基于 XGBoost 和 LSTM 的价格预测及量化投资策略

摘　要

针对问题一，首先按照时间间隔的不同，将附录的数据分为以五分钟为时间间隔、以日为时间间隔和以月/季度为时间间隔的三类数据。其后，以数字经济板块信息的指标为准，利用 Lagrange 插值法对国际市场指标和汇率中的变量的缺失值进行插补。然后，为克服数据量纲差别过大的影响，对数据进行标准化。再后，对数据进行正态性检验，发现仅有部分变量服从正态分布。接着，对服从正态分布的数据计算 Pearson 相关系数，对不服从正态分布的数据计算 Spearman 相关系数，提取出具有较强相关性的变量。再次，利用 XGBoost 模型，带入具有较强相关性的变量分别对数字经济板块信息中的六个指标进行预测，比较每个特征的增益、覆盖度量、频率和重要性。最后，综合相关性分析和 XGBoost 模型的输出结果，我们得出数字经济板块信息中六个变量的相关变量、较重要变量和重要变量。

针对问题二，我们选取问题一种对“成交量”最重要的变量“成交额”，利用“成交量”和“成交额”对“成交量”进行预测。首先对数据集做预处理。其一是对原始数据做归一化，将数据放缩到[0, 1]之间，防止模型自适应取值范围过大的数据；其二是将数据集进行拆分重组，分别建立适用于 XGBoost 的监督学习数据集以及适用于 LSTM 神经网络的数组数据集。接着，将数据分别带入 XGBoost 模型和 LSTM 模型进行拟合和检测。对于 XGBoost 模型，其自身带有正则项，可以较好地抵御过拟合。选择较优的超参数训练后，训练集 MSE 为1.3476×10^{-7}，MAE 为2.7593×10^{-4}，测试集的 MSE 为2.4299×10^{-5}，MAE 为2.0271×10^{-3}。对于 LSTM 神经网络，本文添加了 dropout 层并且设置了早停条件，有效地防止模型过拟合。选择较优的超参数，并设置 MSE 为损失函数。训练后的 LSTM 网络在训练集上的 MSE 为 0.0012，在测试集上为8.979×10^{-5}。为直观体现预测值和真实值的差距，本文还计算出 LSTM 在训练集的 MAE 为 0.0163，在测试集为 0.0059。虽然 XGBoost 的残差较低，但是由于 XGBoost 产生了稍严重的过拟合现象，对预测的数据存在较多异常值，因此选择 LSTM 的预测结果作为最终结果。

针对问题三，利用“开盘价”“收盘价”“最高价”“最低价”对“收盘价”进行预测。XGBoost 在训练集上的 MSE 为1.4560×10^{-7}，MAE 为2.9902×10^{-4}，测试集的 MSE 为4.7569×10^{-3}，MAE 为0.03462。LSTM 在训练集上的 MSE 为 0.0024，测试集上为 0.0015。在训练集上的 MAE 为 0.0352，测试集上为 0.0266。对于测试集的精度，明显 LSTM 较好，且由于本题测试集中没有低于 1450 的数据，导致拟合能力强大的 XGBoost 没有预测低于 1450 的数据的能力，因此保留 LSTM 的预测结果。

针对问题四，首先，利用第三问预测出的 912 个收盘价，计算出每日的涨跌幅，并计算未来七个时间的总涨跌幅R_i。接着，根据附表中所提供的技术指标，选取 BLAS、KDJ 和 RIS 加入数据集。其次，利用归一化后的上述四个指标计算出牛市分数B_i。再次，利用R_i和B_i建立指标交易分数S_i。当$S_i \geq 0.5$时看涨，决策为卖出；当$S_i < 0.5$时看跌，决策为买入。以此得到 912 个时点上的所有操作。当连续看涨和连续开跌时，保留最后一个决策，分别在价格最低和最高的时候买入和卖出。然后，根据实际的收盘价和费率判断买入和卖出是否盈利，如盈利则保留决策。最后，计算 19 日的总收益率、信息比率和最大回撤率。算出 1 月 28 日的三个指标分别为 8.23%，7.03%和 0.30%。

关键词：相关性检验、特征重要性、XGBoost、LSTM、投资策略模型

目录

一、 问题重述

而近年来，随着大数据技术发展，量化投资的地位愈发重要。虽然量化投资通过分析海量的市场数据，能够客观准确地捕捉更多投资机会，但是由于影响产品价格的因素诸多，因此量化投资面临着从海量市场信息中提取出有效指标的问题。

本文根据附录所给数据，需要解决如下问题：

问题一，要求从附录中提取出与“数字经济”板块有关的主要指标。

问题二，要求根据问题一中提取出来的重要指标对成交量进行预测。

问题三，要求根据问题一中提取出来的重要指标对收盘价进行预测。

问题四，要求凭借 100 万元的初始资金以及 0.3%的费率，利用问题三所预测的数据进行交易决策，并计算总收益率、信息比率和最大回撤率。

二、 问题分析

1. 问题一

问题一要求从附录中提取出与“数字经济”板块有关的主要指标。本文首先通过数据集划分、缺失值插补、标准化和正态性检验对数据进行预处理，然后利用 Spearman 相关系数对各变量进行相关性检验，最后利用 XGBoost 算法计算各个相关变量的特征重要性，并凭此判断有关的程度。

2. 问题二的分析

问题二要求对“成交量”进行预测。本文选用问题一中从各个板块提取的对成交量最重要的指标“成交额”预测。我们分别针对 XGBoost 和 LSTM 算法的特点建立了不同的数据集，再带入模型进行拟合。经过计算，选择了较优的超参数，并且较好地预防了过拟合。最后，我们比较两个模型的 MSE，选择较优的模型预测的结果作为最终结果。

3. 问题三的分析

和问题二一样，本文直接利用可以预防过拟合和超参数较好的 XGBoost、LSTM 模型，带入“开盘价”“收盘价”“最高价”“最低价”对“收盘价”进行预测，并选择 MSE 最低的模型的预测结果。

4. *问题四的分析*

根据问题三所预测的收盘价，以及附录中所提供的技术性指标，我们定义并计算了交易分数。当交易分数大于等于 0.5 时看涨，做出卖出的决策；当交易分数小于等于 0.5 时看跌，做出买入的决策。当买入和卖出决策连续时，我们只保留最后一个决策，删除之前连续的决策。

接着，计算保留的买入和卖出决策的收益。若收率大于成本，则保留决策。反之，删除决策。最终得到 19 天交易后的总资产，并凭此计算总收益率、信息比率和最大回撤率。

三、 模型假设

为简化模型，我们定义如下假设：

1. 本题附录所给的数据均真实有效。
2. 缺失值仅来源于国内和国际交易时间的不同，而非人为或者自然因素所导致。
3. 所有做出的决策均可以实时交易完成，不需考虑实际交易因素。

四、 符号说明

符号	说明
x_i	第 i 个观测值
$\bar{x}$	真实值的平均值
μ	X 的均值
σ	X 的标准差
RMSE	损失函数
T	树里面叶子节点的个数
w	树上叶子节点的得分
w	叶子的向量
q	树的结构
y_i'	整个累加模型的输出
x_{norm}	归一化后的数据
x_{min}	变量 X 的最小值
x_{max}	变量 X 的最大值
R_i	总涨跌幅
B_i	牛市分数
S_i	买入分数
$\Omega(f_t)$	正则项
MSE	均方误差
MAE	绝对平均误差
Y_i	第 i 个真实值
$\hat{Y}_i$	第 i 个预测值
y_i	每一个收盘价的观测值
C	实际涨跌幅
$\hat{C}$	预测涨跌幅
BIAS	偏离率
KDJ	随机指标
RSI	相对强弱指标

