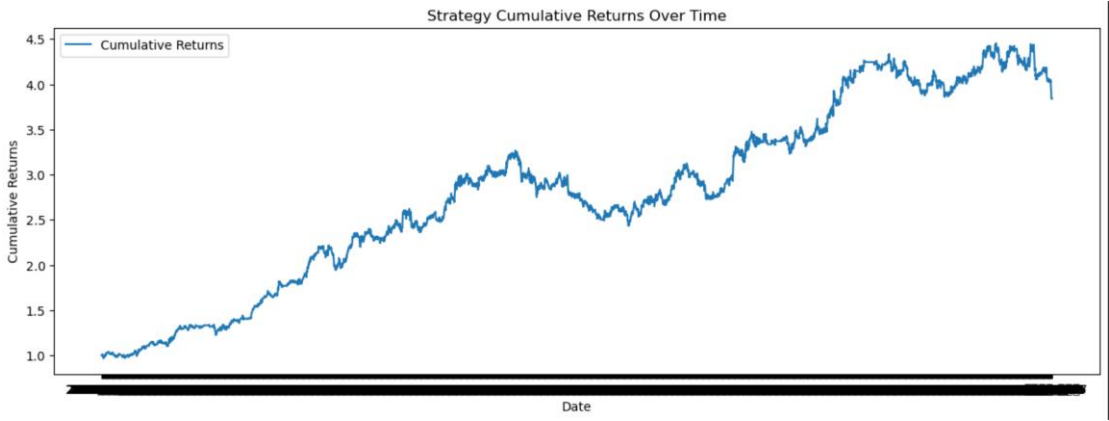


图 5.2-3 策略的累计回报随时间变化图

图 5.2-3 策略的累计回报随时间变化图展示了基于 HMM 模型生成的交易信号所实施策略的累计回报。如果策略的累计回报曲线在大部分时间内高于 1，这表示策略在该时段内是盈利的。

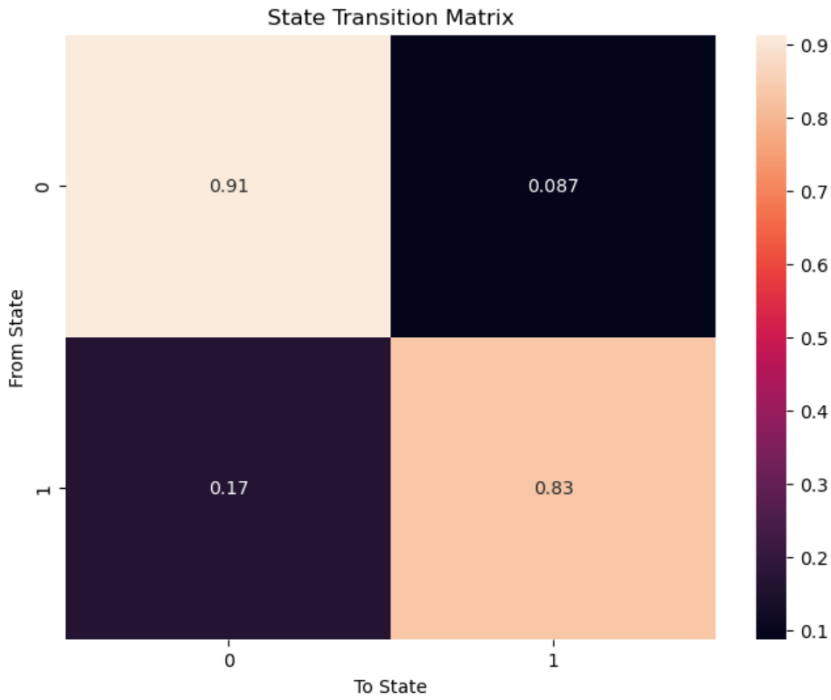


图 5.2-4 状态转移矩阵

这状态转移矩阵图显示了模型状态之间的转移概率。热图中的每个方块代表从一个状态转移到另一个状态的概率。一个高的自我转移概率（从 0 状态到 0 状态）意味着市场有很高的概率保持当前的状态。

