组合投资与数字经济

摘　要

现如今，股票投资成为了越来越多人关注的话题。特别是随着互联网的普及和信息的便捷获取，越来越多的人参与到股票市场中。然而，股票市场的高风险和波动性使得投资者需要具备一定的数据分析能力和策略，以避免损失并获取更好的回报。

针对问题一，经初步观察，附件二为技术指标、国际市场相关指标以及汇率，附件三为国内宏观指标。采用收盘价来代表中证指数，由于中证指数是针对于国内的股票的指数，故主要分析技术指标对其影响，次要分析国际市场对其影响。对于国际市场部分缺失数据进行**局部线性插值**，并对附件数据进行正态分布检验、**Spearman** 相关分析以及 P 值检验，最后结合基于 **LightGBM** 模型的 **SHAP** 方法进行综合分析，筛选出对中证指数影响最大的 10 个指标，分别为 DMA、VMACD、EXPMA、BBI、OBV、BIAS、BOLL、KDJ、RSI 和东京日经 225 指数。

针对问题二，为了选出最值得购买的 10 支股票，构建了包括**总收益率（TR）、相对强弱指数（RSI）、移动平均收敛散度指标（MACD）、价格波动比率（PR）和随机危险系数（RD）**在内的综合评价指标并予以不同的权重。由于附件数据为半年的数据，故采用了**稳健性投资策略**。在初步筛选过程中，首先剔除半年收益率为负的股票，接着通过 **SAW（加权求和）**和 **VIKOR（多目标优化方法）**来计算综合评分，最终确定 10 支最具投资价值的股票，分别是伟星股份、冀中能源、兖矿能源、建发股份、华电国际、华发股份、凌霄泵业、美盈森、保利发展和鲁阳节能。

针对问题三，鉴于投资周期为一个月，对问题二中的 10 支股票进行了调整，选择了**收益较高**且呈**上升趋势**的股票。使用基于**高斯滤波**的 **LSTM 模型**对这些股票的收盘价进行预测，并在预测过程中加入了随机噪声。预测结果中存在股票月收益率为负现象，对此，仍然选择进行投资以模拟实际股市的亏损情况。随后，并根据题目要求进行**线性规划**。结果如表所示，月收益率达到 **11.45%**。

针对问题四，首先进行理论分析，以确定 50 只股票对中证指数的影响。这一阶段的分析将理解这些股票的市场表现如何影响指数的变化。接下来，应用**多元线性回归**方法对数据进行建模。通过回归分析，可以量化这些股票对中证指数的影响程度，并利用模型进行预测。

综合运用技术指标分析、股票选择模型、短期投资调整和股票对中证指数的影响评估，构建了一个系统性的股票投资策略。首先，利技术指标和国际市场数据，筛选出影响中证指数的关键因素，并通过模型确定了 10 支具有投资价值的股票。随后，采用高斯滤波的 LSTM 模型进行短期投资预测。最后，通过多元线性回归量化了这些股票对中证指数的影响，为进一步的市场预测和投资决策奠定了基础。

注意：**股市有风险，投资需谨慎。**

关键词：Spearman　LightGBM　SHAP　SAW　VIKOR　LSTM　多元线性回归

目录

一、 问题重述

随着越来越多的人参与股票投资，股市成为热门话题。为了避免被割韭菜，数据分析能力变得至关重要。

本文提供了从 2021 年 7 月 14 日到 2022 年 1 月 28 日的股票数据，包括 50 支股票的收盘价数据以及中证指数在这段期间的变化情况。此外，附件二提供了一些计量经济学数据，如技术指标、国外市场情况和外币汇率等；附件三则包含了中国的宏观经济数据。基于这些数据，需要解决以下几个问题：

问题一：分析附件二和附件三中的指标对中证指数的影响，并筛选出对中证指数影响最大的 10 个指标。

问题二：根据附件一提供的股票数据，评价这 50 支股票并选择出最值得购买的 10 支股票。

问题三：如果从附件一的 50 支股票中选取 10 支进行组合投资，要求每支股票的仓位不能超过 25%。总体的月收益率必须在中证指数波动率上下 5 个百分点之内（含 5 个点），超出这范围则不允许投资。基于半年历史数据确定投资策略，并预测从 1 月 28 日到 2 月 28 日的月收益率。

问题四：分析中证指数与这 50 支股票之间的关联和影响大小，并根据前面问题的结论对中证指数进行预测。

二、 问题分析

2.1 问题一的分析

经初步观察，附件二包含技术指标、国际市场相关指标以及汇率数据，而附件三则涵盖了国内宏观经济指标。以收盘价作为中证指数的代表进行分析，因为中证指数主要反映国内股票市场的表现，因此，重点将放在技术指标对中证指数的影响，次要分析国际市场因素的作用。对于国际市场数据中的缺失值，将采用局部线性插值进行填补。此外，对附件数据进行正态分布检验、Spearman 相关分析及 P 值检验，以确保数据质量和分析的准确性。最终，结合基于 LightGBM 模型的 SHAP 方法进行综合分析，筛选出对中证指数影响最大的 10 个指标，从而为决策提供更加精准的参考。

2.2 问题二的分析

为了选出最值得购买的 10 支股票，构建了一个综合评价指标体系，包括总收益率（TR）、相对强弱指数（RSI）、移动平均收敛散度指标（MACD）、价格波动比率（PR）和随机危险系数（RD），并为各指标分配不同的权重。考虑到附件数据覆盖了半年的时间范围，采用了稳健性投资策略以降低短期波动的影响。在初步筛选过程中，首先剔除了半年收益率为负的股票，然后通过 SAW（加权求和）和 VIKOR（多目标优化方法）对剩余股票进行综合评分。最终，根据评分结果，确定了 10 支最具投资价值的股票，以便在投资决策中提供更具参考价值的选择。

2.3 问题三的分析

考虑到投资周期为一个月，对问题二中筛选出的 10 支股票进行了进一步调整，优先选择了那些收益较高且呈上升趋势的股票。使用基于高斯滤波的 LSTM 模型对这些股票的收盘价进行预测，并在预测过程中加入了随机噪声，以模拟实际市场中的不确定性。虽然预测结果中出现了部分股票月收益率为负的情况，但为了反映真实股市的亏损风险，仍选择将这些股票纳入投资组合。随后，根据题目要求，进行了线性规划分析，最终实现了月收益率达到 11.45%的投资效果。

2.4 问题的分析

首先进行理论分析，以确定 50 只股票对中证指数的影响。这包括评估各股票的表现特点和市场动态，探讨它们与中证指数之间的关系。在理论分析之后，将应用多元线性回归方法对数据进行建模。通过回归分析，量化这些股票对中证指数的具体影响程度。

三、 模型假设

- 假设数据预处理正确；
- 假设采用收盘价来代表中证指数；
- 假设模型参数达到最优；
- 假设长期投资策略表现稳健；
- 假设短期投资追求高收益；
- 假设投资者有能力承受短期投资的风险。

四、 符号说明

符号	说明
x_i	数据的日期
y_i	对应日期的指标值
z_i	中证指数的收盘价
d_i	第i对数据的排名差异
p_i	每日收盘价
n	样本的数量
Y	指标值的特征矩阵
K	树的数量（LightGBM）
α_k	第k颗树的权重（LightGBM）
TR	总收益率
RSI	相对强弱指数
$MACD$	移动平均收敛散度指标
PR	价格波动比率
RD	随机危险系数
N	周期天数
CR	中证指数波动率

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 模型的建立

线性插值

线性插值是一种常用的数值分析方法，用于估计两个已知数据点之间的数值。在初步分析过程中，发现附件二中的国际市场相关指标存在缺失数据。对此，使用线性插值，基于已有的指标数据点，对缺失数据进行线性插值，以便于后于分析。

$$y = y_0 + \frac{(x - x_0)}{(x_1 - x_0)} \cdot (y_1 - y_0) \tag{1}$$

其中(x_0, y_0)和(x_1, y_1)是已知数据点，x是插值点，y是插值结果。

线性插值效果如下图所示：

东京日经 225 指数		
日期	插值前	插值后
2021-7-16	28,003.08	28,003.08
2021-7-19	27,652.74	27,652.74
2021-7-20	27,388.16	27,388.16
2021-7-21	27,548.00	27,548.00
2021-7-22		27,643.10
2021-7-23		27,738.19
2021-7-26	27,833.29	27,833.29
2021-7-27	27,970.22	27,970.22
2021-7-28	27,581.66	27,581.66
2021-7-29	27,782.42	27,782.42

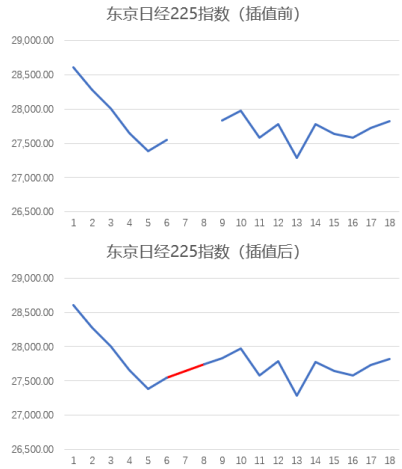

图 1 东京日经 225 指数插值图

Spearman

Spearman 相关系数用于测量两个变量之间的单调关系，而不要求这种关系是线性的。其核心在于通过排名来评估变量间的相关性。对于附件二中技术指标与国际市场相关指标对中证指数的影响，Spearman 相关系数可以有效揭示这些指标与中证指数之间的相关性。

具体过程如下：

1 排序：将每个变量的值按升序排列，并为每个值分配一个排名。如果有相同的值，使用平均排名。

2 计算排名差异：对两个变量的排名差异进行计算，得到每对排名之间的差异d_i、

3 计算相关系数：使用以下公式计算 Spearman 相关系数ρ。

$$\rho = 1 - \frac{6 \cdot \sum d_i^2}{n \cdot \left(n^2 - 1\right)} \tag{2}$$

其中，d_i是第i对数据的排名差异，n是样本数量。

通过使用 Spearman 相关系数，可以有效地识别附件二哪些指标对中证指数有最显著的影响，为后续的模型建立和分析提供重要的参考。

LightGBM

LightGBM 是一种基于梯度提升决策树（Gradient Boosting Decision Tree，GBDT）的机器学习算法，在分析附件二中的指标对中证指数的影响时，可以利用 LightGBM 来建立一个预测模型。

具体过程如下：

1 将附件二中的指标数据作为特征变量。

2 提取中证指数的历史数据作为目标变量。

3 构造目标函数

$$\begin{cases} f\left(Y\right) = \sum_{k=1}^{K} \alpha_k \cdot h_k\left(Y\right) \\ Objective = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - f\left(Y_i\right)\right)^2 + \mathrm{Re}\, g \end{cases} \tag{3}$$

其中，K是树的数量，α_k是第k颗树的权重，$h_k\left(Y\right)$是第k颗树对输入特征Y的预测值。

SHAP

SHAP (SHapley Additive exPlanations) 是一种解释个体预测的算法，基于博弈论和局部解释，旨在提高模型的可解释性。其核心思想是计算个体在合作中的边际贡献，从而评估每个特征对最终预测结果的贡献。

本文采用基于 LightGBM 模型和 SHAP 方法来分析影响中证指数的特征重要性。通过对特征进行全面分析，并将结果进行可视化，可以直观地了解各个指标特征对中证指数的正面或负面影响。

5.1.2 问题求解

首先，对附件数据进行整合，并根据中证指数的日期进行排序。对各个指标进行正态分布检验后发现，仅有少量指标符合正态分布，因此选择采用 Spearman 相关系数进行分析。

Spearman 相关系数不要求变量之间的关系是线性的，因此可以有效地识别变量之间的相关性。根据构建的 Spearman 相关系数矩阵，可以看出，技术指标与中证指

数之间的相关性强于国际市场指标与中证指数之间的相关性。

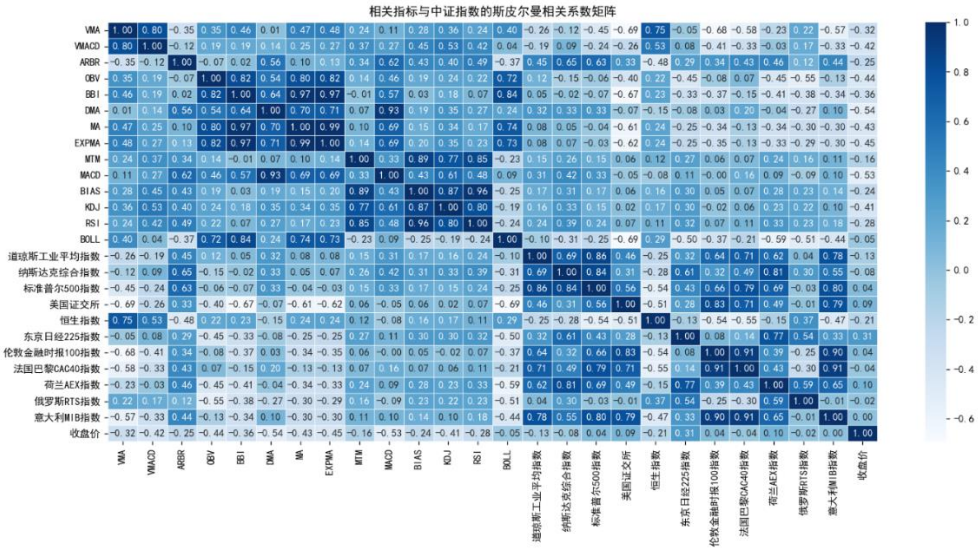

图 2 斯皮尔曼相关系数矩阵

同时，对 Spearman 相关系数进行 P 值检验，以评估观察到的相关性是否具有统计显著性。P 值反映了在原假设成立的情况下，观察到当前结果或更极端结果出现的概率。较小的 P 值（通常小于 0.05）表明观察到的相关性不太可能是偶然出现的，从而支持拒绝原假设，认为结果具有统计显著性。