五、 问题一的求解

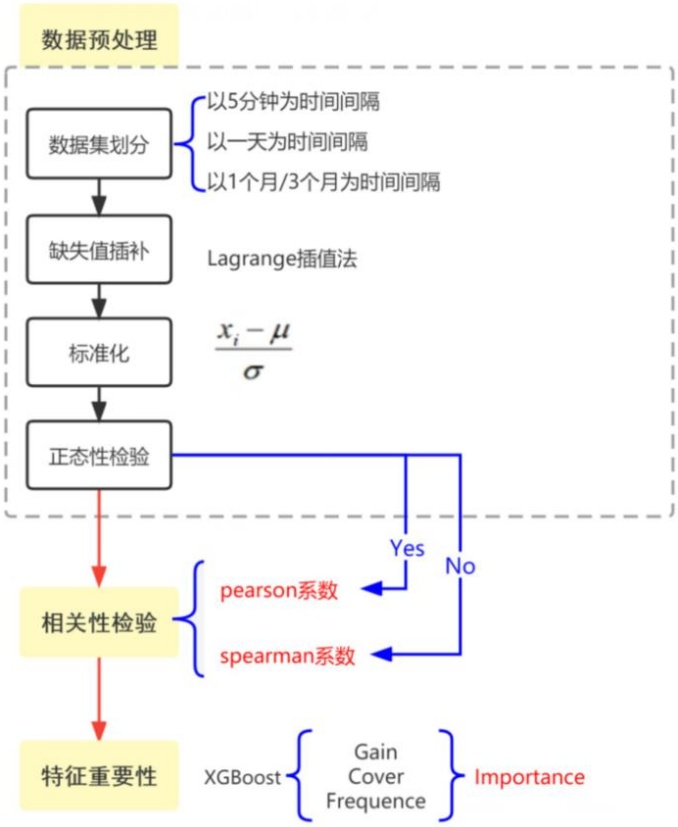


图 1 问题一的解题流程图

问题一要求分析各个板块中与“数字经济板块信息”有关的主要指标。“数字经济板块信息”中存在六个不同的变量，本题即分析与这六个变量有关的指标。

为方便做相关性检验、回归分析以及时间序列分析，首先要对数据集按照时间间隔进行划分。接着，利用 Lagrange 插值法对数据集的缺失值进行插补。再用标准化消除量纲的影响后，进行正态性检验。

如果正态性检验通过，可以计算 Pearson 相关系数进行相关性检验；反之，若正态性检验未通过，则需要计算 Spearman 相关系数。

最后，利用相关的变量，带入非线性的回归模型，通过 XGBoost 的所有叶节点的增益、覆盖度量以及频率计算出特征的重要性，并凭此判断有关的程度。

5.1 数据预处理

5.1.1 数据集的划分

根据本题附录所给的时间序列数据，存在分别以五分钟、日、月和季为时间间隔的问题。经典的时间序列分析方法一般要求时间频率相同，因此，我们以时间间隔为分类标准，将附表的数据分为三种。

1.以五分钟为时间间隔的数据

仅有“数字经济板块信息”内部的六个变量“开盘价”、“收盘价”、“最高价”、“最

低价”、“成交量”、“成交额”存在每隔五分钟的数据信息，而其他表格的时间间隔都大于五分钟。因此，将“数字经济板块信息”单独列为一种。

2.以日为时间间隔的数据

“国内市场指标”“数字经济板块信息”“技术指标”“国内市场指标”“汇率”“其他板块信息”均是以日为时间间隔的数据，可以统一作为一类。

同时，以五分钟为时间间隔的数据也可以提取出以日为时间间隔的数据信息，我们将“数字经济板块信息”的数据人为保留每日收盘的数据，合并到以日为时间间隔的数据中。

3.以月、季为时间间隔的数据

“宏观市场指标 1”和“宏观市场指标 2”内部的变量存在以月和季为时间间隔的数据信息，单独作为第三类。

5.1.2 缺失值的插补

缺失值一般可能由自然因素和人为因素产生。本题所给数据中的指标是连续的，不存在缺失值。但是由于“国际市场指标”和“汇率指标”存在交易政策、统计周期和统计规则上的不同，因此和“数字经济板块信息”的时间产生了差异，因而产生缺失值。时间序列分析和相关性分析一般都要求数据的时间频率相同，故需要对缺失值进行插补。

我们使用前后 4 个共 8 个值，利用 Lagrange 插值法对缺失值进行插补，原理如下：

当已知$X = [x_0, x_1, \cdots, x_n]$，且$X$对应的$Y = [y_0, y_1, \cdots, y_n]$时，计算如下公式：

$$l_i(x) = \frac{(x - x_0)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_0)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)} \tag{1}$$

可以得到若干个基函数，再将每个基函数乘以对应的y_i后相加，即为 Lagrange 插值多项式：

$$L_n(x) = \sum_{i=0}^{n}(y_i l_i(x)) \tag{2}$$

得到多项式后，对已知 X 而 Y 缺失的点，就可以利用多项式进行插补。

以 2022 年 1 月 17 日的指数为例，使用 Lagrange 插值法对缺失数据进行插补，并作出了插补后的拟合曲线。结果如下图所示，显然插值后的数值和原数值可以构成较好的拟合曲线。

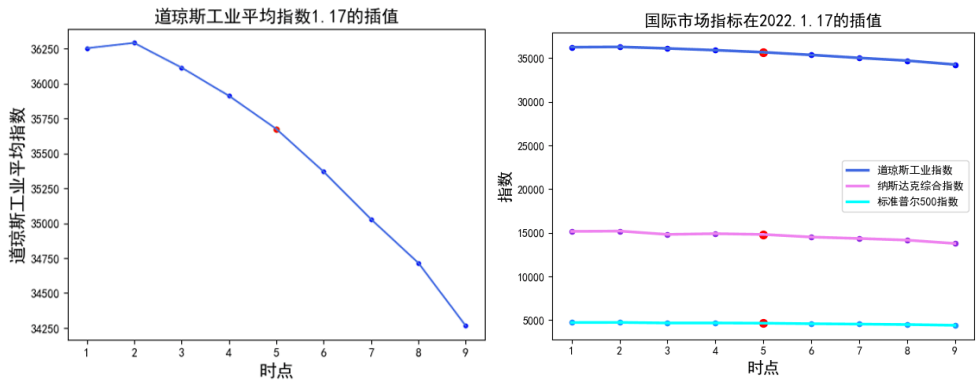


图 2 （左）道琼斯工业平均指数在 2022/1/17 0:00:00 插值后的拟合曲线

（右）国际市场指标的三个变量在 2022/1/17 0:00:00 插值后的拟合曲线

5.1.3 标准化数据

本题所给的数据中，以“数字经济板块信息”为例，开盘价是四位的数据，而成交量和成交额都存在七位及以上的数据。数据的量纲差距较大，在带入模型时，可能会导致模型仅仅对值较大的数据进行分析，忽略值较小的数据的变化影响，对预测结果较为不利。

设x_i为第 i 个观测值，μ为 X 的均值，σ为 X 的标准差，定义标准化为：

$$Z_i = \frac{(x_i - \mu)}{\sigma} \tag{3}$$

5.1.4 正态性检验

经典的统计模型一般都有数据分布为正态分布的基本假设，例如 Pearson 相关系数就要求数据服从正态分布。符合正态分布的数据一般较为稳定，带入模型的拟合效果一般会更好。由于附录所给数据的观测数远大于 50，因此我们选用 Kolmogorov-Smirnov 正态性检验对数据进行检验。对以五分钟、日为时间间隔的数据做正态性检验，结果如下所示。由于以月、季为时间间隔的数据量较少，因此无法做正态性检验。

表 1 数字经济板块信息（五分钟）的数据的 Kolmogorov-Smirnov 正态性检验

变量名	统计量	自由度	P-value
开盘价	0.058	6240	<0.001
收盘价	0.058	6240	<0.001
最高价	0.058	6240	<0.001
最低价	0.057	6240	<0.001
成交量	0.177	6240	<0.001
成交额	0.168	6240	<0.001

Kolmogorov-Smirnov 正态性检验的原假设是认为数据服从正态分布，如上表所示，时间间隔为五分钟的六个变量的 P-value 均小于 0.001，要拒绝原假设，故认为数据不服从正态分布。

但是，当时间间隔变为日时，开盘价、收盘价、最高价和最低价又服从正态分布。服从正态分布的变量的检验结果如下表所示。

表 2 正态检验中数据呈正态分布的变量（每日）

变量名	统计量	自由度	P-value
开盘价	0.067	130	0.200
收盘价	0.067	130	0.200
最高价	0.066	130	0.200
最低价	0.066	130	0.200
沪市:股票:流通市值	0.072	130	0.099
上证综合指数	0.064	130	0.200
中证 500 指数	0.046	130	0.200
创业板指数	0.076	130	0.066
上证 50 指数	0.074	130	0.075
上证 A 股指数	0.064	130	0.200
深证成份指数	0.066	130	0.200

深证综合指数	0.063	130	0.200
VMACD	0.060	130	0.200
EXPMA	0.078	130	0.053
MTM	0.076	130	0.061
RSI	0.074	130	0.074
道琼斯工业平均指数	0.050	130	0.200
纳斯达克综合指数	0.057	130	0.200
恒生指数	0.049	130	0.200
东京日经 225 指数	0.069	130	0.200
伦敦金融时报 100 指数	0.071	130	0.180
荷兰 AEX 指数	0.072	130	0.095
意大利 MIB 指数	0.073	130	0.084
数字媒体	0.062	130	0.200