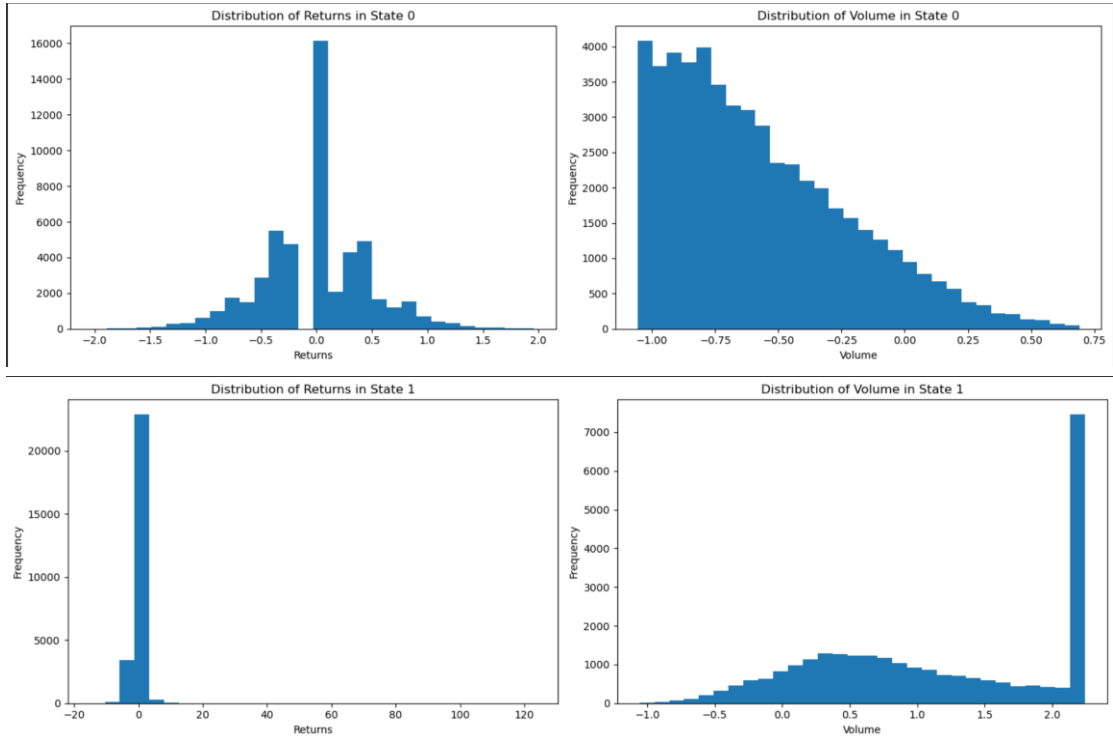


图 5.2-5 各状态下的收益率和成交量分布图

这些直方图显示了在模型定义的每个状态下，收益率和成交量的分布情况。有助于投资者了解不同状态下市场行为的特征，在一个状态可能表现出高波动性和低交易量。

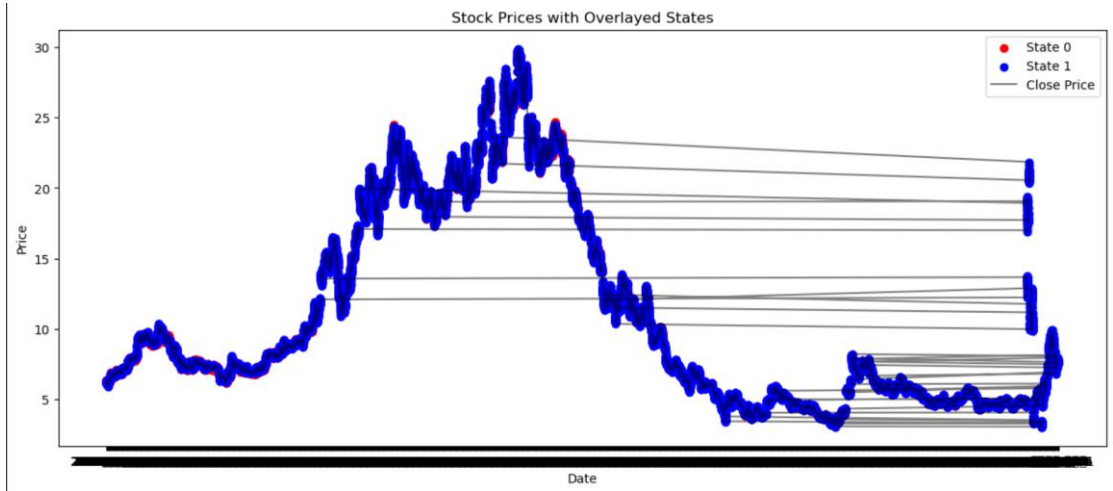


图 5.2-6 叠加状态的股价图

图 5.2-6 叠加状态的股价图将不同的市场状态叠加在股票价格图上，通过不同的颜色区分不同状态。有助于可视化模型预测的状态与实际价格之间的关系，并揭示模式或趋势。



图 5.2-7 策略与市场累计回报对比图