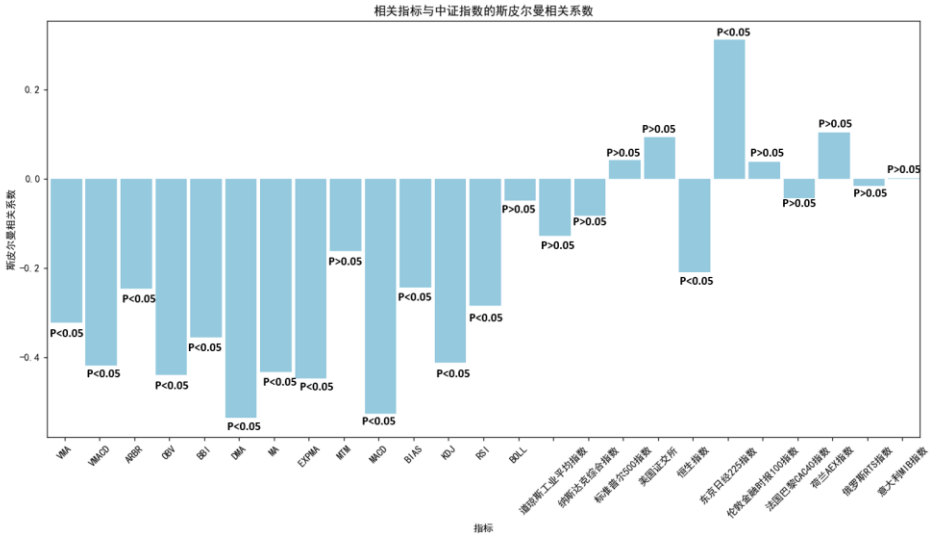

图 3 P 值检验图

由于传统的特征重要性计算方法存在一定局限性，因此利用 SHAP 值来计算特征重要性，即利用每个特征 SHAP 值绝对值的平均值表示该特征的重要性。

对于模型训练，在进行数据集划分时，先对数据进行 MIN-MAX 标准化处理，公式如下：

$$y_i^{'} = \frac{y_i - y_{\min}}{y_{\max} - y_{\min}} \tag{4}$$

标准化结果（部分）如下图所示：

VMACD	ARBR	OBV	BBI
214824880.6627	50.3031	27817549.5559	1643.3478
209078029.7327	48.9231	28140107.8012	1640.9668
231155529.8904	53.3311	27781242.7749	1636.944
178547963.565	66.254	28055779.4219	1635.8629
90002191.3321	78.8494	28273536.4494	1635.8776

⬇ 标准化

VMACD	ARBR	OBV	BBI
0.90620056	0.01108564	0.64953294	0.95456927
0.89830427	0	0.75075957	0.94354102
0.92863919	0.03540977	0.63813899	0.92490833
0.85635537	0.13922034	0.72429527	0.91990092
0.73469177	0.24040008	0.79263273	0.919969

图 4 标准化示意图

数据集被划分为训练集和测试集，其中训练集占 70%，测试集占 30%。这种划分方式使得训练集可以用来训练模型，而测试集则用于评估模型的性能和泛化能力。

对于模型参数，采用了基于随机搜索的超参数调优方法。该方法通过在预定义的超参数空间内随机选择不同的参数组合进行试验，从而寻找最优的超参数配置。这种随机搜索的方式相比于网格搜索能够更高效地探索超参数空间，并且能够减少计算成本和时间开销，有助于提升模型的性能和泛化能力。

对于模型选择，由于数据量较小且变量较多，使用 LightGBM 相比于 XGBoost 和 RandomForest 可能更具优势。LightGBM 在处理小数据集时表现出色，具有更高的计算效率和更快的训练速度，同时在处理高维数据时也能有效避免过拟合。

模型的对比结果如下表所示：

表 1 模型对比

评价指标	LightGBM	XGBoost	RandomForest
R^2	**0.9349**	0.8867	0.8812
RMSE	**57.9154**	57.7845	78.2603
MAE	**46.4599**	76.4419	60.0734
综合排名	**1**	2	3

为了更直观的判断给出了各参数的重要性排名，对于相关指标的 SHAP 值进行可视化，可视化结果如下：

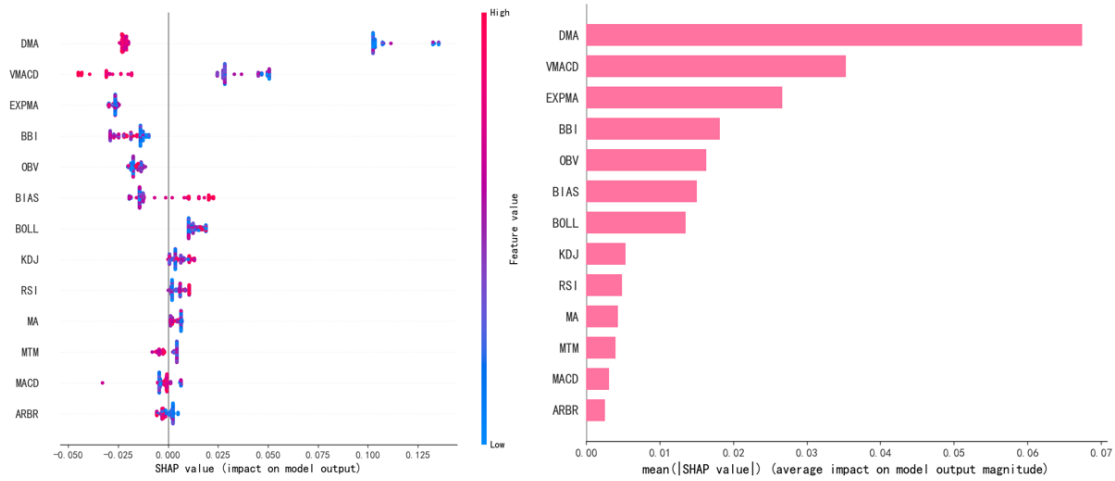

图 5SHAP 图

在整体样本上，展示不同特征的 SHAP 值分布。左图中，每一行代表一个特征，横坐标显示该特征的 SHAP 值。每个点代表一个样本，颜色越红表示特征值越大，颜色越蓝则表示特征值越小。

综合 Spearman 相关系数与 SHAP 方法综合筛选出对中证指数影响最大的 10 个指标的结果如下表所示：

表 2 对中证指数影响最大的 10 个指标

对中证指数影响最大的 10 个指标			
1	DMA	6	BIAS
2	VMACD	7	BOLL
3	EXPMA	8	KDJ
4	BBI	9	RSI
5	OBV	10	东京日经 225 指数

5.2 问题二模型的建立与求解

5.2.1 模型建立

为了综合评估中附件一中的 50 支股票，本文将选取以下指标作为评价标准，以分析每支股票的表现和潜力：

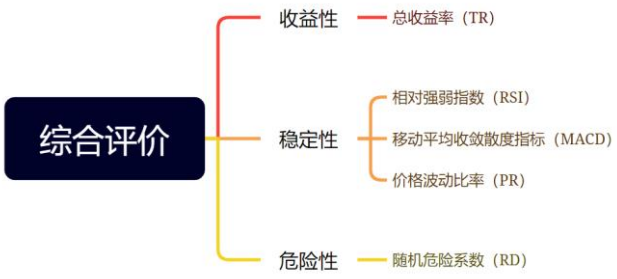

图 6 综合评价标准示意图

总收益率（TR）：总收益率是评估股票投资表现的重要指标，反映了投资者从股票投资中获得的回报。通常以百分比形式表示，用于衡量股票在一定时间段内的盈利能力。较高的收益率通常意味着公司运营良好，能够为股东创造价值，而较低的收益率可能表明公司面临挑战或市场环境不佳。

总收益率计算公式如下：

$$TR = \sum_{n=1}^{n} \log\left(\frac{p_i}{p_{i-1}}\right) \tag{5}$$

其中，y_i 是每日的收盘价，对数收益率相比于简单算术收益率的优点包括可加性、比较性和稳定性等。

相对强弱指数（RSI）：相对强弱指数是一个动量振荡器，用于评估股票价格的变动速度和幅度，从而判断股票的超买或超卖状态。RSI 值范围从 0 到 100，通常以 14 天为计算周期，但可以根据需要调整。一般而言，RSI 值高于 70 通常表示股票可能被超买，价格可能过高，回调风险增加；RSI 值低于 30 则可能表示股票被超卖，价格可能过低，存在反弹的潜力。高 RSI 值可能表明股票处于强势趋势中，而低 RSI 值则可能表明股票处于弱势趋势中。

计算公式如下：

$$\begin{cases} RS = \dfrac{\text{平均涨幅}}{\text{平均跌幅}} \\ RSI = 100 - \dfrac{100}{1+RS} \end{cases} \tag{6}$$

其中，平均涨幅（跌幅）的计算周期为 14 天。

移动平均收敛散度指标（MACD）：移动平均收敛/发散指标是一种广泛使用的技术分析工具，用于识别股票价格的趋势、动量及潜在的买卖信号。它通过比较短期和长期的移动平均线来生成交易信号，从而帮助投资者把握市场动向。

计算每日移动平均收敛/发散指标的步骤如下：

1 计算短期和长期的指数移动平均（EMA）

$$EMA_i = p_i \times \left(\frac{2}{N+1}\right) + EMA_{i-1} \times \left(1 - \frac{2}{N+1}\right) \tag{7}$$

其中， N 是周期天数（12 或 26）， EMA_{i-1}是前一天的EMA值 。

2 计算 MACD 线

$$MACD\text{线}_i = EMA_{12,i} - EMA_{26,i} \tag{8}$$

3 计算信号线

$$信号线_i = MACD线_i \times \left(\frac{2}{9+1}\right) + 信号线_{i-1} \times \left(1-\frac{2}{9+1}\right) \tag{9}$$

4 计算 MACD 柱状图

$$MACD柱状图_i = MACD线_i - 信号线_t \tag{10}$$

当 MACD 线从下向上穿越信号线时，通常被视为买入信号。相反，当 MACD 线从上向下穿越信号线时，则通常视为卖出信号。当柱状图变长（即 MACD 线与信号线之间的差距增大）时，通常表明当前趋势可能在增强；而柱状图变短则可能意味着趋势正在减弱。

价格波动比率（PR）：价格波动比率是衡量市场价格波动性的一个指标，通过计算收盘价的最高价与最低价之间的比例来评估价格的波动幅度。具体来说，价格波动比率是最高收盘价除以最低收盘价，计算公式为：

$$PR = \frac{p_{\max}}{p_{\min}} \tag{11}$$

其中，$p_{\max}$ 是收盘价的最大值，$p_{\min}$ 是收盘价的最小值。

随机危险系数（RD）：随机危险系数（RD）是一种用于模拟和评估股票外在风险的指标。通过引入一个随机危险系数，投资者可以对市场中的不确定性和潜在风险进行量化。

5.2.2 问题求解

首先计算 50 支股票的半年总收益率，首先剔除半年收益率为负的股票，排名前 5 的股票如下表所示：

表 3 股票排名表 1

排名	股票	半年总收益率
1	伟星股份 002003.SZ	8.742698e-01
2	兖矿能源 600188.SH	4.453286e-01
3	冀中能源 000937.SZ	4.335831e-01
4	保利发展 600048.SH	4.177391e-01
5	华阳股份 600348.SH	3.307506e-01

其中，对半年总收益率最高与最低的股票进行可视化如下：

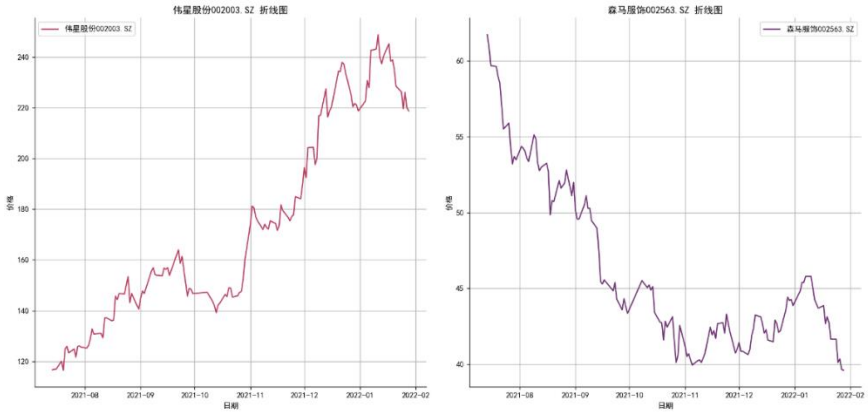

图 7 股票总收益率图

伟星股份表现出持续上涨的趋势，而森马服饰则显示出持续下跌的趋势，与计算结果一致。

共有 23 支股票的半年总收益率为正，塔牌集团 002233.SZ 的半年总收益率接近于 0，故不予考虑，后续分析将基于排名前 22 支股票进行分析评价。

由于相对强弱指数（RSI）是用一个数值来代表股票的强弱状态，因此在分析时，统计了 RSI 值位于 40 到 60 之间的天数，以衡量股票在这一中性区间内的表现。这一范围通常反映了市场处于一种相对平衡的状态，既没有过度买入也没有过度卖出。通过这种方式，可以估计股票在这一阶段的稳定性。

对与 RSI 在 40-60 之间天数最高的兖矿能源（72）与最低的伟星股份（34）进行可视化如下：

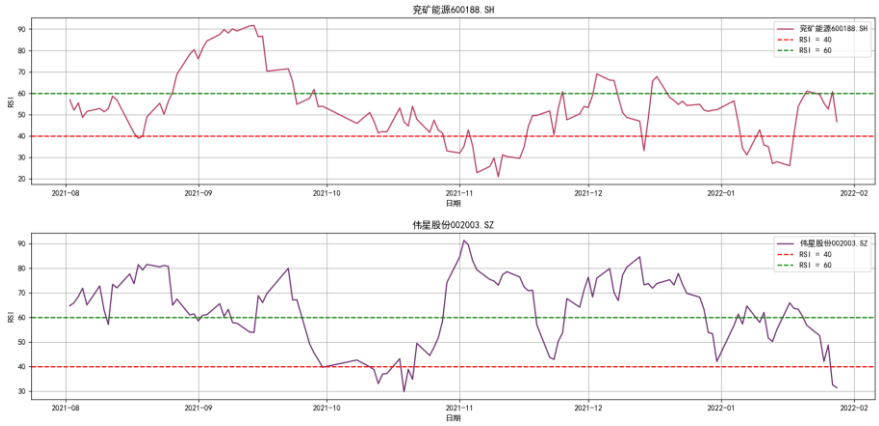

图 8 股票 RSI 图

对于中长期的 RSI 分析来说，RSI 值位于 40 到 60 之间的天数越多，通常反映出股票的稳定程度越高。这一范围表明股票在市场上处于一个相对平衡的状态，既没有过度买入也没有过度卖出，显示出市场对该股票的需求和供应趋于平衡。因此，RSI 在这一区间内的天数增多，往往意味着股票表现较为稳定，波动性较低。