5.2 相关性分析

1. 以五分钟为时间间隔的数据

根据节 5.1.4 的检验结果，以五分钟为时间间隔的数据不服从正态分布，因此不能使用 Pearson 相关系数，应该使用 Spearman 相关系数做相关性检验。检验结果如下表所示。

表 3 数字经济板块信息（每五分钟）的数据的 Spearman 相关系数（保留四位小数）

	开盘价	收盘价	最高价	最低价	成交量	成交额
开盘价	1.0000	0.9990	0.9995	0.9995	0.2729	0.3534
收盘价	0.9990	1.0000	0.9995	0.9995	0.2741	0.3545
最高价	0.9995	0.9995	1.0000	0.9993	0.2798	0.3621
最低价	0.9995	0.9995	0.9993	1.0000	0.2666	0.3453
成交量	0.2729	0.2741	0.2798	0.2666	1.0000	0.9051
成交额	0.3534	0.3545	0.3621	0.3453	0.9051	1.0000

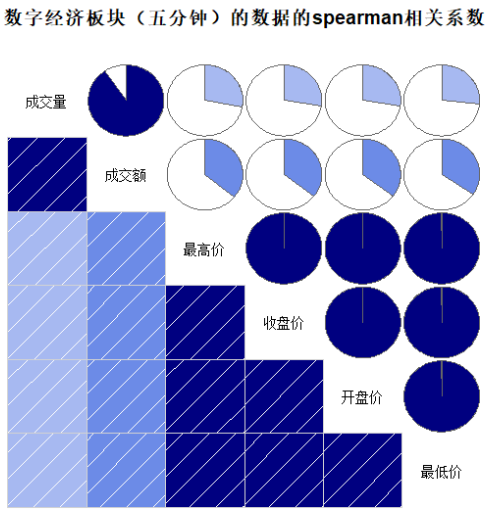


图 3 数字经济板块信息（五分钟）的数据的 Spearman 相关系数（蓝色为正相关）

上图中，蓝色代表正相关性，且颜色越深，相关性越强。显然，所有的变量都是正相关的。其中开盘价和收盘价、开盘价和最高价、开盘价与最低价、收盘价与最高价、收盘价与最低价、最高价与最低价、成交量与成交额具有较强的正相关性。而开盘价与成交量、开盘价与成交额、收盘价与成交量、收盘价与成交额、最高价与成交量、最高价与成交额、最低价与成交量、最低价与成交额只有较弱的正相关性。

2. 以日为时间间隔的数据

首先，由于以日为时间间隔的数据较多，我们先比较“数字经济板块信息”内部的六个变量的相关性。结果如下表所示，显然，无论是以日为时间间隔，还是以五分钟为时间间隔，“数字经济板块信息”的六个变量的相关性是相同的。

表 4 数字经济板块信息（每日）的数据的斯皮尔曼相关系数（保留四位小数）

	开盘价	收盘价	最高价	最低价	成交量	成交额
开盘价	1.0000	0.9998	0.9999	0.9999	0.3278	0.4749
收盘价	0.9998	1.0000	0.9999	0.9999	0.3323	0.4790
最高价	0.9999	0.9999	1.0000	0.9999	0.3288	0.4757
最低价	0.9999	0.9999	0.9999	1.0000	0.3304	0.4771
成交量	0.3278	0.3323	0.3288	0.3304	1.0000	0.9450
成交额	0.4749	0.4790	0.4757	0.4771	0.9450	1.0000

接着，我们再比较其他板块的变量和“数字经济板块信息”的六个变量的相关性。我们定义相关系数的绝对值在 0.6-0.8 为相关，在 0.8-1.0 为强相关，并提取对“数字经济板块信息”的每个变量的相关变量，结果如下表所示。

表 5 与“数字经济板块”的六个变量相关的变量及对应的相关系数

变量名	相关变量	相关系数	变量名	相关变量	相关系数
开盘价	沪深 300 指数	0.6249	**收盘价**	沪深 300 指数	0.6234
	创业板指数	0.7910		创业板指数	0.7917
	深证成份指数	0.8271		深证成份指数	0.8262
	OBV	0.8297		OBV	0.8290
	BBI	0.8951		BBI	0.8957
	DMA	0.6948		DMA	0.6955
	MA	0.9261		MA	0.9268
	EXPMA	0.9548		EXPMA	0.9554
	MACD	0.7180		MACD	0.7182
	BOLL	0.6301		BOLL	0.6304
最高价	沪深 300 指数	0.6238	**最低价**	沪深 300 指数	0.6240
	创业板指数	0.7915		创业板指数	0.7914
	深证成份指数	0.8268		深证成份指数	0.8265
	OBV	0.8299		OBV	0.8296
	BBI	0.8957		BBI	0.8953
	DMA	0.6960		DMA	0.6950
	MA	0.9266		MA	0.9261
	EXPMA	0.9552		EXPMA	0.9550

成交量	MACD	0.7189	成交额	MACD	0.7177
	BOLL	0.6300		BOLL	0.6307
	VMA	0.7899		成交金额.上证综合指数	0.6038
	VMACD	0.7132		沪市.股票.流通市值	-0.6051
	恒生指数	0.6486		VMA	0.8150
	伦敦金融时报 100 指数	-0.6101		VMACD	0.6888
	美国.美元兑人民币	0.6297		美国证交所	-0.6310
	数字孪生	-0.6221		恒生指数	0.6753
	快手概念	-0.6004		伦敦金融时报 100 指数	-0.6241
				美国.美元兑人民币	0.6710

从上表可以看出，对于开盘价、收盘价、最高价、最低价四个强相关的变量，与它们相关的变量完全相同，且相关系数接近。对于成交量和成交额这两个强相关的变量，与它们相关的变量大部分相同，仅有个别不同。

这说明开盘价、收盘价、最高价、最低价，成交量、成交额自身就存在较强联系，其他板块和它们之间存在的相关性可能是随机因素导致的伪相关。这种伪相关在进行回归时并不能对变量的预测起到较好的作用，甚至可能对预测产生干扰。为消除这种干扰，我们继续对“数字经济板块信息”的六个变量做回归分析。

3. 以月/季为时间间隔的数据

“宏观市场指标 1”中的变量“中国香港：少于 2.万人民币的存款利率：12 个月：期内平均数”是一个常数，恒为 0.3500%，显然常数和变量不存在相关性。

对“宏观市场指标 1”和“宏观市场指标 2”之中其他的变量，我们提取出相关系数的绝对值大于 0.6 的变量，如下表所示。

表 6 数字经济板块信息（月、季）的数据的 Spearman 相关系数（保留三位小数）

变量名	相关变量	相关系数	变量名	相关变量	相关系数
开盘价	金融机构人民币贷款利率区间占比:等于 LPR	-0.943	收盘价	金融机构人民币贷款利率区间占比:等于 LPR	-0.943
最高价	金融机构人民币贷款利率区间占比:等于 LPR	-0.943	最低价	金融机构人民币贷款利率区间占比:等于 LPR	-0.943
成交量	中国战略性新兴产业采购经理指数(EPMI):当月值	-0.600	成交额	中国战略性新兴产业采购经理指数(EPMI):当月值	-0.714
				社会消费品零售总额:当月值	-0.600

5.3 特征重要性检验

寻找变量之间的关系，不仅需要考虑相关，还要考虑回归。本题提供的数据为时间序列数据，在同一时刻，传统的回归模型可能难以计算出其他变量对需要预测的变量的影响大小。因此，我们选择使用非线性模型对回归预测的重要性进行计算。

以决策树或回归树为基学习器的集成学习模型，往往都可以凭借计算增益、覆盖和频率进而计算出每个变量的特征重要性。经典的集成学习模型包含 Adaboost、GBDT、随机森林以及 XGBoost 模型。一般来说，XGBoost 由于带入了正则项，可以更好地抵御过拟合，且具有计算效率高等优点。因此，我们选用 XGBoost 算法来计算变量之间相互预测的特征重要性。

由于使用 XGBoost 进行回归而不是分类，因此选择 RMSE 作为损失函数。RMSE 表示均方根误差，是预测值与真实值的平均数的差的平方和除以 n 后再开方得到的数。其作为拟合优度的一种度量，可以更好地反映预测的离散程度。设x_i为第 i 个观测值，$\bar{x}$为真实值的平均值，n为观测次数，则定义：