图 5.2-7 策略与市场累计回报对比图比较了策略的累计回报和市场指数（如标普 500 指数）的累计回报。可以展示策略相对于整个市场表现的有效性。

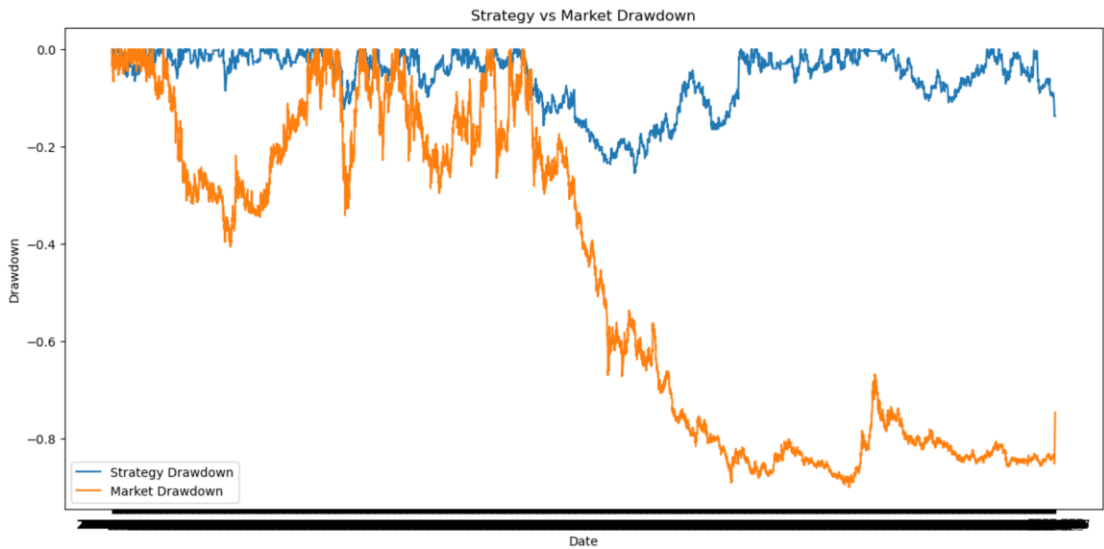


图 5.2-8 策略与市场回撤对比图

图 5.2-8 策略与市场回撤对比图显示了策略和市场的最大回撤，可以帮助投资者评估策略在不同市场条件下的风险敞口。

5.3 回测评价指标

年化收益率 (Annualized Return) :

年化收益率是将策略或基准的累积收益率转换成年度的平均收益率。这是通过复合增长率计算的，它表明如果每日收益率持续一个完整的年度，投资者可以期望的平均回报。年化收益率考虑了资本的复利效应。

夏普比率 (Sharpe Ratio) :