同样，对于移动平均收敛散度指标（MACD），通过统计其在信号线以内的天数，可以评估股票的稳定性。MACD 线位于信号线之下的天数较多，通常表明股票的短期走势相对弱于长期趋势，市场情绪可能偏向卖出。相反，MACD 线在信号线之上的天

数增多，则表明股票的短期走势强于长期趋势，市场情绪可能偏向买入。因此，MACD 在信号线以内的天数统计有助于判断股票的市场表现稳定性。

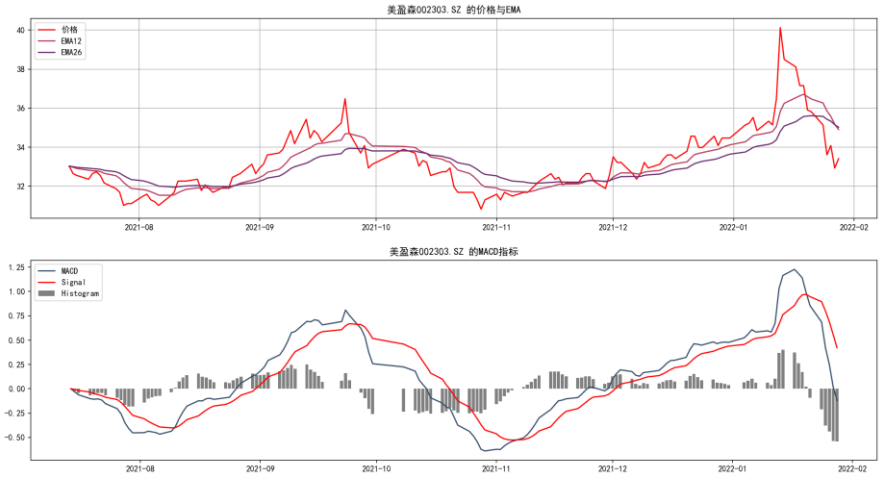

图 9 股票 MACD 图 1

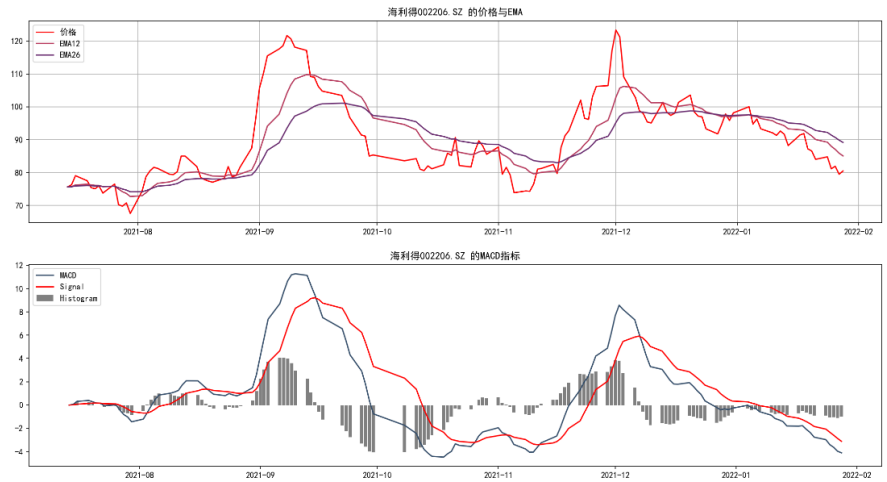

图 10 股票 MACD 图 2

在简化分析的框架下，可以认为移动平均收敛散度指标线位于信号线之上的天数越多，通常反映出股票的表现越好。这是因为 MACD 线在信号线之上，表明股票的短期走势强于长期趋势，市场情绪通常偏向买入，预示着股票可能处于上涨趋势中。较长时间的 MACD 线上方位置表明市场对该股票的需求较强，趋势较为健康，投资者对其未来表现持乐观态度。

对于价格波动比率，可以采用股票每日的收盘价最大值除以最小值来进行计算。这一比率用于衡量股票在一个交易日内的价格波动程度。具体来说，该比率提供了一个衡量股票价格波动性的指标，反映了股票的日内波动幅度。较高的比率表明股票在该交易日内的价格波动较大，可能意味着市场情绪较为激烈或存在较大的不确定性。相对较低的比率则表明价格变动较小，市场可能较为稳定。通过分析价格波动比率，投资者可以更好地评估股票的短期波动性及其市场风险。

对价格波动比率最大与最小可视化如下：

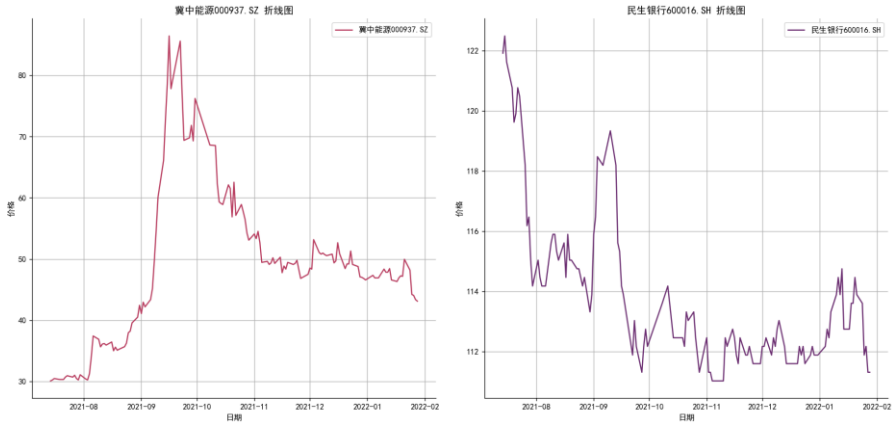

图 11 股票价格波动比率图

其中冀中能源的收盘价波动从约 30 元到约 86 元，幅度超过 50 元，而民生银行虽然总体呈下跌趋势，但价格波动却在 10 元左右。

对于随机危险系数，可以通过生成随机数来评估股票或资产在不同市场条件下的风险水平。具体方法是生成一定范围内的随机数（如 1 到 3 之间），这些数值用以模拟和量化资产在不同情景下的风险水平。每个生成的随机数代表一个风险系数，该系数反映了在特定的市场条件下资产的潜在风险。

在计算完上述 5 项评价指标后，采用综合 SAW（加权求和）和 VIKOR（多目标优化方法）进行评分。SAW（加权求和）和 VIKOR（多目标优化方法）是常用的多准则决策方法，用于综合评分和优化决策。

SAW 方法通过将各个评价指标的分值与其权重相乘，再将这些加权分值相加，得到每个选项的总评分。SAW 方法直观简洁，适用于指标数量较少且权重明确的情况。

具体步骤包括：

1 首先将各项指标的评分标准化（MIN-MAX），以确保它们在同一尺度上；

2 将每个标准化指标乘以其对应的权重，求和得到总分；

其中，权重设置为 TR: 0.7, MACD: 0.05, RSI: 0.05, PR: 0.1, RD: 0.1。

3 最后根据总分对选项进行排序，分数最高的被认为是最优选择。

VIKOR 方法则侧重于在多目标决策中找到一个尽可能接近理想解的方案。VIKOR 方法特别适合复杂多目标问题，能够详细考虑目标间的权衡和整体最优解。

具体步骤包括：

1 首先确定各指标的理想解和反理想解；

其中，理想解：TR、MACD 和 RSI　反理想解：PR 和 RD。

2 接着计算每个选项到这些解的距离；

3 然后构建一个优先序列，综合考虑到达理想解的距离和相对劣势；最后，根据综合评分和排序选择最接近理想解的方案。

选择综合使用 SAW 和 VIKOR 方法选取最值得购买的 10 支股票。

计算结果如下表所示：

表 4 股票排名表 2

排名	股票
1	伟星股份 002003.SZ
2	冀中能源 000937.SZ
3	兖矿能源 600188.SH
4	建发股份 600153.SH
5	华电国际 600027.SH
6	华发股份 600325.SH
7	凌霄泵业 002884.SZ
8	美盈森 002303.SZ
9	保利发展 600048.SH
10	鲁阳节能 002088.SZ

结果可视化如下：

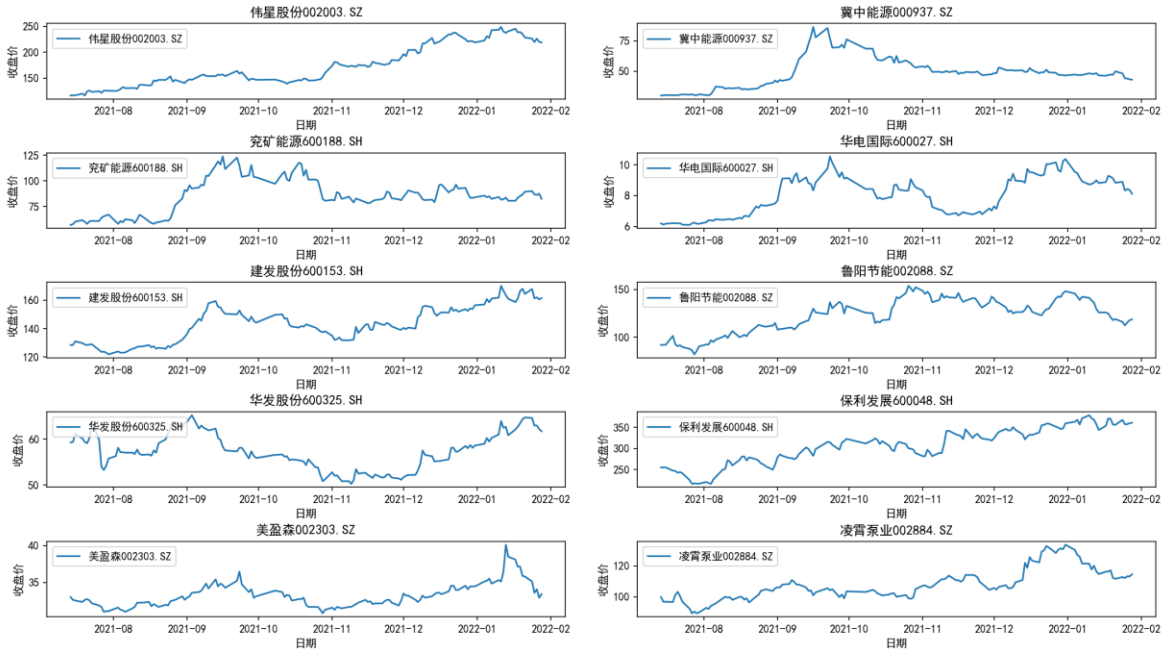

图 12 股票示意图 1

5.3 问题三模型的建立与求解

5.3.1 模型建立

高斯滤波

高斯滤波（Gaussian Filtering）是一种图像处理技术，用于平滑图像、去除噪声以及模糊图像。其基本思想是使用高斯函数作为卷积核对图像进行滤波。将高斯滤波应用于股票数据的平滑处理可以帮助去除噪声并更好地揭示数据中的趋势。

公式如下：

$$G(p_i)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{x^2}{2\sigma^2}} \tag{12}$$

其中，p_i是股票每日收盘价，σ是高斯分布的标准差，决定了滤波的宽度。

滤波过程：

$$S(p_i)=\sum_j G(p_i-p_j)\cdot p_j \tag{13}$$

其中，$G(p_i-p_j)$是高斯核的值，p_j是数据点，求和遍历领域内的数据点。

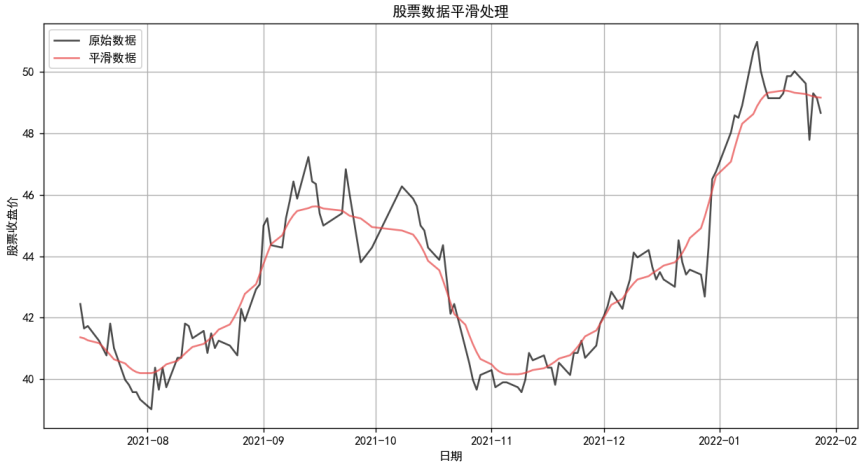

图 13 股票数据处理图

通过这种方法，可以平滑每支股票的收盘价数据，结合 LSTM 模型更准确地预测未来收盘价。

LSTM

LSTM（长短期记忆网络）是一种专门设计用于处理序列数据的深度学习模型，特别适合于股票收益率预测。LSTM 能够有效捕捉时间序列中的长期依赖关系，通过遗忘门、输入门和输出门来调控信息流动，从而解决传统 RNN 中存在的梯度消失问题。利用 LSTM，可以分析历史数据中的时间序列特征，并生成未来的收益预测。其主要组成部分包括：