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}} \tag{4}$$

1. 以五分钟为时间间隔的数据

利用 XGBoost 模型，选择较优的超参数调参拟合后，得到分别以“数字经济板块”内含的六个变量为预测变量的六个模型，其 RMSE 如下表所示。

表 7 XGBoost 分别对数字经济板块信息的变量的拟合程度

预测变量	**RMSE**
开盘价	0.002465
收盘价	0.000653
最高价	0.000645
最低价	0.000642
成交量	0.000842
成交额	0.000816

由上表可以看出，对数据进行标准化后，RMSE 均小于 0.01，这说明 XGBoost 对变量的拟合程度较好。以对成交额的预测为例，由于本题的数据是时序数据，具有前后关联的特性。因此我们选取前 90%的数据作训练集，后 10%的数据作为测试集，预测结果如下图所示。图中，红色的曲线代表预测值，蓝色的曲线代表真实值。显然两条曲线重复度较高，拟合程度较好。

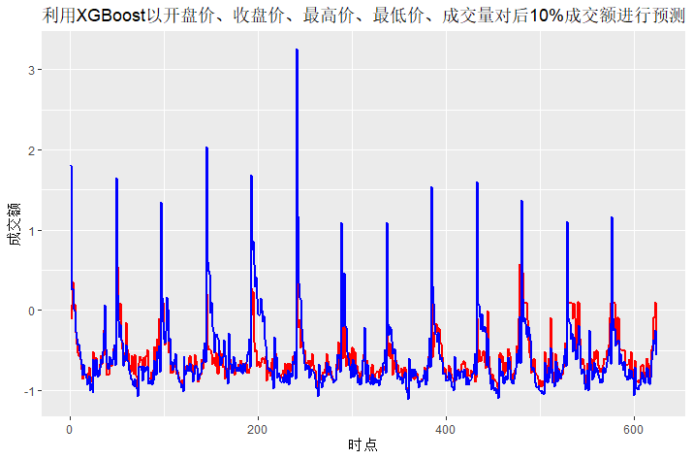


图 4 利用 XGBoost 以开盘价、收盘价、最高价、最低价和成交量对成交额预测

表 8 开盘价、收盘价、最高价、最低价和成交量对成交额预测的特征重要性

特征	增益	覆盖	频率	重要性
成交量	0.92806141	0.4928407	0.24659012	0.92806141
开盘价	0.03442206	0.1307557	0.40768340	0.03442206
收盘价	0.01582355	0.1100011	0.17449415	0.01582335
最高价	0.01423486	0.1433005	0.09211771	0.01423486
最低价	0.00745812	0.1231019	0.7911462	0.00745812

再计算出开盘价、收盘价、最高价、最低价以及成交量五个变量对预测成交额的重要程度。如上表所示，XGBoost 利用增益、覆盖和频率计算出每个变量的重要性指标，其中：

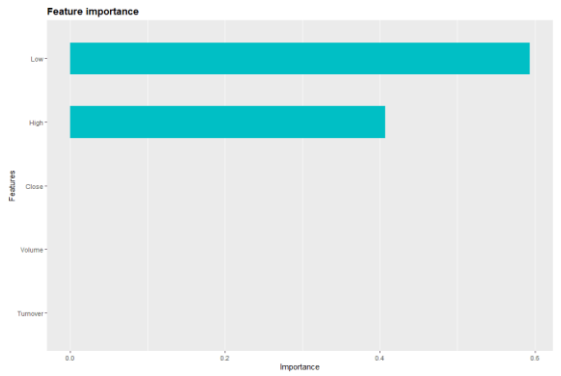
增益（Gain），是指相应的特征通过对模型中的每个树采取每个特征的贡献而计算出的模型的相对贡献。增益值越高，占的比例越大，意味着其所表示的变量对于预测变量的影响最大。

覆盖度量（Cover），是指与该特征相关的观测的相对数量。覆盖度量越高，说明该变量中所包含的成分指标在不同树上的叶节点上的观测值最多。

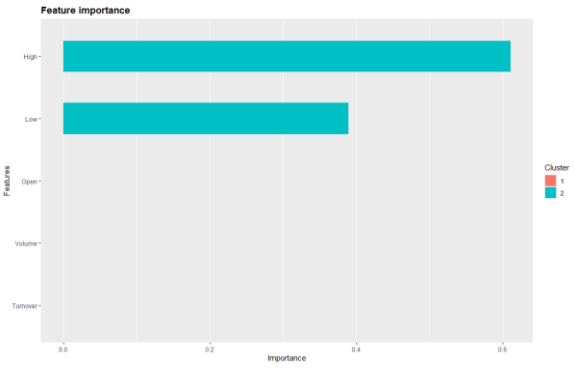
频率（Frequence），是指特定特征在模型树中发生的相对次数的百分比。变量的频率最高，说明变量占所有变量权重上的百分比权重最高。

重要性（Importance），即是综合增益、覆盖度量和频率的函数，当前三者确定时，重要性随之确定。重要性可以较为直观地体现特征对于被预测地特征的重要程度。

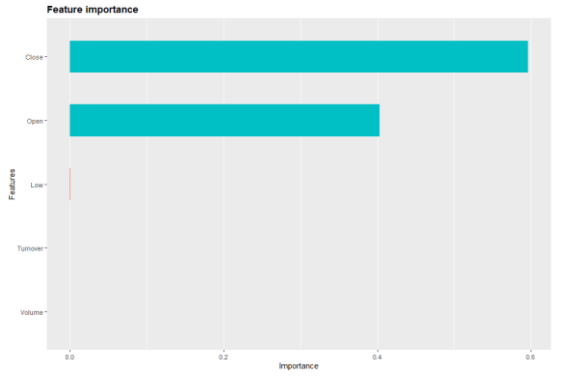
重复上述操作，将得到的重要性绘制成图像，如下所示：



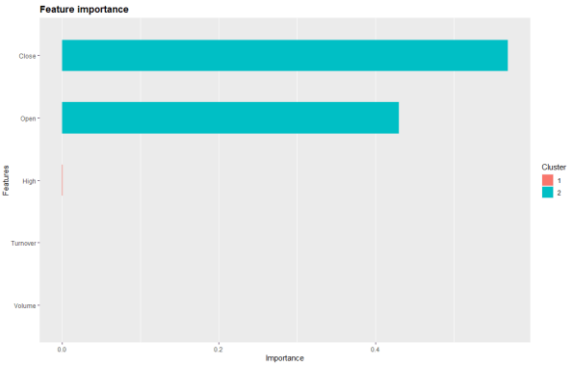
（a）开盘价的特征重要性



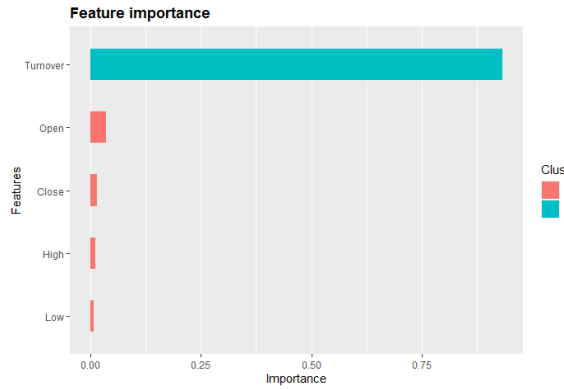
（b）收盘价的特征重要性



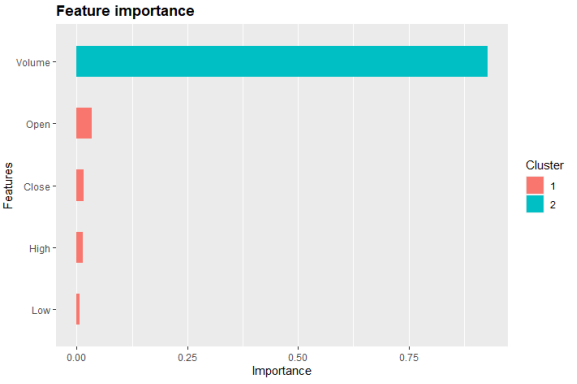
（c）最高价的特征重要性



（d）最低价的特征重要性



（e）成交量的特征重要性



（f）成交额的特征重要性

图 5 对数字经济板块信息的六个变量预测的特征重要性图像（五分钟）

图中，红色代表重要性较低，蓝绿色代表重要性高。如图所示，对于开盘价，最重要的是最低价，其次是最高价（左上）；对收盘价，最重要的也是最高价，其次是最低价（右上）；对于最高价，最重要的是收盘价，其次是开盘价（左中）；对于最低价，最重要的是收盘价，其次是开盘价（右中）；对于成交量，最重要的是成交额（左下）；对于成交额，最重要的是成交量（右下）。整理后的结果如下表所示。