夏普比率衡量了每单位总风险获得的超额回报（超过无风险利率的回报）。它是一个风险调整后的性能指标，允许投资者了解策略的回报是否是由于聪明的投资决策还是一个过度的风险。

索提诺比率 (Sortino Ratio) :

索提诺比率类似于夏普比率，但它只通过下行风险来分母，即只考虑不利方向的波动，而不是总波动。

信息比率 (Information Ratio) :

信息比率衡量了策略相对于基准的表现。它是超额回报的年化值除以追踪误差（策略回报和基准回报之差的标准差）。

贝塔系数 (Beta) :

贝塔系数衡量了策略相对于市场整体波动的敏感度。如果一个策略的贝塔系数是 1，那么策略的价格将与市场同步变动。贝塔高于 1 表示策略价格波动大于市场，贝塔小于 1 表示波动小于市场。

阿尔法系数 (Alpha) :

阿尔法系数衡量策略相对于市场基准的表现。一个正的阿尔法值意味着策略在调整了市场风险后仍然获得了超额回报。

追踪误差 (Tracking Error) :

追踪误差是衡量策略回报与基准回报之间差异波动性的指标。它是一个年化值，用于衡量策略与基准之间的差异。

下行风险 (Downside Risk) :

下行风险是指策略回报低于无风险回报时的波动性。它是一个专注于不利波动的风险指标。

最大回撤 (Max Drawdown) :