1 遗忘门：决定忘记多少之前的记忆。

$$f_t=\sigma\left(W_f\cdot\left[h_{t-1},x_t\right]+b_f\right) \tag{14}$$

2 输入门：决定新信息的存储。

$$i_t=\sigma\left(W_i\cdot\left[h_{t-1},x_t\right]+b_i\right) \tag{15}$$

$$\tilde{C}_t=\tanh\left(W_C\cdot\left[h_{t-1},x_t\right]+b_C\right) \tag{16}$$

3 更新记忆单元状态：结合遗忘门和输入门来更新记忆。

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{17}$$

4 输出门：决定输出多少记忆信息。

$$o_t = \sigma\left(W_o \cdot \left[h_{t-1}, x_t\right] + b_o\right) \tag{18}$$

$$h_t = o_t \cdot \tanh\left(C_t\right) \tag{19}$$

将 LSTM 应用于投资组合优化问题，可以使用历史数据训练 LSTM 模型，以预测从 1 月 28 日到 2 月 28 日的月收益率。有助于评估每支股票的预测收益，并优化投资组合。

随机高斯噪声

随机高斯噪声是一种常见的噪声类型，在许多金融数据模型中被用来模拟随机波动和不确定性。高斯噪声以其均值为 0、方差为σ^2的正态分布来描述。将价格的随机波动建模为高斯噪声。在优化投资组合时，考虑到高斯噪声可以帮助测试策略的稳健性。通过对收益率数据施加随机高斯噪声，可以评估策略在面对市场波动时的稳定性。

蒙特卡洛模拟

蒙特卡洛模拟是一种基于随机采样的方法，用于估算复杂问题的概率和风险。常用于金融领域的投资组合优化和风险评估。通过生成大量随机样本，模拟可能的结果分布。在每次模拟中，计算投资组合的收益率及其风险，然后评估这些结果的分布情况。

5.3.2 问题求解

由于是短期投资，故在问题二的基础上，变更部分股票，选择了收益较高且呈上升趋势的股票。

选取股票可视化如下所示：

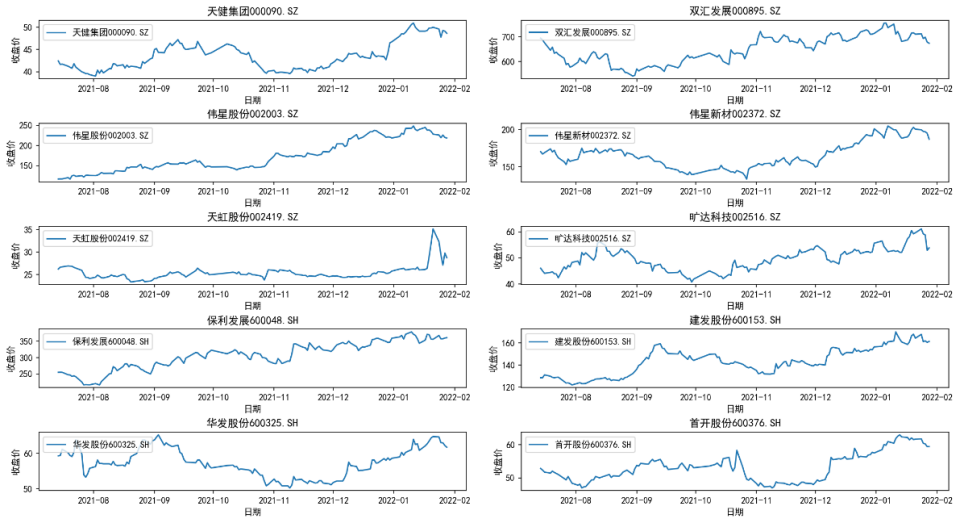

图 14 股票示意图 2

使用高斯滤波可以使股票数据平滑，从而提升模型的拟合程度，模型对比如下表所示：

表 5 模型对比表 2

表现	GF-LSTM	LSTM
训练数据 RMSE	0.7029	1.0513
测试数据 RMSE	0.7327	1.1902
训练数据 MAE	0.5630	0.8283
测试数据 MAE	0.5916	0.8421
训练数据 R^2	0.8866	0.7731
测试数据 R^2	**0.9197**	0.8171

GF-LSTM 模型在训练数据上的表现如下图所示：

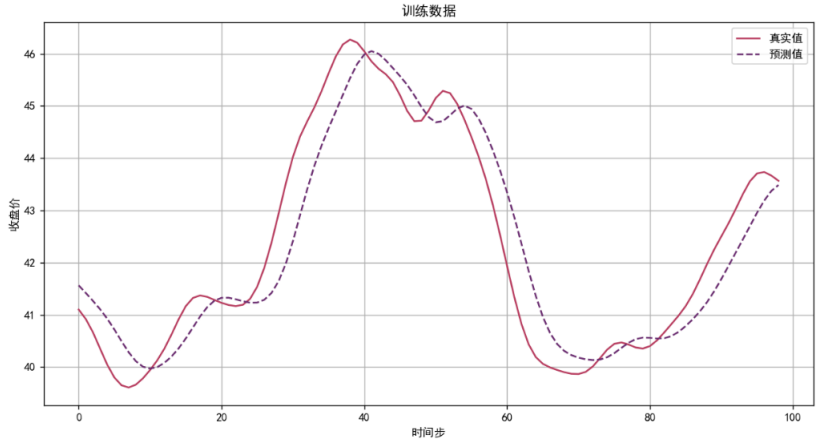

图 15 训练集示意图

GF-LSTM 模型在测试数据上的表现如下图所示：

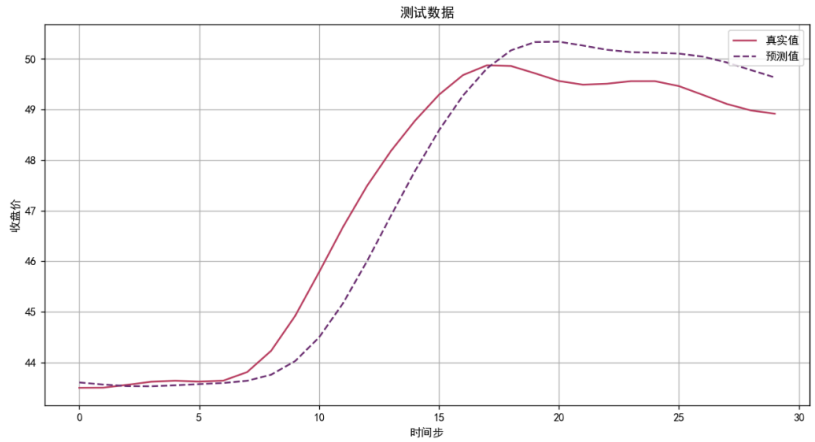

图 16 测试集示意图

训练模型之后，预测每支股票未来一个月的每日收盘价，其中，股票“天健集团”的预测存在分歧现象，及可能出现上涨或是下跌趋势，如下图所示：

图 17 股票预测图 1

该图显示该股票在未来一个月会出现持续上涨现象。

图 18 股票预测图 2

该图显示该股票在未来一个月会出现持续下跌现象。

对该股票进行门特卡罗模拟，结果如下：

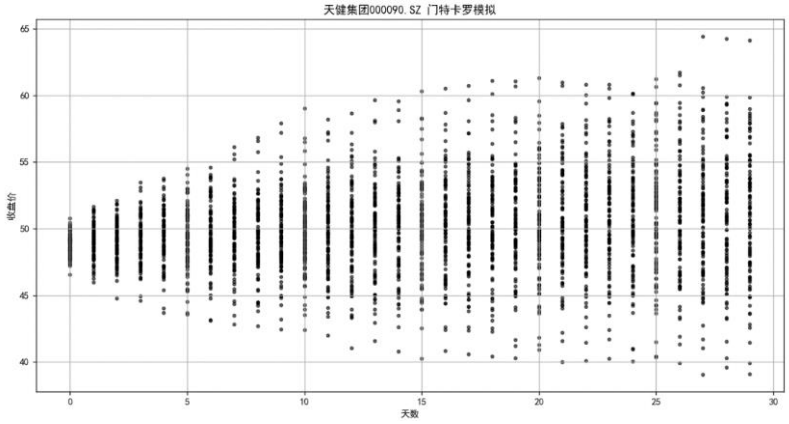

图 19 门特卡罗模拟图

结合该支股票的预测以及门特卡罗模拟，对此进行的处理为去除这支股票。

在计算月收益率时，由于模型预测较为平滑，故引用噪声数据去模拟实际的股票收盘价变动，可视化如下：

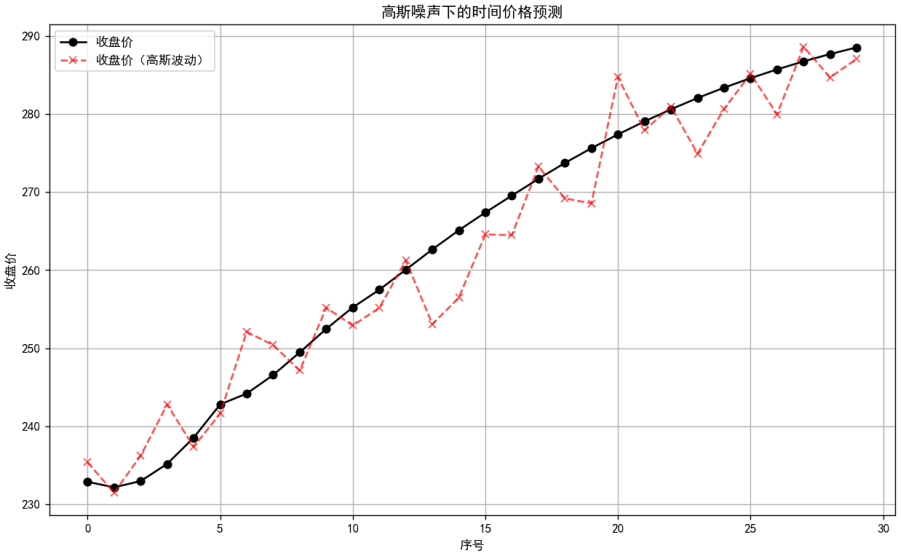

图 20 高斯噪声下的时间价格预测图

随后，计算中证指数的半年波动率区间，公式如下：

$$CR=\left(\left(\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{p_i-p_{i-1}}{p_{i-1}}\right)}\times\sqrt{135}\right)\pm 0.05\right) \tag{20}$$

计算结果为 (8.51%, 18.51%)。

随后如下定义规划问题：

$$\begin{aligned} &Max=-4.5194\times x_1+20.670\times 6x_2-7.6727\times x_3+11.0869\times x_4\\ &-3.0345\times x_5+5.1189\times x_6-7.258\times x_7+9.356\times x_8+17.8715\times x_9 \\ &s.t.\begin{cases} -4.5194\times x_1+20.670\times 6x_2-7.6727\times x_3+11.0869\times x_4\\ -3.0345\times x_5+5.1189\times x_6-7.258\times x_7+9.356\times x_8+17.8715\times x_9\le 18.51\\ x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9=1\\ 0.05\le x_i\le 0.25, i=1,2,...,9 \end{cases} \end{aligned} \tag{21}$$

其中，x_i 表示第 i 支股票，9 支股票的仓位和为 1，求解不同仓位下月收益率的最大值，同时月收益率在中证指数波动率的正负 5%以内。

求解结果如下表所示：

表 6 股票月收益率和仓位占比

股票代码	月收益率	仓位占比
双汇发展 000895.SZ	-4.5194%	0.05
伟星股份 002003.SZ	20.6706%	0.25
伟星新材 002372.SZ	-7.6727%	0.05
天虹股份 002419.SZ	11.0869%	0.2
旷达科技 002516.SZ	-3.0345%	0.05
保利发展 600048.SH	5.1189%	0.05
建发股份 600153.SH	-7.258%	0.05
华发股份 600325.SH	9.356%	0.05
首开股份 600376.SH	17.8715%	0.25

结果如表所示，最大月收益率为 11.45%。

5.4 问题四问题的分析与求解

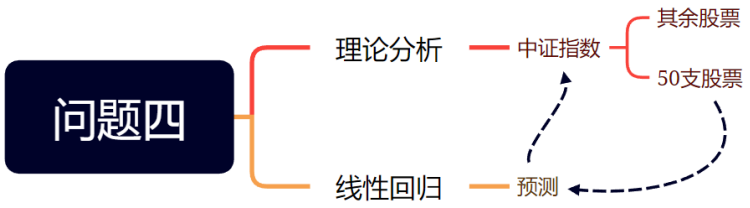

图 21 问题四分析图

理论分析

中证指数是由中证指数有限公司编制和发布的一系列证券市场指数，旨在反映中国证券市场的整体表现及特定板块的动态。计算中证指数的基本方法包括以下几个步骤：

首先，指数基于特定的股票集合进行计算，这些股票的选择依据市场表现、公司规模等标准。指数的计算通常采用市值加权法或等权重法，其中市值加权法根据成分股的市值（股价 × 流通股数）来决定各股票在指数中的权重，而等权重法则使所有成分股的权重相同。

计算公式方面，市值加权指数的计算公式为：

$$\text{指数} = \frac{\text{当前成分股总市值}}{\text{基期成分股总市值}} \times \text{基期指数} \tag{22}$$

具体计算步骤包括确定样本股票及其权重，计算成分股的总市值，然后用当前总市值与基期总市值进行比值计算，并乘以基期指数。

为了确保指数的准确性和代表性，中证指数会定期调整，包括成分股的增减以及权重的调整。这些调整帮助指数真实反映市场的变化和发展趋势。

综上所述，50 支股票在计算中证指数的股票集合里，故中证指数与这 50 支股票存在一定的影响。

多元线性回归

多元线性回归用于分析多个自变量（50 支股票的收益率）对一个因变量（中证指数收益率）的影响，并建立预测模型。通过拟合一个线性模型来理解和预测中证指数如何受到多个股票的影响。模型如下：

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \varepsilon \tag{23}$$

其中，Y 是中证指数（收盘价），$X_1, X_2, ..., X_p$ 是 50 支股票的收盘价，ε 是误差项，表示模型无法解释的部分。

模型结果为决定系数 R^2：0.9972，均方误差 MSE：117

在使用模型根据 50 支股票的每日收盘价来预测中证指数时，尽管这种方法可以提供一些有价值的见解，但存在一定的限制性。因此，为了提高预测的准确性，必须结合具体实际情况进行多因素分析。

首先，基于 50 支股票的每日收盘价的模型主要依赖于这些股票的历史数据来进行预测。这种模型可以通过建立回归模型、时间序列分析或机器学习算法来估计指数的未来走势。然而，这种方法的准确性可能受到多个因素的影响，包括市场波动、经济政策变化、行业动态以及突发事件等，这些因素都可能导致实际市场表现与模型预测之间存在偏差。