表 9 以五分钟为时间间隔的数据的重要特征

预测变量	重要特征（从大到小排列）
开盘价	最低价、最高价
收盘价	最高价、最低价
最高价	收盘价、开盘价
最低价	收盘价、开盘价

成交量	成交额
成交额	成交量

2. 以日为时间间隔的数据

由于以日为间隔的变量较多，直接带入机器学习模型可能会导致运算时间较长，影响模型效率。因此，我们选用节 5.2.2 中计算出的分别和数字经济板块信息中的 6 个变量相关性较高的变量，带入 XGBoost 模型计算特征重要性，拟合结果如下表所示。

表 10 以日为时间间隔的数据的 XGBoost 模型的拟合程度

预测变量	RMSE
开盘价	0.000476
收盘价	0.000440
最高价	0.000425
最低价	0.000391
成交量	0.000604
成交额	0.000565

我们选取开盘价作为范例，绘制预测值与真实值的可视化图像，如下所示。显然，模型拟合较好。

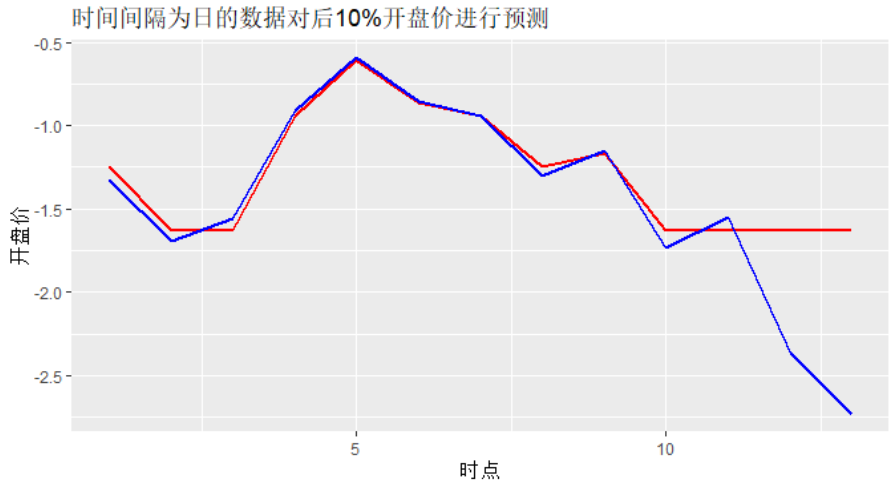
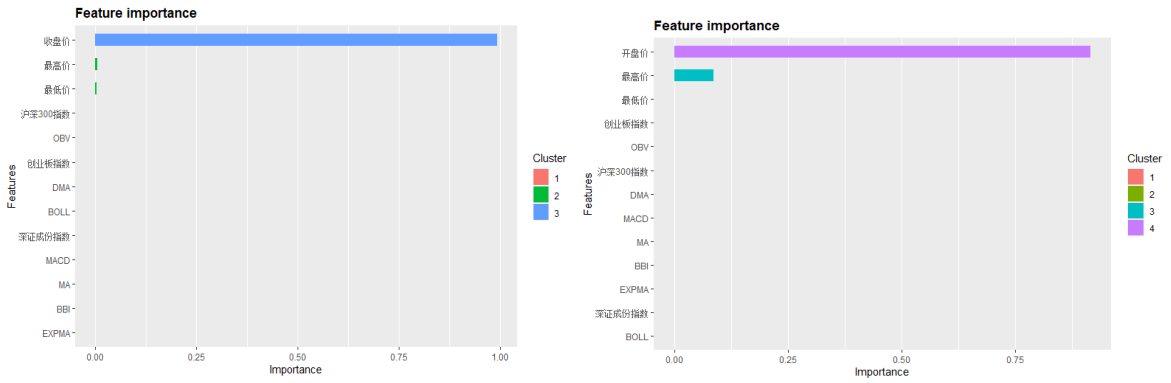
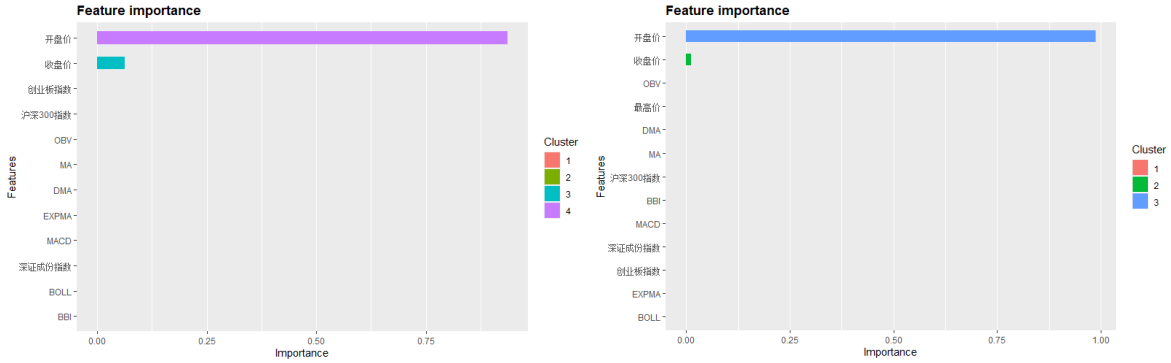


图 6 以日为时间间隔的开盘价数据的 XGBoost 预测图像

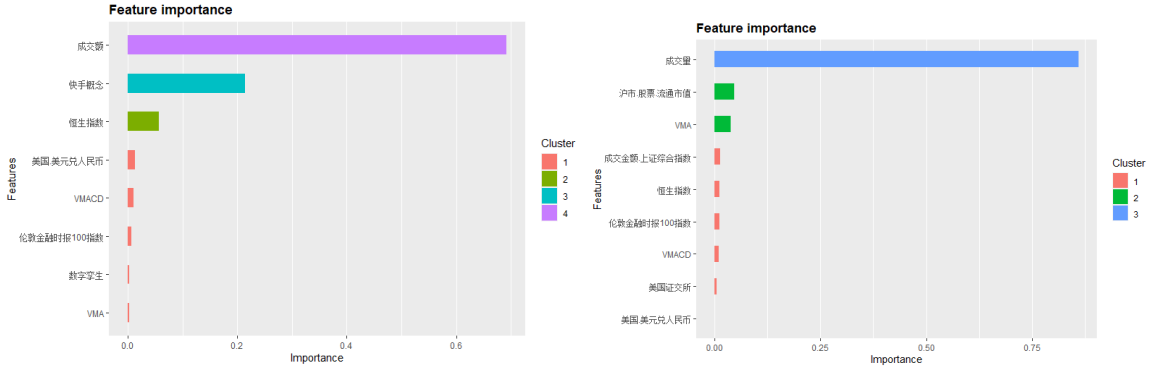
分别对数字经济板块信息中的六个变量做特征重要性计算，结果如下图表所示：



（a）开盘价的特征重要性　　（b）收盘价的特征重要性



（c）最高价的特征重要性　　（d）最低价的特征重要性



（e）成交量的特征重要性　　（f）成交额的特征重要性

图 7 对数字经济板块信息的六个变量预测的特征重要性图像（日）

图中，除图(e)存在第二个较为重要的变量以外，其他五个变量都只有一个明显最重要的变量。为判断是否需要保留“快手概念”来预测成交量，我们列出详细的重要性指标，如下表所示。成交额对于成交量的预测的重要性接近 7 成，且成交额和成交量一样是以五分钟为时间间隔的时间序列数据，而快手概念对成交量的预测重要性仅为 2 成，仅是相对于其他变量较高，且快手概念是以日为时间间隔的数据。因此，舍去变量“快手概念”，仅保留“成交额”作为成交量的重要特征。

表 11 以日为时间间隔的数据对成交量的特征重要性指标

特征	增益	覆盖	频率	重要性
成交额	0.691563246	0.49380933	0.36430318	0.691563246
快手概念	0.214531475	0.07141046	0.06308068	0.214531475
恒生指数	0.057291673	0.06004344	0.07775061	0.057291673

美国.美元兑人民币	0.013532289	0.09803641	0.10562347	0.013532289
VMACD	0.010676070	0.09803641	0.08679707	0.010676070
伦敦金融时报100 指数	0.006737332	0.04876462	0.09046455	0.006737332
数字孪生	0.003065081	0.07190659	0.09902200	0.003065081
VMA	0.002602833	0.08421076	0.11295844	0.002602833