最大回撤是指策略在任何时间点向下跌落的最大百分比，通常用来衡量策略或投资的潜在亏损。

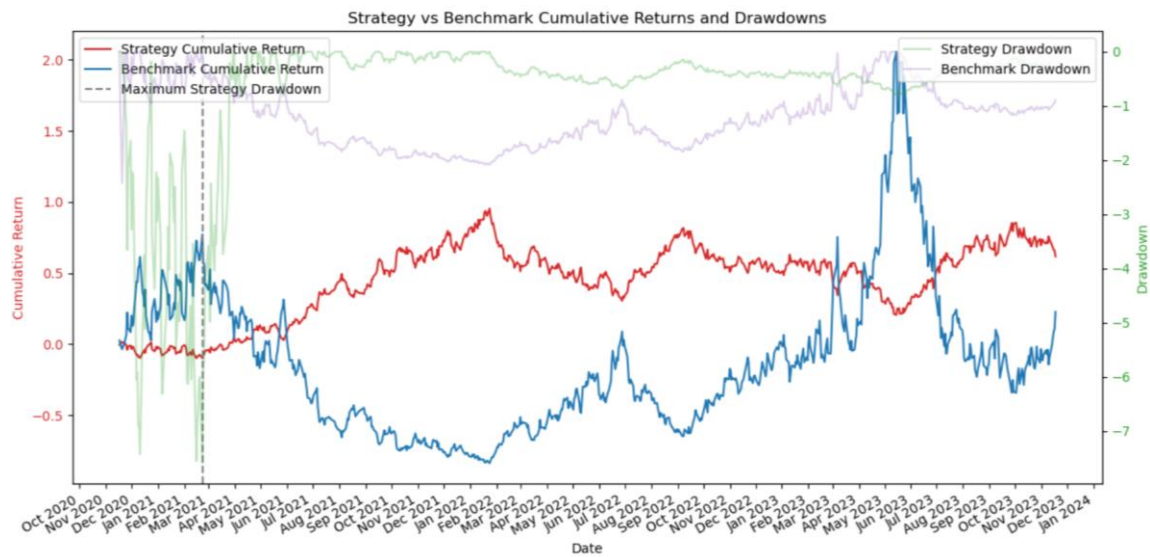


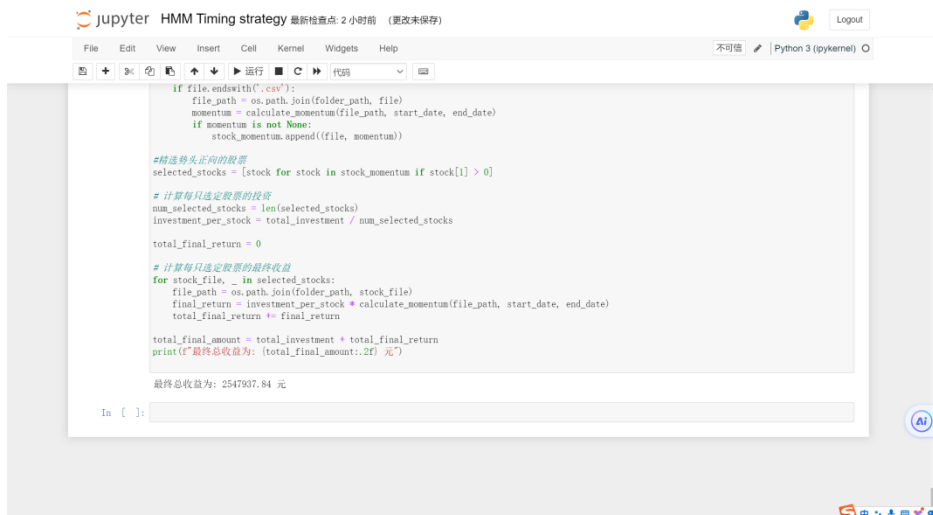
图 5.2-9 策略相对于基准的累计回报图



图 5.2-10 最大回撤图

6. 模型评估与感想

模型结果分析



```
if file.endswith('.csv'):
    file_path = os.path.join(folder_path, file)
    momentum = calculate_momentum(file_path, start_date, end_date)
    if momentum is not None:
        stock_momentum.append((file, momentum))

# 筛选势头正向的股票
selected_stocks = [stock for stock in stock_momentum if stock[1] > 0]

# 计算每只选定股票的投资
num_selected_stocks = len(selected_stocks)
investment_per_stock = total_investment / num_selected_stocks

total_final_return = 0

# 计算每只选定股票的最终收益
for stock_file in selected_stocks:
    file_path = os.path.join(folder_path, stock_file)
    final_return = investment_per_stock * calculate_momentum(file_path, start_date, end_date)
    total_final_return += final_return

total_final_amount = total_investment + total_final_return
print(f'最终总收益为: {total_final_amount:.2f} 元')

最终总收益为: 2547937.84 元
```

图 6 策略与市场回撤对比图

本次实验的结果表明，通过对恒生指数日线的预测和结合五分钟成分股的数据信息，实施的交易策略在起始资金为 100 万的情况下，最终实现了 2543797.84 元的收益。这意味着策略不仅收回了初始投资，还获得了相当可观的额外收益。在这次特定的实验中，使用隐马尔可夫模型（HMM）进行的市场时机选择和股票投资策略是有效的，并且超过了初始的预期。实验结果显示，所采取的方法能够识别出有利可图的交易机会，并且在实验期间获得了高于平均的回报。

任何投资策略都具有一定的市场风险，且过去的表现不代表未来的结果。此外，由于缺乏详细的实验设计、风险管理措施、市场条件分析和长期性能评估等信息，因此不能简单地推广这一结论，认为该策略在所有市场条件下都会同样有效。实际应用中，需要更全面的分析和测试，以确保策略的稳健性和适应性。

感想

在这个项目中，我们使用机器学习技术对恒生指数进行了增强，时间跨度从 2020 年 11 月到 2023 年 11 月，初始资金为 100 万元人民币。通过结合日线和 5 分钟时间间隔的数据分析、LSTM 和 Transformer 模型进行预测以及马尔可夫决策过程模型进行日内交易策略，您最终获得了可观的最终利润，超过了初始投资。

然而，需要注意的是，投资策略固有市场风险，并且过去的业绩并不保证未来结果。这个特定实验的成功归因于有效的择时策略和股票选择方法论，这些策略识

别了有利可图的机会。这强调了进行全面分析和严格测试的必要性，以确保这样的策略在不同市场条件下的稳健性和适应性。