此外，中证指数的计算还涉及成分股的权重调整和定期更换。模型中只考虑股票的每日收盘价可能忽视了股票权重变动对指数的影响。为了提高预测的准确性，模型需要考虑股票的市场权重、流动性以及公司财务状况等因素。此外，市场情绪和宏观经济因素也是影响指数走势的重要因素，这些因素需要通过多因素分析来纳入模型中。

因此，综合考虑股票的基本面分析、技术面分析以及宏观经济数据，可以帮助更全面地理解和预测中证指数的变动。通过结合多种分析方法和实际情况，可以在一定程度上弥补模型的限制性，从而提供更加准确和可靠的预测结果。

六、 模型的评价、改进与推广

6.1 模型的优点

模型的优点在于将 LSTM 用于预测时结合了高斯滤波技术，这使得对股票收盘价的预测更加平滑和稳定。通过引入高斯滤波，模型能够有效地减小预测过程中由于噪声和短期波动带来的干扰，从而提高了预测的准确性和可靠性。这种平滑处理有助于更好地捕捉长期趋势，并减少对随机波动的敏感性，从而提供更为清晰的市场走势分析。

6.2 模型的缺点

模型的缺点在于在问题四中变量特征较多而数据量相对较少，这导致了多元线性回归模型出现了过拟合现象。具体而言，过多的特征相对于有限的数据样本，使得模

型能够在训练数据上表现得非常好，但却可能在实际应用中无法泛化得很好。这种过拟合现象意味着模型可能捕捉到了训练数据中的噪声和随机波动，而非真实的、一般化的模式，从而导致在新数据上的预测性能下降。因此，在这种情况下，模型的泛化能力受到限制，可能需要采取更有效的特征选择、正则化技术或增加训练数据量来缓解过拟合问题。

6.3 模型的改进

模型的改进在于引入市场风险影响因素，这一调整旨在提高模型对实际市场环境的适应性和预测准确性。通过将市场风险因素纳入模型，可以更全面地考虑可能影响股票价格波动的外部变量，例如经济指标、政策变化或市场情绪等。这种改进有助于模型更好地捕捉市场的复杂动态，从而提供更加精准的预测和分析。此外，考虑市场风险因素可以增强模型的鲁棒性，使其在面对突发事件或市场波动时表现得更加稳定，从而提高了模型的实用性和可靠性。

七、 参考文献

[1] 冯强,赵建光,杨茸,等.时间序列中非平稳性和波动性的建模及预测[J/OL].计算机科学与探索,1-10[2024-08-22].

[2] 杨园园,鲁统宇,任婷婷,等.基于时间加权和 AdaBoost 集成的动态多因子选股模型[J/OL].系统工程,1-16[2024-08-22].

[3] 岑健铭,封全喜,张丽丽,等.基于 DE-lightGBM 模型的上市公司高送转预测实证研究[J].计算机科学,2022,49(S2):137-143.

[4] 刁海璨,张延群.沪深 300 股票配对交易策略优化研究——基于 LSTM 动态参数调整和市值筛选方法[J].价格理论与实践,2024,(03):149-155.

[5] 杨科,张洲深,田凤平.高频数据环境下我国股票市场的波动率预测——基于机器学习和 HAR 模型的融合研究[J].计量经济学报,2023,3(03):886-904.

[6] 杨智勇,叶玉玺,周瑜.基于 BiLSTM-SA-TCN 时间序列模型在股票预测中的应用[J].南京信息工程大学学报(自然科学版),2023,15(06):643-651.

[7] 汤兴恒,郭强,徐天慧,等.基于多尺度核自适应滤波的股票收益预测[J].计算机应用,2023,43(05):1385-1393.

[8] Rath S ,R. N D ,Kumar B P .Stacked BI-LSTM and E-Optimized CNN-A Hybrid Deep Learning Model for Stock Price Prediction[J].Optical Memory and Neural Networks,2024,33(2):102-120.

[9] Zhu W ,Dai W ,Tang C , et al.PMANet: a time series forecasting model for Chinese stock price prediction[J].Scientific Reports,2024,14(1):18351-18351.

附录

附录 1

介绍：支撑材料的文件列表

问题一

----SHAP
--------模型对比
---------- data.xlsx
----------LightGBM.py
----------RandomForest.py
----------XGBoost.py
----------对比结果.txt
--------data.xlsx
--------LightGBM.py
----图片
--------SHAP.png
--------中证指数.png
--------中证指数矩阵.png
--------插值对比.png
--------标准化.png
----斯皮尔曼
--------data.xlsx
--------斯皮尔曼.py
--------相关系数.txt
----正太分布检验
--------data.xlsx
--------检验.py
----空值处理
--------data.xlsx
--------国际指数.xlsx
--------插值可视化.py

问题二

----图片
--------MACD1.png
--------MACD2.png
--------RSI.png
--------价格波动比率.png
--------收益率.png
--------股票.png
----股票选择 1
--------data.xlsx
--------TR.py

--------收益率.txt
--------股票选择 2
----------MACD.py
----------MACD.txt
----------PR.py
----------PR.txt
----------RSI.py
----------RSI.txt
----------股票选择 3
----------data.xlsx
----------MCDA.py
----------股票.txt
----------可视化
----------股票可视化.py

问题三

----中证指数波动率
--------data.xlsx
--------波动率.py
----图片
--------LSTM 测试数据.png
-------- LSTM 训练数据.png
--------股票.png
--------蒙特卡洛模拟.png
--------预测 1.png
--------预测 2.png
--------预测 3.png
--------高斯波动.png
--------高斯滤波.png
----波动处理
--------伟星股份 002003.SZ.txt
--------随机波动.py
----股票仓位
--------结果.txt
--------规划.py
----股票选择
--------data.xlsx
--------GF-LSTM.py
--------LSTM.py
--------LSTM.txt
--------LSTM 预测.py
--------股票展示.py
--------蒙特卡洛模拟.py
--------高斯滤波.py

问题四

----data.xlsx

----多元线性回归.py

附录 2

介绍：代码

问题一

问题一 LightGBM.py

```
# -*- coding: utf-8 -*-

from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import randint, uniform
import matplotlib.pyplot as plt
import lightgbm as lgb
import pandas as pd
import numpy as np
import shap

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet3')

# 分离特征和目标变量
X = df.drop(columns='收盘价')
y = df['收盘价']

# 初始化 MinMaxScaler 对特征进行归一化
scaler_X = MinMaxScaler()
X_scaled = scaler_X.fit_transform(X)

scaler_y = MinMaxScaler()
```

```
y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1)).flatten()

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.3, random_state=42)

# 定义参数分布
param_dist = {
    'n_estimators': randint(100, 1000),
    'max_depth': randint(3, 10),
    'learning_rate': uniform(0.01, 0.1),
    'num_leaves': randint(20, 100),
    'min_child_samples': randint(1, 20),
    'subsample': uniform(0.5, 0.5),
    'colsample_bytree': uniform(0.5, 0.5)
}

# 创建 LightGBM 回归器对象
lgb_reg = lgb.LGBMRegressor()
# 创建 RandomizedSearchCV 对象
random_search = RandomizedSearchCV(
    estimator=lgb_reg,  # 需要优化的模型，这里是 LGBMRegressor 实例
    param_distributions=param_dist,  # 超参数分布的字典，用于指定要搜索的超参数范围
    n_iter=100,  # 随机搜索的迭代次数，即从参数分布中随机选择多少组参数组合进行评估
    cv=5,  # 交叉验证的折数，即将数据划分为 5 份进行交叉验证
    scoring='r2',  # 评估模型性能的指标，这里使用 R² 评分（决定系数），用于回归任务
    verbose=1,  # 控制输出的详细程度，1 表示输出一些基本的信息
    random_state=42,  # 随机种子，用于保证每次运行时的结果一致
    n_jobs=-1  # 并行运行的作业数，-1 表示使用所有可用的 CPU 核心
)
# 执行随机搜索
random_search.fit(X_train, y_train)

# 使用最优参数的模型进行预测和评估
best_xgb_model = random_search.best_estimator_
y_pred_scaled = best_xgb_model.predict(X_test)

# 逆归一化预测结果和真实值
y_test_original = scaler_y.inverse_transform(y_test.reshape(-1, 1)).flatten()
y_pred_original = scaler_y.inverse_transform(y_pred_scaled.reshape(-1, 1)).flatten()

# 计算 MAE 和 RMSE
```

```
mae = mean_absolute_error(y_test_original, y_pred_original)
rmse = np.sqrt(mean_squared_error(y_test_original, y_pred_original))

# 计算模型的 R^2 分数
score = best_xgb_model.score(X_test, y_test)

print('-' * 10)
print("R^2: ", round(score, 4))
print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")
print('-' * 10)

explainer = shap.TreeExplainer(best_xgb_model)
shap_values = explainer.shap_values(X)  # 传入特征矩阵 X，计算 SHAP 值

# 可视化解释
# #ff73a0 #f30070 #b605a2
shap.initjs()
shap.force_plot(explainer.expected_value, shap_values[0, :], X.iloc[0, :])
shap.summary_plot(shap_values, X)
shap.summary_plot(shap_values, X, plot_type="bar", color='#b605a2')
```

问题一 LightGBM.py

```
# -*- coding: utf-8 -*-

from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import randint, uniform
import lightgbm as lgb
import pandas as pd
import numpy as np

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')
# 分离特征和目标变量
X = df.drop(columns='收盘价')
y = df['收盘价']

# 初始化 MinMaxScaler 对特征进行归一化
scaler_X = MinMaxScaler()
```

```
X_scaled = scaler_X.fit_transform(X)

scaler_y = MinMaxScaler()
y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1)).flatten()

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.3, random_state=42)

# 定义参数分布
param_dist = {
    'n_estimators': randint(100, 1000),
    'max_depth': randint(3, 10),
    'learning_rate': uniform(0.01, 0.1),
    'num_leaves': randint(20, 100),
    'min_child_samples': randint(1, 20),
    'subsample': uniform(0.5, 0.5),
    'colsample_bytree': uniform(0.5, 0.5)
}

# 创建 LightGBM 回归器对象
lgb_reg = lgb.LGBMRegressor()

# 创建 RandomizedSearchCV 对象
random_search = RandomizedSearchCV(
    estimator=lgb_reg,  # 需要优化的模型，这里是 LGBMRegressor 实例
    param_distributions=param_dist,  # 超参数分布的字典，用于指定要搜索的超参数范围
    n_iter=100,  # 随机搜索的迭代次数，即从参数分布中随机选择多少组参数组合进行评估
    cv=5,  # 交叉验证的折数，即将数据划分为 5 份进行交叉验证
    scoring='r2',  # 评估模型性能的指标，这里使用 R² 评分（决定系数），用于回归任务
    verbose=1,  # 控制输出的详细程度，1 表示输出一些基本的信息
    random_state=42,  # 随机种子，用于保证每次运行时的结果一致
    n_jobs=-1  # 并行运行的作业数，-1 表示使用所有可用的 CPU 核心
)
# 执行随机搜索
random_search.fit(X_train, y_train)

# 使用最优参数的模型进行预测和评估
best_xgb_model = random_search.best_estimator_
y_pred_scaled = best_xgb_model.predict(X_test)

# 逆归一化预测结果和真实值
```

```
y_test_original = scaler_y.inverse_transform(y_test.reshape(-1, 1)).flatten()
y_pred_original = scaler_y.inverse_transform(y_pred_scaled.reshape(-1, 1)).flatten()

# 计算 MAE 和 RMSE
mae = mean_absolute_error(y_test_original, y_pred_original)
rmse = np.sqrt(mean_squared_error(y_test_original, y_pred_original))

# 计算模型的 R^2 分数
score = best_xgb_model.score(X_test, y_test)

print('-' * 10)
print('LightGBM')
print("R^2: ", round(score, 4))
print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")
print('-' * 10)
```

问题一 RandomForest.py

```
# -*- coding: utf-8 -*-

from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import randint
import pandas as pd
import numpy as np
import warnings

warnings.filterwarnings("ignore")

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')
# 分离特征和目标变量
X = df.drop(columns='收盘价')
y = df['收盘价']

# 数据归一化
scaler_X = MinMaxScaler()
X_scaled = scaler_X.fit_transform(X)

scaler_y = MinMaxScaler()
```

```
y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1)).flatten()

# 数据分割
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.3, random_state=42)

# 创建 RandomForestRegressor 模型
model = RandomForestRegressor()

# 参数搜索空间
param_dist = {
    'n_estimators': [50, 100, 200],  # Discrete values
    'max_features': ['auto', 'sqrt', 'log2'],  # Options for max_features
    'max_depth': [None, 10, 20, 30, 40, 50],  # Discrete values for max_depth
    'min_samples_split': randint(2, 20),  # Random integer between 2 and 20 for min_samples_split
    'min_samples_leaf': randint(1, 20),  # Random integer between 1 and 20 for min_samples_leaf
    'bootstrap': [True, False]  # Boolean for bootstrap parameter
}

# 设置随机搜索
random_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_dist,
    n_iter=100,  # 迭代次数
    scoring='neg_mean_squared_error',  # 对于回归任务使用负均方误差
    cv=5,  # 交叉验证折数
    random_state=42,
    n_jobs=-1
)

# 执行随机搜索
random_search.fit(X_train, y_train)

# 输出最佳参数
# print(f"最佳参数: {random_search.best_params_}")

# 使用最佳参数的模型进行预测
best_model = random_search.best_estimator_
y_pred_scaled = best_model.predict(X_test)

# 逆归一化预测结果和真实值
y_test_original = scaler_y.inverse_transform(y_test.reshape(-1, 1)).flatten()
```

```
y_pred_original = scaler_y.inverse_transform(y_pred_scaled.reshape(-1, 1)).flatten()

# 计算评估指标
mae = mean_absolute_error(y_test_original, y_pred_original)
rmse = np.sqrt(mean_squared_error(y_test_original, y_pred_original))
score = best_model.score(X_test, y_test)

print('-' * 10)
print('RandomForest')
print("R^2: ", round(score, 4))
print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")
print('-' * 10)
```