因此，得出结论：对于开盘价，最重要的是收盘价（左上）；对于收盘价，最重要的是开盘价（右上）；对于最高价，最重要的是开盘价（左中）；对于最低价，最重要的是开盘价（右中）；对于成交量，最重要的是成交额，重要性的占比接近七成，而由于“快手概念”这一变量在对成交量的影响中，重要性仅占两成，因此不能认为“快手概念”对于成交量具有显著的重要性（左下）；对于成交额，最重要的是成交量（右下）。和以五分钟为时间间隔的数据相同，我们同样列出结果表格进行对比，如下所示：

表 12 数字经济板块信息的六个变量的重要特征（五分钟和日为时间间隔）

预测变量	重要特征（五分钟）	重要特征（日）
开盘价	最低价、最高价	收盘价
收盘价	最高价、最低价	开盘价
最高价	收盘价、开盘价	开盘价
最低价	收盘价、开盘价	开盘价
成交量	成交额	成交额
成交额	成交量	成交量

5.4 有关变量汇总结果

综合节 5.2 和节 5.3 的信息，我们将与“数字经济板块信息”有关分为三类：重要、较重要以及相关。汇总有关的指标的结果如下表所示。

表 13 数字经济板块信息的六个变量的有关特征

预测变量	重要特征	较重要特征	相关特征
开盘价	最低价	最高价	沪深 300 指数
	最高价	最低价	创业板指数
	收盘价		深证成份指数
			OBV
			BBI
			DMA
			MA
			EXPMA
			MACD
			BOLL
			金融机构人民币贷款利率区间占比:等于 LPR
收盘价	最高价	最高价	沪深 300 指数
	最低价		创业板指数
	开盘价		深证成份指数

			OBV BBI DMA MA EXPMA MACD BOLL 金融机构人民币贷款利率区间占比:等于 LPR
最高价	收盘价 开盘价	收盘价	沪深 300 指数 创业板指数 深证成份指数 OBV BBI DMA MA EXPMA MACD BOLL 金融机构人民币贷款利率区间占比:等于 LPR
最低价	收盘价 开盘价	收盘价	沪深 300 指数 创业板指数 深证成份指数 OBV BBI DMA MA EXPMA MACD BOLL 金融机构人民币贷款利率区间占比:等于 LPR
成交量	成交额	快手概念 恒生指数	VMA VMACD 恒生指数 伦敦金融时报 100 指数 美国.美元兑人民币 数字孪生 快手概念 中国战略性新兴产业采购经理指数(EPMI):当月值
成交额	成交量	沪市.股票. 流通市值 VMA	成交金额. 上证综合指数 沪市.股票. 流通市值 VMA

VMACD
美国证交所
恒生指数
伦敦金融时报
100 指数
美国.美元兑人民币
中国战略性新兴产业采购经理指数(EPMI):当月值
社会消费品零售总额:当月值

六、 问题二的求解

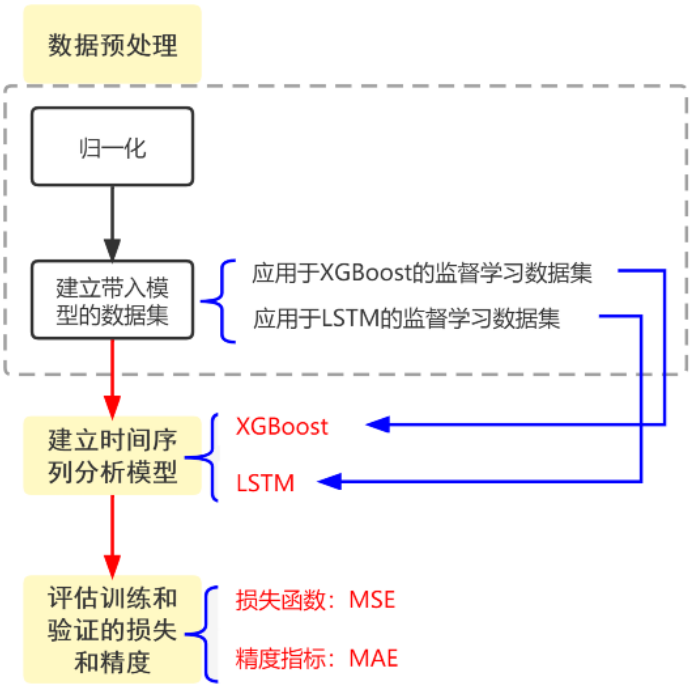


图 8 问题二、三的思维导图

6.1 数据预处理

6.1.1 归一化

成交量的最小值为 0，最大值为 884910620，取值范围为一位数到九位数之间，取值波动较大。如果直接将差距过大的数据带入，拟合能力较强的集成学习模型和神经网络模型会自适应这种波动，导致学习更加困难，预测波动更大。

为消除这种波动，采用归一化法处理数据，将每个数据的取值范围放缩到[0,1]之间。设x_{norm}为归一化后的数据，x_i为第 i 个观测值，x_{min}为变量 X 的最小值，x_{max}为变量 X 的最大值，则归一化为：

$$x_{norm} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{5}$$

6.1.2 建立带入模型的数据集

1. 应用于 XGBoost 的监督学习数据集

本题所给数据为时间序列数据，而 XGBoost 是典型的监督学习模型，无法拟合时间序列数据。为方便 XGBoost 对时间序列数据进行拟合，我们改变数据集的格式。将“成交量”“成交额”的 48 个值作为 96 个预测变量，将“成交量”的值作为预测值，建立监督学习格式的数据集。此时数据集可直接代入 XGBoost 模型并进行训练和预测。

2. 应用于 LSTM 的三维数组数据集

由于预测“成交量”时，不仅需要“成交量”本身的时间序列数据，还需要“成交额”的数据，且 LSTM 需要以过去的一段时间来预测来的一个时点。因此，对原始数据集进行拆分，将附录的表“数字经济板块信息”按照时间升序，且仅保留“成交量”“成交额”两列。

此时读取的数据格式为(6240, 2)，若希望用过去一天的数据来预测下一个数据，则进入 LSTM 的数据格式为(48, 2)，输出数据的格式为(1,1)，则需要对原始数据进行拆分。我们将原始数据集按照题目所要求的时间，拆分为格式为(5280, 2)的训练集和格式为(960, 2)的测试集。再经过每 48 行的数据打包成一个输出后，得到训练集格式为(5232, 48, 2)，测试集格式为(912, 48, 2)。此时就可以带入 LSTM 神经网络进行训练和检测。

6. 2 时间序列分析模型

1. XGBoost

XGBoost 算法属于增强法中一种提高算法，是在 GBDT 的基础上对 boosting 算法进行的改进，内部决策树使用的也是回归树，其目标函数为：

$$J(f_t) = \sum_{i=1}^{n} L\left(y_i, \hat{y}^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + C \tag{6}$$

用 Taylor 展开式有：

$$f(x + \Delta x) \cong f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2 \tag{7}$$

其中：

- 目标函数中的 L 即为损失函数（比如平方损失函数：$L(y_i, y^i) = (y_i - y^i)\text{^}2$)
- $\Omega(f_t)$为正则项，C 为常数项
- 对于$f(x)$，XGBoost 利用泰勒展开三项，做一个近似。$f(x)$表示的是其中一颗回归树。

XGBoost 的核心算法思想即为：

①不断地进行特征分裂来添加树，每次添加一个树，其实就是学习一个新函数 f(x)，去拟合前面树的预测结果与真实值的残差。

②训练完成得到 k 棵树后，根据样本的特征，在每棵树中落到对应的一个叶子节点，每个叶子节点就对应一个分数。

③最后将每棵树的预测分数相加，得到样本的预测值。

显然，我们的目标是要使得树群的预测值y_i'尽量接近真实值y_i，而且有尽量大的泛化能力。

A.目标函数第一部分：训练误差。

$$\hat{y}_i^{(0)} = 0$$

$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$

$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$

……

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

接下来，需要在每一轮选取一个 f 使得目标函数尽量最大地降低。这里 f 可以使用 Taylor 展开公式近似。

$$J(f_t) = \sum_{i=1}^{n} L\left(y_i, \hat{y}^{(t)}\right) + \sum_{i=1}^{t} \Omega(f_i)$$

$$= \sum_{i=1}^{n} L\left(y_i, \hat{y}^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + C$$

把样本分配到对应的叶子结点即得到一个 J，优化的实质就是对 J 进行优化。也就是分裂节点到叶子不同的组合，不同的组合对应不同 J。

B.目标函数第二部分：正则项。

首先定义树的复杂度：

- 一个是树里面叶子节点的个数 T。
- 一个是树上叶子节点的得分 w 的 L2 模平方（对 w 进行 L2 正则化，相当于针对每个叶结点的得分增加 L2 平滑，以避免过拟合）。

$$f_t(x) = w_{q(x)} \quad w \in R^T, q = R^d \to \{1,2,\dots,T\} \tag{8}$$

其中 w 是叶子的向量，q 为树的结构。

把树拆分成结构部分 q 和叶子权重部分 w，树的复杂度函数：

$$\Omega(f_t) = \Upsilon \cdot T_t + \lambda \cdot \frac{1}{2} \sum_{j=1}^{T} w_j^2 \tag{9}$$

C. 最终目标函数：损失函数揭示训练误差 + 正则化定义复杂度

$$J(\phi) = \sum_i L(y_i' - y_i) + \sum_k \Omega(f_t) \tag{10}$$

对于上式而言，y_i'是整个累加模型的输出，正则化项$\sum_k \Omega(f_t)$是则表示树的复杂度的函数，值越小复杂度越低，泛化能力越强。

2. LSTM

LSTM 是一种深度神经网络，属于对 RNN 的优化。标准的 RNN 网络在有用信息和处理信息的距离较大时，无法学习到有用信息，可能导致预测任务的失败。而 LSTM 网络通过构建单元内部具有四个网络层的链式结构，能解决标准 RNN 网络不能解决的长依赖问题，其神经元的结构如下所示。

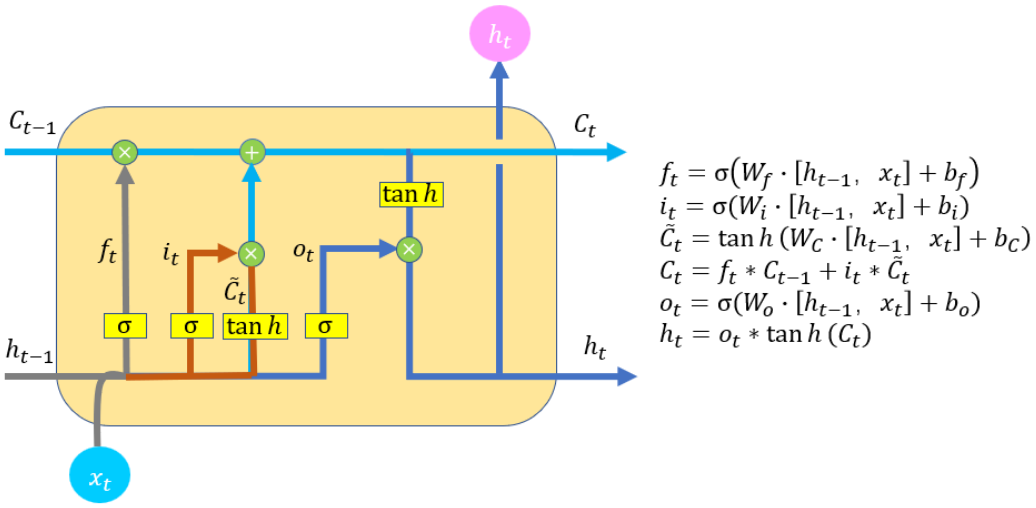


图 9 LSTM 神经元模型

如上图所示，LSTM 的内部通过“门”来控制传输状况，从而丢弃或者增加信息，最终实现遗忘或记忆的功能。门由一个σ函数和一个点乘操作组成，σ函数的输出值在[0，1]区间，1 时完全通过，0 时完全丢弃。每个 LSTM 单元有三个门，分别是“遗忘门”、“输入门”和“输出门”。“遗忘门”是由上一单元输出的h_{t-1}和这一单元输入的x_t构成一个σ函数，为C_{t-1}中的每一项产生一个区间在[0，1]之间的值，来控制上一单元的遗忘程度。“输入门”和一个$\tan h$函数控制被加入的信息，并更新本单元的信息。最后将单元激活后，“输出门”会给每一项产生一个[0，1]的值，来过滤信息。

6.3 模型建立与求解

为评估训练和验证的损失和精度，我们 MSE 作为损失函数，以 MAE 作为精度指标。其中，MSE 为均方误差，指的是预测值与真实值的差的平方和除以 n。这是由于在训练过程中，我们需要对每一个样本的误差求和，直接相减可能会导致和为 0。设Y_i为第 i 个真实值，$\hat{Y}_i$为第 i 个预测值，则 MSE 为：

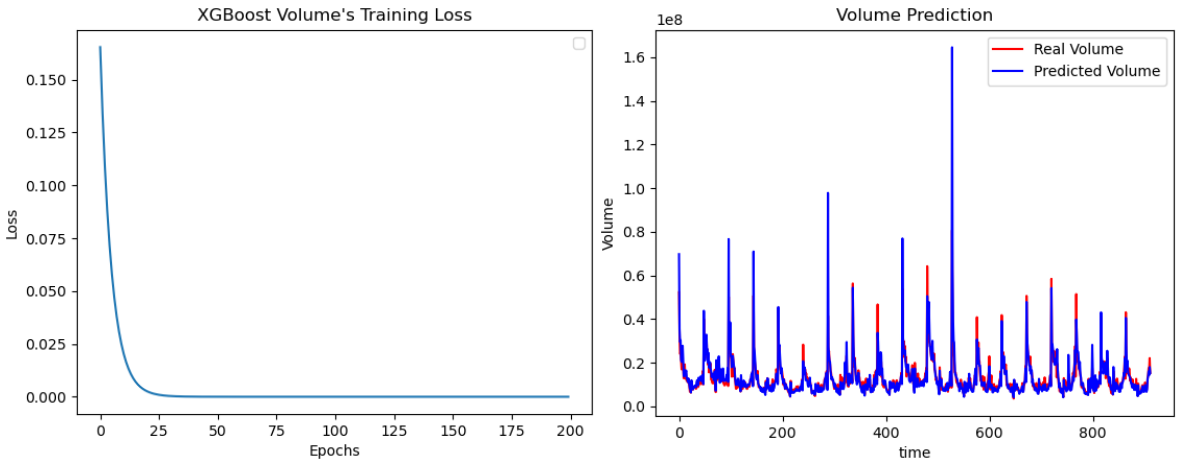
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{11}$$

MAE 为绝对平均误差，是绝对误差的平均值，也可以说是更一般形式的误差均值。与均方误差不同的是，绝对平均误差采用绝对值来防止正负误差相互抵消。MAE 可以直观体现预测值与目标值之间的差值的绝对值。定义 MAE 为：

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i| \tag{12}$$

6.3.1 XGBoost

由于本题是回归预测，因此将 XGBoost 的策略设置为回归。给学习率、树的深度寻找到较好的超参数后，设置迭代次数为 200，训练精度如下图(a)所示，对 2022 年 1 月 4 日到 2022 年 1 月 28 的预测结果如下图(b)所示。



(a)XGBoost 的 MSE　　(b)XGBoost 的预测效果

图 10 XGBoost 的训练误差和预测结果

训练集的 MSE 为1.3476×10^{-7}，MAE 为2.7593×10^{-4}，测试集的 MSE 为2.4299×10^{-5}，MAE 为2.0271×10^{-3}。显然 XGBoost 的拟合能力极强，将损失函数降低到了极低的水准，预测误差极小。

6.3.2 LSTM

训练集含有 5328 条数据，测试集含有 912 条数据，带入模型进行拟合。由于数据量较少，为防止过拟合，我们建立较小的神经网络进行预测。

经过训练，我们输出对成交量在 2022.1.4-2022.1.28 的 912 条数据的预测值与真实值进行比较，绘制可视化图像，如下所示：

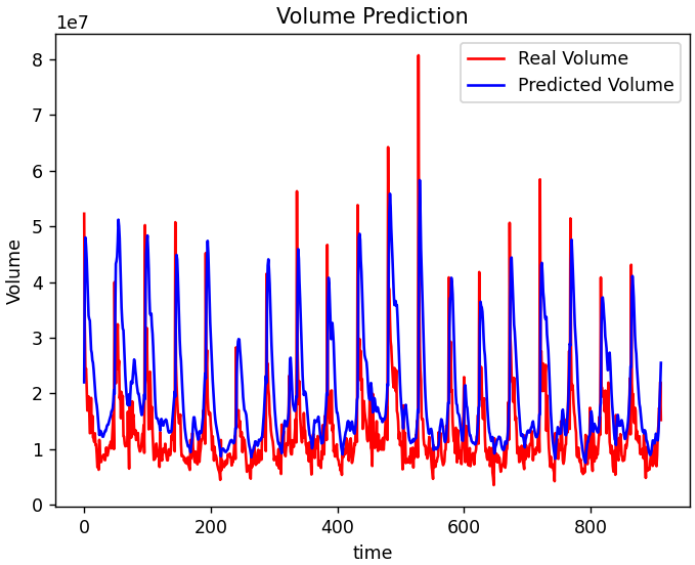


图 11 LSTM 对成交量在 2022.1.4-2022.1.28 的预测值与真实值的比较

图中，红色表示成交量的真实值，蓝色表示成交量的预测值。在趋势上，LSTM 预测的数值和真实值较为相同，但是在成交量取值较少时，误差稍大。

为比较误差，输出训练损失和预测损失、训练精度和预测精度的图像，如下所示。

图 12 LSTM 对成交量的训练数据和测试数据的验证损失和精度

显然，随着迭代次数的升高，MSE 和 MAE 总体趋势是不断减小，预测的精度不断提高。当迭代到第十轮后，MAE 的值为 0.0132，误差已经很小。由于成交量的量纲过大，带来的误差也就较大，但是在比例上而言已经相对较小。