问题一 XGBoost.py

```
# -*- coding: utf-8 -*-

from sklearn.metrics import mean_absolute_error, mean_squared_error
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import randint, uniform
import xgboost as xgb
import pandas as pd
import numpy as np

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')
# 分离特征和目标变量
X = df.drop(columns='收盘价')
y = df['收盘价']

# 数据归一化
scaler_X = MinMaxScaler()
X_scaled = scaler_X.fit_transform(X)
scaler_y = MinMaxScaler()
y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1)).flatten()

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, test_size=0.3, random_state=42)

# 定义参数分布
```

```
param_dist = {
    'n_estimators': randint(100, 1000),
    'max_depth': randint(3, 10),
    'learning_rate': uniform(0.01, 0.1),
    'gamma': uniform(0, 0.5),
    'subsample': uniform(0.5, 0.5),
    'colsample_bytree': uniform(0.5, 0.5)
}

# 创建 XGBoost 回归器对象
xgb_reg = xgb.XGBRegressor()

# 创建 RandomizedSearchCV 对象
random_search = RandomizedSearchCV(
    estimator=xgb_reg,  # 需要优化的模型，这里是 XGBRegressor 实例
    param_distributions=param_dist,  # 超参数分布的字典，用于指定要搜索的超参数范围
    n_iter=100,  # 随机搜索的迭代次数，即从参数分布中随机选择多少组参数组合进行评估
    cv=5,  # 交叉验证的折数，即将数据划分为 5 份进行交叉验证
    scoring='r2',  # 评估模型性能的指标，这里使用 R² 评分（决定系数），用于回归任务
    verbose=1,  # 控制输出的详细程度，1 表示输出一些基本的信息
    random_state=42,  # 随机种子，用于保证每次运行时的结果一致
    n_jobs=-1  # 并行运行的作业数，-1 表示使用所有可用的 CPU 核心
)

# 执行随机搜索
random_search.fit(X_train, y_train)

# 使用最优参数的模型进行预测和评估
best_model = random_search.best_estimator_
y_pred_scaled = best_model.predict(X_test)

# 逆归一化预测结果和真实值
y_test_original = scaler_y.inverse_transform(y_test.reshape(-1, 1)).flatten()
y_pred_original = scaler_y.inverse_transform(y_pred_scaled.reshape(-1, 1)).flatten()

# 计算评估指标
mae = mean_absolute_error(y_test_original, y_pred_original)
rmse = np.sqrt(mean_squared_error(y_test_original, y_pred_original))
score = best_model.score(X_test, y_test)

print('-' * 10)
```

```
print('XGBoost')
print("R^2: ", round(score, 4))
print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")
print('-' * 10)
```

问题一 斯皮尔曼.py

```
# -*- coding: utf-8 -*-

from scipy.stats import spearmanr
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import warnings

warnings.filterwarnings("ignore")
# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 1. 加载数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')

# 2. 分离特征和目标变量
X = df.drop(columns='收盘价')
y = df['收盘价']

# 创建一个新的数据框，仅包含特征与目标变量
df_corr = X.copy()
df_corr['收盘价'] = y

# 计算特征与目标变量之间的斯皮尔曼相关系数
correlation_with_target = df_corr.corr(method='spearman')['收盘价'].drop('收盘价')

# 进行显著性检验
p_values = {}
for feature in X.columns:
    corr, p_val = spearmanr(X[feature], y)
    p_values[feature] = p_val

# 打印特征与目标变量之间的相关系数和显著性
print("相关系数：")
```

```
print(correlation_with_target)
print("\n 显著性检验 p 值：")
print(p_values)

# 3. 可视化相关系数矩阵
# 计算特征与目标变量的斯皮尔曼相关系数矩阵
corr_matrix = df_corr.corr(method='spearman')

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='Blues', fmt='.2f', linewidths=0.5)
plt.title('相关指标与中证指数的斯皮尔曼相关系数矩阵')
plt.show()

# 4. 可视化相关系数条形图
plt.figure(figsize=(10, 6))
barplot = sns.barplot(x=correlation_with_target.index, y=correlation_with_target.values,
color='skyblue')
# 调整条形宽度
for patch in barplot.patches:
    patch.set_width(0.9)  # 修改条形宽度，范围从 0 到 1，0 表示无宽度，1 表示宽度为图表的宽度

plt.xlabel('指标')
plt.ylabel('斯皮尔曼相关系数')
plt.title('相关指标与中证指数的斯皮尔曼相关系数')
plt.xticks(rotation=45)
plt.grid(False)
plt.show()
```

问题一 检验.py

```
# -*- coding: utf-8 -*-

from scipy.stats import normaltest
import pandas as pd

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')

# 对所有变量进行正态分布检验
for column in df.columns:
    # 检查列是否为数值型数据
    if pd.api.types.is_numeric_dtype(df[column]):
```

```
            stat, p = normaltest(df[column])
            # print(f'变量 {column} 的正态性检验结果:')
            # print(f'Statistic={stat:.4f}, p-value={p:.4f}')

            alpha = 0.05
            if p > alpha:
                print(f'变量 {column} 符合正态分布假设')
            else:
                print(f'变量 {column} 不符合正态分布假设')
        else:
            print(f'变量 {column} 不是数值型数据，跳过检验。')
```

问题一 插值可视化.py

```
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import seaborn as sns
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

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')

# 线性插值
df.interpolate(method='linear', inplace=True)

# 对每个变量进行可视化
for column in df.columns[1:]:  # 从第二列开始，因为第一列是日期

    plt.figure(figsize=(10, 6))
    sns.lineplot(x='日期', y=column, data=df)
    plt.title(f'{column}')
    plt.xlabel('日期')
    plt.ylabel(column)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.grid(True)
    plt.show()

# 保存处理后的数据到 Excel 文件
```

```
# df.to_excel("国际指数.xlsx", index=False)
```

问题二

问题二 TR.py

```
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')

# 确保日期列被正确解析为日期格式
df['日期'] = pd.to_datetime(df['日期'])
df.set_index('日期', inplace=True)
df.sort_index(inplace=True)

# 获取所有股票列名（排除日期列）
stock_columns = [col for col in df.columns if col != '日期']

# 创建一个字典存储每支股票的总收益率
total_returns = {}

for stock in stock_columns:
    # 获取股票的收盘价数据
    prices = df[stock]

    # 计算每日对数收益率
    daily_log_returns = np.log(prices / prices.shift(1)).dropna()  # 对数收益率 = log(今天的价格 / 昨天的价格)

    # 计算累计对数收益率
    cumulative_log_return = daily_log_returns.sum()  # 所有每日对数收益率的总和
```

```
        # 将对数收益率转化为百分比形式的收益率
        cumulative_return = np.exp(cumulative_log_return) - 1
        # total_returns[stock] = cumulative_return
        if cumulative_return > 2.230446e-16:
            # 存储到字典中
            total_returns[stock] = cumulative_return
        else:
            continue

# 将结果转换为 DataFrame
total_returns_df = pd.DataFrame(list(total_returns.items()), columns=['股票', '总收益率
'])

# 按总收益率从大到小排序
total_returns_df_sorted = total_returns_df.sort_values(by='总收益率', ascending=False)

# 打印排序后的总收益率数据
print(total_returns_df_sorted)

# 选择股票数据
stock_1 = '伟星股份 002003.SZ'
stock_2 = '森马服饰 002563.SZ'

# 创建图形和子图
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

# 绘制股票 1 的折线图
ax1.plot(df.index, df[stock_1], label=stock_1, color='#b83b5e')
ax1.set_title(stock_1 + ' 折线图')
ax1.set_xlabel('日期')
ax1.set_ylabel('价格')
ax1.legend()
ax1.grid(True)
ax1.spines['top'].set_visible(False)
ax1.spines['right'].set_visible(False)

# 绘制股票 2 的折线图
ax2.plot(df.index, df[stock_2], label=stock_2, color='#6a2c70')
ax2.set_title(stock_2 + ' 折线图')
ax2.set_xlabel('日期')
ax2.set_ylabel('价格')
ax2.legend()
ax2.grid(True)
ax2.spines['top'].set_visible(False)
```

```
ax2.spines['right'].set_visible(False)

# 调整子图间距
plt.tight_layout()

# 显示图形
plt.show()
```

问题二 MACD.py

```
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import pandas as pd

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 读取数据
df = pd.read_excel("../data.xlsx", sheet_name='Sheet1')

# 确保日期列被正确解析为日期格式
df['日期'] = pd.to_datetime(df['日期'])

# 获取股票列名（排除日期列）
stock_columns = ['伟星股份 002003.SZ', '兖矿能源 600188.SH', '冀中能源 000937.SZ',
'保利发展 600048.SH',
                 '华阳股份 600348.SH', '华电国际 600027.SH', '鲁阳节能
002088.SZ', '建发股份 600153.SH',
                 '旷达科技 002516.SZ', '凌霄泵业 002884.SZ', '天健集团
000090.SZ', '首开股份 600376.SH',
                 '伟星新材 002372.SZ', '天虹股份 002419.SZ', '浙能电力
600023.SH', '海利得 002206.SZ',
                 '万年青 000789.SZ', '华发股份 600325.SH', '中国石化 600028.SH',
'雅戈尔 600177.SH',
                 '威孚高科 000581.SZ', '美盈森 002303.SZ']


# 计算 MACD
def calculate_macd(df, stock_col, short_window=12, long_window=26,
signal_window=9):
    df['EMA12'] = df[stock_col].ewm(span=short_window, adjust=False).mean()
```

```
    df['EMA26'] = df[stock_col].ewm(span=long_window, adjust=False).mean()
    df['MACD'] = df['EMA12'] - df['EMA26']
    df['Signal'] = df['MACD'].ewm(span=signal_window, adjust=False).mean()
    df['Histogram'] = df['MACD'] - df['Signal']
    return df


# 统计 MACD 大于信号线的天数
results = {}
for stock in stock_columns:
    df_macd = calculate_macd(df, stock)
    # 统计 MACD 大于信号线的天数
    days_macd_above_signal = (df_macd['MACD'] > df_macd['Signal']).sum()
    results[stock] = days_macd_above_signal

# 将结果存储到 DataFrame 并排序
results_df = pd.DataFrame(list(results.items()), columns=['股票', 'MACD 大于信号线的
天数'])
results_df = results_df.sort_values(by='MACD 大于信号线的天数', ascending=False)

# 打印排序后的结果
print(results_df)


# 绘制单独的 MACD 图
def plot_macd(df, stock, title):
    df_macd = calculate_macd(df, stock)

    plt.figure(figsize=(14, 7))

    # 绘制价格与 EMA 图
    plt.subplot(2, 1, 1)
    plt.plot(df_macd['日期'], df_macd[stock], label='价格', color='red')
    plt.plot(df_macd['日期'], df_macd['EMA12'], label='EMA12', color='#b83b5e')
    plt.plot(df_macd['日期'], df_macd['EMA26'], label='EMA26', color='#6a2c70')
    plt.title(f'{title} 的价格与 EMA')
    plt.grid(True)
    plt.legend(loc='upper left')

    # 绘制 MACD 与信号线图
    plt.subplot(2, 1, 2)
    plt.bar(df_macd['日期'], df_macd['Histogram'], label='Histogram', color='grey')
    plt.plot(df_macd['日期'], df_macd['MACD'], label='MACD', color='#364f6b')
    plt.plot(df_macd['日期'], df_macd['Signal'], label='Signal', color='red')
```

```
    plt.title(f'{title} 的 MACD 指标')
    plt.legend(loc='upper left')

    plt.tight_layout()
    plt.show()


# 绘制三钢闽光的 MACD 图
plot_macd(df, '美盈森 002303.SZ', '美盈森 002303.SZ')

# 绘制华联控股的 MACD 图
plot_macd(df, '海利得 002206.SZ', '海利得 002206.SZ')
```

问题二 PR.py

```
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import pandas as pd

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 读取 Excel 文件中的数据
df = pd.read_excel("../data.xlsx", sheet_name='Sheet1')

# 确保日期列被正确解析为日期格式
df['日期'] = pd.to_datetime(df['日期'])

# 获取股票列名（排除日期列）
stock_columns = ['伟星股份 002003.SZ', '兖矿能源 600188.SH', '冀中能源 000937.SZ',
'保利发展 600048.SH',
                 '华阳股份 600348.SH', '华电国际 600027.SH', '鲁阳节能
002088.SZ', '建发股份 600153.SH',
                 '旷达科技 002516.SZ', '凌霄泵业 002884.SZ', '天健集团
000090.SZ', '首开股份 600376.SH',
                 '伟星新材 002372.SZ', '天虹股份 002419.SZ', '浙能电力
600023.SH', '海利得 002206.SZ',
                 '万年青 000789.SZ', '华发股份 600325.SH', '中国石化 600028.SH',
'雅戈尔 600177.SH',
                 '威孚高科 000581.SZ', '美盈森 002303.SZ']
```

```
# 计算每列的最大值与最小值的比率
max_values = df[stock_columns].max()
min_values = df[stock_columns].min()
ratio_data = max_values / min_values

# 将结果转为DataFrame 以便于查看
ratio_df = pd.DataFrame({
    'Max/Min Ratio': ratio_data
}, index=stock_columns)

# 按比率从大到小排序
sorted_ratio_df = ratio_df.sort_values(by='Max/Min Ratio', ascending=False)

# 打印结果
print(sorted_ratio_df)

# 选择股票数据
stock_1 = '冀中能源 000937.SZ'
stock_2 = '民生银行 600016.SH'

# 创建图形和子图
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6))

# 绘制冀中能源的折线图
ax1.plot(df['日期'], df[stock_1], label=stock_1, color='#b83b5e')
ax1.set_title(stock_1 + ' 折线图')
ax1.set_xlabel('日期')
ax1.set_ylabel('价格')
ax1.legend()
ax1.grid(True)
ax1.spines['top'].set_visible(False)
ax1.spines['right'].set_visible(False)

# 绘制民生银行的折线图
ax2.plot(df['日期'], df[stock_2], label=stock_2, color='#6a2c70')
ax2.set_title(stock_2 + ' 折线图')
ax2.set_xlabel('日期')
ax2.set_ylabel('价格')
ax2.legend()
ax2.grid(True)
ax2.spines['top'].set_visible(False)
ax2.spines['right'].set_visible(False)

# 调整子图间距
```

```
plt.tight_layout()

# 显示图形
plt.show()
```

问题二 RSI.py

```
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import pandas as pd

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 读取 Excel 文件中的数据
df = pd.read_excel("../data.xlsx", sheet_name='Sheet1')

# 确保日期列被正确解析为日期格式
df['日期'] = pd.to_datetime(df['日期'])

# 获取股票列名（排除日期列）
stock_columns = ['伟星股份 002003.SZ', '兖矿能源 600188.SH', '冀中能源 000937.SZ',
'保利发展 600048.SH',
                 '华阳股份 600348.SH', '华电国际 600027.SH', '鲁阳节能
002088.SZ', '建发股份 600153.SH',
                 '旷达科技 002516.SZ', '凌霄泵业 002884.SZ', '天健集团
000090.SZ', '首开股份 600376.SH',
                 '伟星新材 002372.SZ', '天虹股份 002419.SZ', '浙能电力
600023.SH', '海利得 002206.SZ',
                 '万年青 000789.SZ', '华发股份 600325.SH', '中国石化 600028.SH',
'雅戈尔 600177.SH',
                 '威孚高科 000581.SZ', '美盈森 002303.SZ']


def calculate_rsi(prices, period=14):
    """计算相对强弱指数（RSI）"""
    delta = prices.diff()
    gain = (delta.where(delta > 0, 0)).rolling(window=period).mean()
    loss = (-delta.where(delta < 0, 0)).rolling(window=period).mean()
    rs = gain / loss
    rsi = 100 - (100 / (1 + rs))
```

```
    return rsi


# 初始化结果字典
rsi_in_range_counts = {stock: 0 for stock in stock_columns}

# 计算每支股票的RSI并统计符合条件的天数
for stock in stock_columns:
    df[f'{stock}_RSI'] = calculate_rsi(df[stock])
    rsi_in_range_counts[stock] = ((df[f'{stock}_RSI'] >= 40) & (df[f'{stock}_RSI'] <=
60)).sum()

# 将结果转换为DataFrame并排序
rsi_df = pd.DataFrame(list(rsi_in_range_counts.items()), columns=['Stock', 'Days in
Range'])
rsi_df_sorted = rsi_df.sort_values(by='Days in Range', ascending=False)

# 打印结果
print(rsi_df_sorted)

# 计算每支股票的RSI
df['兖矿能源 600188.SH'] = calculate_rsi(df['兖矿能源 600188.SH'])
df['伟星股份 002003.SZ'] = calculate_rsi(df['伟星股份 002003.SZ'])

# 绘制RSI图表
plt.figure(figsize=(14, 7))
# 绘制宇通客车 RSI
plt.subplot(2, 1, 1)
plt.plot(df['日期'], df['兖矿能源 600188.SH'], label='兖矿能源 600188.SH',
color='#b83b5e')
plt.axhline(y=40, color='r', linestyle='--', label='RSI = 40')
plt.axhline(y=60, color='g', linestyle='--', label='RSI = 60')
plt.title('兖矿能源 600188.SH')
plt.xlabel('日期')
plt.ylabel('RSI')
plt.legend()
plt.grid(True)

# 绘制建发股份 RSI
plt.subplot(2, 1, 2)
plt.plot(df['日期'], df['伟星股份 002003.SZ'], label='伟星股份 002003.SZ',
color='#6a2c70')
plt.axhline(y=40, color='r', linestyle='--', label='RSI = 40')
plt.axhline(y=60, color='g', linestyle='--', label='RSI = 60')
```

```
plt.title('伟星股份 002003.SZ')
plt.xlabel('日期')
plt.ylabel('RSI')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

问题二 MCDA.py

```
# -*- coding: utf-8 -*-

import pandas as pd
import numpy as np

# 读取数据
df = pd.read_excel('data.xlsx', sheet_name='Sheet1')
df['RD'] = np.random.uniform(1, 3, size=len(df))


# 定义数据的最小值和最大值用于标准化
def standardize(data, maximize):
    if maximize:
        return (data - data.min()) / (data.max() - data.min())
    else:
        return (data.max() - data) / (data.max() - data.min())


# 数据标准化
df_standardized = df.copy()

print(df.head())
print(df.columns)

# 对前三个变量标准化（越大越好）
df_standardized['TR'] = standardize(df['TR'], maximize=True)
df_standardized['RSI'] = standardize(df['RSI'], maximize=True)
df_standardized['MACD'] = standardize(df['MACD'], maximize=True)

# 对后两个变量标准化（越小越好）
df_standardized['PR'] = standardize(df['PR'], maximize=False)
df_standardized['RD'] = standardize(df['RD'], maximize=False)
```

```
print("标准化数据：")
print(df_standardized)

# 权重设置（根据需要调整）
weights = {
    'TR': 0.7,
    'MACD': 0.05,
    'RSI': 0.05,
    'PR': 0.1,
    'RD': 0.1
}

# 计算 SAW 评分
df_standardized['SAW_Score'] = (
        df_standardized['TR'] * weights['TR'] +
        df_standardized['MACD'] * weights['MACD'] +
        df_standardized['RSI'] * weights['RSI'] +
        df_standardized['PR'] * weights['PR'] +
        df_standardized['RD'] * weights['RD']
)

# 计算最优解和最差解
best = df_standardized[['TR', 'MACD', 'RSI', 'PR', 'RD']].max()
worst = df_standardized[['TR', 'MACD', 'RSI', 'PR', 'RD']].min()

# 计算每个选项的偏离度
df_standardized['S'] = (
        ((df_standardized[['TR', 'MACD', 'RSI', 'PR', 'RD']] - best) ** 2).sum(axis=1) **
0.5
)
df_standardized['R'] = (
        ((df_standardized[['TR', 'MACD', 'RSI', 'PR', 'RD']] - worst) ** 2).sum(axis=1)
** 0.5
)

# 计算 VIKOR 评分
df_standardized['VIKOR_Score'] = (
        df_standardized['S'] / df_standardized['S'].max() +
        (df_standardized['R'] / df_standardized['R'].max()) * 0.5
)

# 设置 SAW 和 VIKOR 评分的权重
weights_saw_vikor = {
```

```
    'SAW_Score': 0.5,
    'VIKOR_Score': 0.5
}

# 计算综合评分
df_standardized['Combined_Score'] = (
        df_standardized['SAW_Score'] * weights_saw_vikor['SAW_Score'] +
        df_standardized['VIKOR_Score'] * weights_saw_vikor['VIKOR_Score']
)

# 输出 SAW、VIKOR 和综合评分
print("SAW 评分：")
print(df_standardized[['股票', 'SAW_Score']])

print("VIKOR 评分：")
print(df_standardized[['股票', 'VIKOR_Score']])

# 根据综合评分排序
df_sorted = df_standardized[['股票', 'Combined_Score']].sort_values(by='Combined_Score', ascending=False)

print("综合评分：")
print(df_sorted)
```

问题二 股票可视化.py

```
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import pandas as pd

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')

# 确保日期列被正确解析为日期格式
df['日期'] = pd.to_datetime(df['日期'])

# 获取所有股票列名（排除日期列）
stock_columns = ['伟星股份 002003.SZ', '冀中能源 000937.SZ', '兖矿能源 600188.SH',
```

```
                    '华电国际 600027.SH', '建发股份 600153.SH', '鲁阳节能 002088.SZ',
                    '华发股份 600325.SH','保利发展 600048.SH','美盈森 002303.SZ',
                    '凌霄泵业 002884.SZ']

# 设置 5 行 2 列的子图
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(15, 20))

# 将股票数据绘制到子图中
for i, stock in enumerate(stock_columns):
    ax = axes[i // 2, i % 2]
    ax.plot(df['日期'], df[stock], label=stock)
    ax.set_title(stock)
    ax.set_xlabel('日期')
    ax.set_ylabel('收盘价')
    ax.legend(loc='upper left')

# 调整布局
plt.tight_layout()

# 显示图表
plt.show()
```

问题三

问题三 波动率.py

```
# -*- coding: utf-8 -*-

import pandas as pd
import numpy as np

# 读取数据
df = pd.read_excel("data.xlsx")  # 替换为你的数据文件

# 确保'日期'和'收盘价'列被正确解析
df['日期'] = pd.to_datetime(df['日期'])
df.set_index('日期', inplace=True)

# 计算日收益率
df['收益率'] = df['收盘价'].pct_change().dropna()
```

```
# 计算日波动率
daily_volatility = df['收益率'].std()

# 半年化波动率计算，135 是半年大约的交易天数
annualized_volatility = daily_volatility * np.sqrt(135)

# 计算波动率区间 (假设 +/-5% 是一个误差范围，你可以调整这个范围)
lower_bound = annualized_volatility - 0.05
upper_bound = annualized_volatility + 0.05

print(f"半年化波动率: {annualized_volatility:.2%}")
print(f"波动率区间: ({lower_bound:.2%}, {upper_bound:.2%})")
```

问题三 随机波动.py

```
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import numpy as np

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 原始价格预测数据
price_predict = [690.8492 , 687.59924, 685.6097, 684.6211, 684.2823,
                 684.16425, 682.75946, 681.8368, 681.0941, 680.3478,
                 679.5085, 678.5631, 677.7143, 676.86835, 675.99774,
                 675.10675, 674.21075, 673.3237, 672.42816, 671.524,
                 670.61505, 669.70374, 668.79004, 667.8721, 666.95105,
                 666.028, 665.1034, 664.1773, 663.2501, 662.3225
                 ]

# 设置随机种子以便结果可复现
np.random.seed(42)

# 生成高斯噪声
mean = 0   # 均值
std_dev = 3   # 标准差，控制波动幅度
noise = np.random.normal(mean, std_dev, len(price_predict))

# 创建添加噪声后的数据
price_with_noise = np.array(price_predict) + noise
```

```
# 计算收益率
returns = sum(np.diff(price_with_noise) / price_with_noise[:-1]) * 100

# 打印收益率
print(f"月收益率: {round(returns, 4)}%")

# 设置图形大小
plt.figure(figsize=(12, 7))

# 绘制原始价格预测图
plt.plot(range(len(price_predict)), price_predict, marker='o', linestyle='-', color='k', label='收盘价')

# 绘制带有高斯波动的价格图
plt.plot(range(len(price_with_noise)), price_with_noise, marker='x', linestyle='--', color='red', alpha=0.7, label='收盘价（高斯波动）')

# 添加标题和标签
plt.title('高斯噪声下的时间价格预测')
plt.xlabel('序号')
plt.ylabel('收盘价')
plt.legend()

# 显示网格
plt.grid(True)

# 显示图形
plt.show()
```

问题三 规划.py

```
# -*- coding: utf-8 -*-

import pulp

# 定义 LP 问题设置
prob = pulp.LpProblem("MyProbLP", sense=pulp.LpMaximize)

# Decision variables with unique names
x1 = pulp.LpVariable('x1', lowBound=0.05, upBound=0.25, cat='Continuous')
x2 = pulp.LpVariable('x2', lowBound=0.05, upBound=0.25, cat='Continuous')
x3 = pulp.LpVariable('x3', lowBound=0.05, upBound=0.25, cat='Continuous')
```

```
x4 = pulp.LpVariable('x4', lowBound=0.05, upBound=0.25, cat='Continuous')
x5 = pulp.LpVariable('x5', lowBound=0.05, upBound=0.25, cat='Continuous')
x6 = pulp.LpVariable('x6', lowBound=0.05, upBound=0.25, cat='Continuous')
x7 = pulp.LpVariable('x7', lowBound=0.05, upBound=0.25, cat='Continuous')
x8 = pulp.LpVariable('x8', lowBound=0.05, upBound=0.25, cat='Continuous')
x9 = pulp.LpVariable('x9', lowBound=0.05, upBound=0.25, cat='Continuous')

# 目标函数
objective = -4.5194 * x1 + 20.6706 * x2 - 7.6727 * x3 + 11.0869 * x4 - 3.0345 * x5 +
5.1189 * x6 - 7.258 * x7 + 9.356 * x8 + 17.8715 * x9
prob += objective

# 制约因素
prob += -4.5194 * x1 + 20.6706 * x2 - 7.6727 * x3 + 11.0869 * x4 - 3.0345 * x5 + 5.1189
* x6 - 7.258 * x7 + 9.356 * x8 + 17.8715 * x9 <= 18.51
prob += x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 == 1

# 解决问题
prob.solve()

# 打印结果
print("Status:", pulp.LpStatus[prob.status])
print("Objective Value:", pulp.value(objective))
for v in prob.variables():
    print(v.name, "=", v.varValue)
```

问题三 GF-LSTM.py

```
# -*- coding: utf-8 -*-

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from tensorflow.keras.layers import Dense, Dropout, LSTM
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.optimizers import Adam
from scipy.ndimage import gaussian_filter1d
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import warnings

warnings.filterwarnings('ignore')
```

```
# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 常量
feature_num = 1   # 仅使用一个特征
window = 5   # 时间窗口大小

# 加载数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')

# 确保'日期'列被解析为日期时间格式
df['日期'] = pd.to_datetime(df['日期'])

# 排除日期列并对特征进行缩放
features = df[['天健集团 000090.SZ']]

# 应用高斯滤波
sigma = 2   # 设置高斯滤波的标准差
filtered_features = gaussian_filter1d(features.values.flatten(), sigma=sigma)
filtered_features = filtered_features.reshape(-1, 1)

scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(filtered_features)

# 准备训练和测试数据
data = scaled_features
sequence_length = window + 1
result = [data[i: i + sequence_length] for i in range(len(data) - sequence_length)]
result = np.array(result)

# 划分训练集和测试集
cut = 30
train = result[:-cut]
x_train = train[:, :-1]
y_train = train[:, -1]
x_test = result[-cut:]
y_test = x_test[:, -1]

# 重塑数据以适应LSTM
X_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], feature_num))
X_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], feature_num))

# 构建并训练模型
```

```
model = Sequential([
    LSTM(64, input_shape=(window, feature_num), return_sequences=True),
    Dropout(0.2),
    LSTM(16, return_sequences=False),
    Dropout(0.2),
    Dense(4, activation='relu'),
    Dense(1)
])

model.compile(loss='mse', optimizer=Adam(), metrics=['mae'])
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0.1)

# 模型摘要
model.summary()

# 预测
y_train_predict = model.predict(X_train)
y_test_predict = model.predict(X_test)

# 逆标准化
# 逆标准化预测结果
y_train = scaler.inverse_transform(y_train.reshape(-1, 1)).reshape(-1)
y_test = scaler.inverse_transform(y_test.reshape(-1, 1)).reshape(-1)
y_train_predict = scaler.inverse_transform(y_train_predict.reshape(-1, 1)).reshape(-1)
y_test_predict = scaler.inverse_transform(y_test_predict.reshape(-1, 1)).reshape(-1)

# 计算评价指标
train_rmse = np.sqrt(mean_squared_error(y_train, y_train_predict))
test_rmse = np.sqrt(mean_squared_error(y_test, y_test_predict))
train_mae = mean_absolute_error(y_train, y_train_predict)
test_mae = mean_absolute_error(y_test, y_test_predict)
train_r2 = r2_score(y_train, y_train_predict)
test_r2 = r2_score(y_test, y_test_predict)

print('-' * 30)
print(f"训练数据 RMSE: {train_rmse:.4f}")
print(f"测试数据 RMSE: {test_rmse:.4f}")
print(f"训练数据 MAE: {train_mae:.4f}")
print(f"测试数据 MAE: {test_mae:.4f}")
print(f"训练数据 R²: {train_r2:.4f}")
print(f"测试数据 R²: {test_r2:.4f}")
print('-' * 30)

# 绘制训练数据
```

```
plt.figure(figsize=(12, 6))
plt.plot(range(len(y_train)), y_train, label='真实值', c='#b83b5e')
plt.plot(range(len(y_train_predict)), y_train_predict, label='预测值', linestyle='--', c='#6a2c70')
plt.legend(loc='upper right')
plt.title("训练数据")
plt.xlabel("时间步")
plt.ylabel("收盘价")
plt.grid(True)
plt.show()

# 绘制测试数据
plt.figure(figsize=(12, 6))
plt.plot(range(len(y_test)), y_test, label='真实值', c='#b83b5e')
plt.plot(range(len(y_test_predict)), y_test_predict, label='预测值', linestyle='--', c='#6a2c70')
plt.legend(loc='upper right')
plt.title("测试数据")
plt.xlabel("时间步")
plt.ylabel("收盘价")
plt.grid(True)
plt.show()
```

问题三 LSTM.py

```
# -*- coding: utf-8 -*-

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from tensorflow.keras.layers import Dense, Dropout, LSTM
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import warnings

warnings.filterwarnings('ignore')

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False
```

```
# 常量
feature_num = 1   # 仅使用一个特征
window = 5   # 时间窗口大小

# 加载数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')

# 确保'日期'列被解析为日期时间格式
df['日期'] = pd.to_datetime(df['日期'])

# 排除日期列并对特征进行缩放
features = df[['天健集团 000090.SZ']]
scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(features)

# 准备训练和测试数据
data = scaled_features
sequence_length = window + 1
result = [data[i: i + sequence_length] for i in range(len(data) - sequence_length)]
result = np.array(result)

# 划分训练集和测试集
cut = 30
train = result[:-cut]
x_train = train[:, :-1]
y_train = train[:, -1]
x_test = result[-cut:]
y_test = x_test[:, -1]

# 重塑数据以适应LSTM
X_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], feature_num))
X_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], feature_num))

# 构建并训练模型
model = Sequential([
    LSTM(64, input_shape=(window, feature_num), return_sequences=True),
    Dropout(0.2),
    LSTM(16, return_sequences=False),
    Dropout(0.2),
    Dense(4, activation='relu'),
    Dense(1)
])

model.compile(loss='mse', optimizer=Adam(), metrics=['mae'])
```

```
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0.1)

# 模型摘要
model.summary()

# 预测
y_train_predict = model.predict(X_train)
y_test_predict = model.predict(X_test)

# 逆标准化
# 逆标准化预测结果
y_train = scaler.inverse_transform(y_train.reshape(-1, 1)).reshape(-1)
y_test = scaler.inverse_transform(y_test.reshape(-1, 1)).reshape(-1)
y_train_predict = scaler.inverse_transform(y_train_predict.reshape(-1, 1)).reshape(-1)
y_test_predict = scaler.inverse_transform(y_test_predict.reshape(-1, 1)).reshape(-1)

# 计算评价指标
train_rmse = np.sqrt(mean_squared_error(y_train, y_train_predict))
test_rmse = np.sqrt(mean_squared_error(y_test, y_test_predict))
train_mae = mean_absolute_error(y_train, y_train_predict)
test_mae = mean_absolute_error(y_test, y_test_predict)
train_r2 = r2_score(y_train, y_train_predict)
test_r2 = r2_score(y_test, y_test_predict)

print('-' * 30)
print(f"训练数据 RMSE: {train_rmse:.4f}")
print(f"测试数据 RMSE: {test_rmse:.4f}")
print(f"训练数据 MAE: {train_mae:.4f}")
print(f"测试数据 MAE: {test_mae:.4f}")
print(f"训练数据 R²: {train_r2:.4f}")
print(f"测试数据 R²: {test_r2:.4f}")
print('-' * 30)

# 绘制训练数据
plt.figure(figsize=(12, 6))
plt.plot(range(len(y_train)), y_train, label='真实值', c='#b83b5e')
plt.plot(range(len(y_train_predict)), y_train_predict, label='预测值', linestyle='--',
c='#6a2c70')
plt.legend(loc='upper right')
plt.title("训练数据")
plt.xlabel("时间步")
plt.ylabel("收盘价")
plt.grid(True)
plt.show()
```

```
# 绘制测试数据
plt.figure(figsize=(12, 6))
plt.plot(range(len(y_test)), y_test, label='真实值', c='#b83b5e')
plt.plot(range(len(y_test_predict)), y_test_predict, label='预测值', linestyle='--', c='#6a2c70')
plt.legend(loc='upper right')
plt.title("测试数据")
plt.xlabel("时间步")
plt.ylabel("收盘价")
plt.grid(True)
plt.show()
```

问题三 LSTM 预测.py

```
# -*- coding: utf-8 -*-

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from tensorflow.keras.layers import Dense, Dropout, LSTM
from tensorflow.keras.models import Sequential
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.optimizers import Adam
from scipy.ndimage import gaussian_filter1d
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import warnings

warnings.filterwarnings('ignore')

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 常量
feature_num = 1   # 仅使用一个特征
window = 5   # 时间窗口大小

# 加载数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')
'''
Index(['日期', '天健集团 000090.SZ', '双汇发展 000895.SZ', '伟星股份 002003.SZ',
       '伟星新材 002372.SZ', '天虹股份 002419.SZ', '旷达科技 002516.SZ', '保利发
```

```
展 600048.SH',
          '建发股份 600153.SH', '华发股份 600325.SH', '首开股份 600376.SH'],
         dtype='object')
'''

# 确保'日期'列被解析为日期时间格式
df['日期'] = pd.to_datetime(df['日期'])

# 排除日期列并对特征进行缩放
features = df[['伟星股份 002003.SZ']]

# 应用高斯滤波
sigma = 2  # 设置高斯滤波的标准差
filtered_features = gaussian_filter1d(features.values.flatten(), sigma=sigma)
filtered_features = filtered_features.reshape(-1, 1)

scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(filtered_features)

# 准备训练和测试数据
data = scaled_features
sequence_length = window + 1
result = [data[i: i + sequence_length] for i in range(len(data) - sequence_length)]
result = np.array(result)

# 划分训练集和测试集
cut = 30
train = result[:-cut]
x_train = train[:, :-1]
y_train = train[:, -1]
x_test = result[-cut:]
y_test = x_test[:, -1]

# 重塑数据以适应 LSTM
X_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], feature_num))
X_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], feature_num))

# 构建并训练模型
model = Sequential([
    LSTM(64, input_shape=(window, feature_num), return_sequences=True),
    Dropout(0.2),
    LSTM(16, return_sequences=False),
    Dropout(0.2),
    Dense(4, activation='relu'),
```

```
    Dense(1)
])

model.compile(loss='mse', optimizer=Adam(), metrics=['mae'])
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0.1)

# 模型摘要
# model.summary()

# 预测
y_train_predict = model.predict(X_train)
y_test_predict = model.predict(X_test)

# 准备预测数据
last_sequence = data[-window:]  # 获取最新的 window 个数据点
predictions = []

for _ in range(30):  # 预测一个月（假设每天一个数据点）
    last_sequence = np.reshape(last_sequence, (1, window, feature_num))  # 重塑数据
    prediction = model.predict(last_sequence)
    predictions.append(prediction[0, 0])  # 存储预测值
    new_point = np.reshape(prediction, (1, 1, feature_num))  # 将预测值重塑为适合模型输入的形状
    # 使用 np.concatenate 更新 last_sequence
    last_sequence = np.concatenate((last_sequence[:, 1:, :], new_point), axis=1)  # 更新 last_sequence

# 反缩放预测值
predictions = scaler.inverse_transform(np.array(predictions).reshape(-1, 1))
print(predictions)

# 生成未来一个月的日期
future_dates = pd.date_range(start=df['日期'].max() + pd.Timedelta(days=1), periods=30, freq='B')

# 可视化
plt.figure(figsize=(12, 6))
plt.plot(df['日期'], df['伟星股份 002003.SZ'], label='历史收盘价')
plt.plot(future_dates, predictions, label='预测收盘价', color='red')
plt.xlabel('日期')
plt.ylabel('收盘价')
plt.title('伟星股份 002003.SZ 未来一个月收盘价预测')
plt.legend()
plt.grid(True)
```

```
plt.show()
```

问题三 股票展示.py

```
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import pandas as pd

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')

# 确保日期列被正确解析为日期格式
df['日期'] = pd.to_datetime(df['日期'])

# 获取所有股票列名（排除日期列）
stock_columns = [col for col in df.columns if col != '日期']

# 设置 5 行 2 列的子图
fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(15, 20))

# 将股票数据绘制到子图中
for i, stock in enumerate(stock_columns):
    ax = axes[i // 2, i % 2]
    ax.plot(df['日期'], df[stock], label=stock)
    ax.set_title(stock)
    ax.set_xlabel('日期')
    ax.set_ylabel('收盘价')
    ax.legend(loc='upper left')

# 调整布局
plt.tight_layout()

# 显示图表
plt.show()
```

问题三 蒙特卡洛模拟.py

```
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')

# 确保'日期'列被解析为日期时间格式
df['日期'] = pd.to_datetime(df['日期'])

# 计算历史收益率
df['收益率'] = df['天健集团 000090.SZ'].pct_change().dropna()

# 计算收益率的均值和标准差
mean_return = df['收益率'].mean()
std_dev = df['收益率'].std()

# 模拟参数
n_simulations = 100  # 模拟次数
n_days = 30  # 模拟天数（例如 1 年）
initial_price = df['天健集团 000090.SZ'].iloc[-1]  # 最后一个已知价格

# 创建存储模拟结果的矩阵
simulations = np.zeros((n_days, n_simulations))

# 进行蒙特卡洛模拟
for i in range(n_simulations):
    daily_returns = np.random.normal(mean_return, std_dev, n_days)
    price_path = initial_price * np.exp(np.cumsum(daily_returns))
    simulations[:, i] = price_path

# 可视化结果
plt.figure(figsize=(14, 7))
for i in range(n_simulations):
    plt.scatter(range(n_days), simulations[:, i], color='k', alpha=0.6, s=10)
plt.xlabel('天数')
plt.ylabel('收盘价')
plt.title('天健集团 000090.SZ 门特卡罗模拟')
```

```
plt.grid(True)
plt.show()
```

问题三 高斯滤波.py

```
# -*- coding: utf-8 -*-

from scipy.ndimage import gaussian_filter1d
import matplotlib.pyplot as plt
import pandas as pd

# 设置字体和负号显示
plt.rcParams['font.family'] = 'SimHei'
plt.rcParams['axes.unicode_minus'] = False

# 读取数据
df = pd.read_excel("data.xlsx", sheet_name='Sheet1')

# 确保'日期'列被解析为日期时间格式
df['日期'] = pd.to_datetime(df['日期'])

# 排除日期列并选择特征
features = df[['天健集团 000090.SZ']]

# 对特征进行平滑处理
window_size = 5  # 选择窗口大小（可以调整）
smoothed_data = gaussian_filter1d(features.values.flatten(), sigma=window_size)

# 将平滑数据添加到 DataFrame 中
df['平滑数据'] = smoothed_data

# 可视化原始数据和平滑数据
plt.figure(figsize=(12, 6))
plt.plot(df['日期'], features, label='原始数据', alpha=0.7, c='k')
plt.plot(df['日期'], df['平滑数据'], label='平滑数据', alpha=0.7, linestyle='-', c='#e84545')
plt.xlabel('日期')
plt.ylabel('股票收盘价')
plt.title('股票数据平滑处理')
plt.legend()
plt.grid(True)
plt.show()
```

问题四 多元线性回归.py

```
# -*- coding: utf-8 -*-

from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import MinMaxScaler
import pandas as pd

# 加载数据
df = pd.read_excel("data.xlsx")
df = df.drop(columns='日期')  # 删除日期列

# 分离特征和目标变量
X = df.drop(columns='收盘价')  # 特征
y = df['收盘价']  # 目标变量

# 标准化特征数据
scaler_X = MinMaxScaler()
X_scaled = scaler_X.fit_transform(X)  # 标准化特征数据

# 标准化目标变量
scaler_y = MinMaxScaler()
y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1)).flatten()  # 标准化目标变量

# 建立多元线性回归模型
model = LinearRegression()
model.fit(X_scaled, y_scaled)

# 预测
y_pred_scaled = model.predict(X_scaled)

# 将预测结果反标准化回原始尺度
y_pred = scaler_y.inverse_transform(y_pred_scaled.reshape(-1, 1)).flatten()

# 6. 计算评价指标
mse = mean_squared_error(y, y_pred)
r2 = r2_score(y, y_pred)

print(f"均方误差 (MSE): {mse}")
print(f"决定系数 (R^2): {r2}")

# 打印模型系数
coefficients = model.coef_
```

```
features = X.columns
for feature, coef in zip(features, coefficients):
    print(f"{feature}的系数: {coef}")
```