但是，上图所示的 MSE 和 MAE 并不是绝对递减，这说明到达某一轮次时，模型可能开始过拟合。为避免这种现象，我们对模型设置早停条件：当 MAE 连续两轮增加时，停止拟合模型，保留上一轮的模型。同时，我们在神经网络之中带入 dropout 层来减轻过拟合的影响。第二次拟合的 LSTM 的预测结果和 MSE、MAE 如下所示。

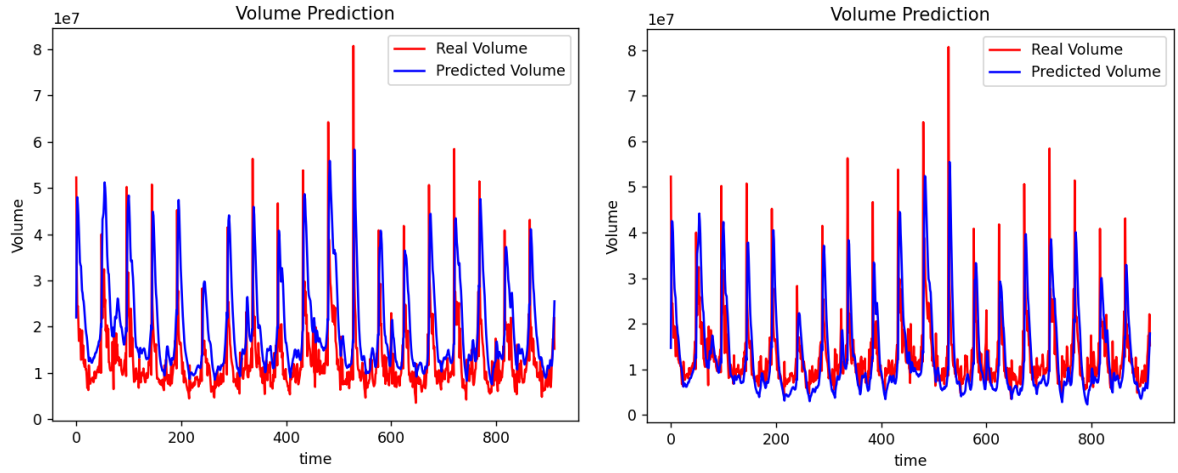


图 13 LSTM（左）以及早停的 LSTM（右）对成交量的预测比较

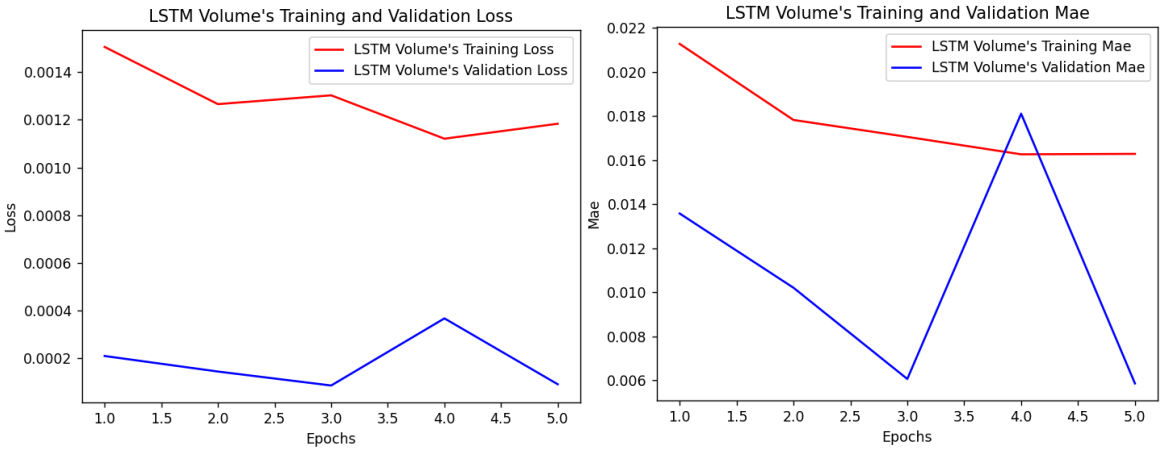


图 14 LSTM（早停）对成交量的训练数据和测试数据的验证损失和精度

设置早停条件后，模型在第五轮自动停止，并且输出图像。此时 MSE 和 MAE 呈更加明显的递减趋势。拟合结束后，训练集的 MSE 为 0.0012，MAE 为 0.0163，测试集的 MSE 为 0.00008979，MAE 为 0.0059。

虽然 XGBoost 的拟合效果更强，但是对于本题的数据，XGBoost 的过拟合较为严重，且预测出了异常值。因此，综合考虑之下，我们认为 LSTM 的预测值和真实值明显贴合度更高，预测效果更好，保留 LSTM 的预测结果作为最终结果。

七、 问题三的求解

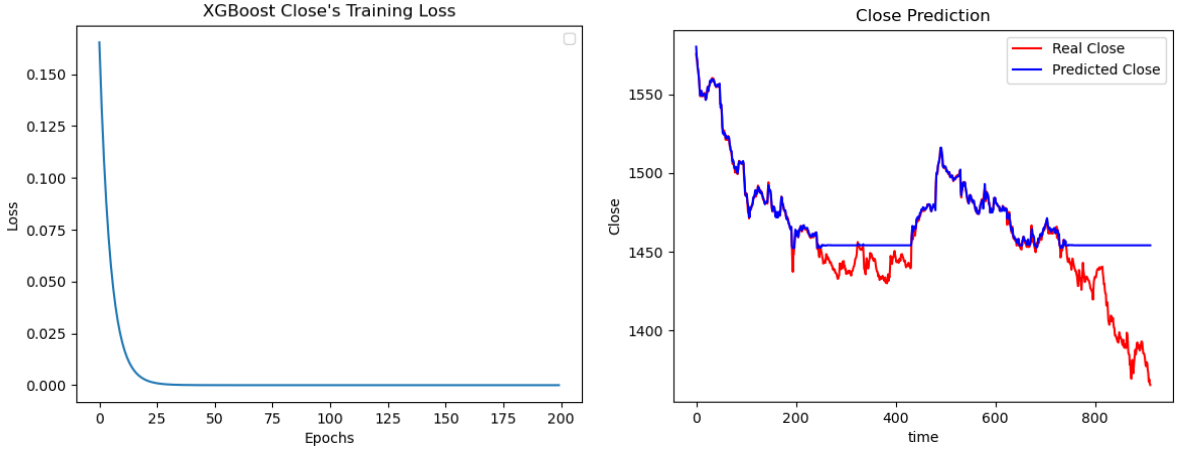
7.1 数据预处理

与节 6.1 相同，对预测目标的收盘价以及最高价、最低价、开盘价左归一化。同时分别建立应用于 XGBoost 和 LSTM 的数据集。

7.2 时间序列分析模型的求解

7.2.1 XGBoost

利用问题二已经决定好的模型，设置较优的超参数，结果如下所示。



(a)XGBoost 的 MSE　　(b)XGBoost 的预测效果

图 15 XGBoost 的训练误差和预测结果

训练集的 MSE 为1.4560×10^{-7}，MAE 为2.9902×10^{-4}，测试集的 MSE 为4.7569×10^{-3}，MAE 为0.03462。

同时，由于数据本身的特点，训练集中没有低于 1450 的数据，因此拟合能力极强的 XGBoost 没有预测低于 1450 的数据的能力，过拟合较为严重，预测结果较差。

7.2.2 LSTM

拟合带有 dropout 层和设置早停的 LSTM 模型，预测值和真实值的比较、MSE 和 MAE 的图像如下所示。

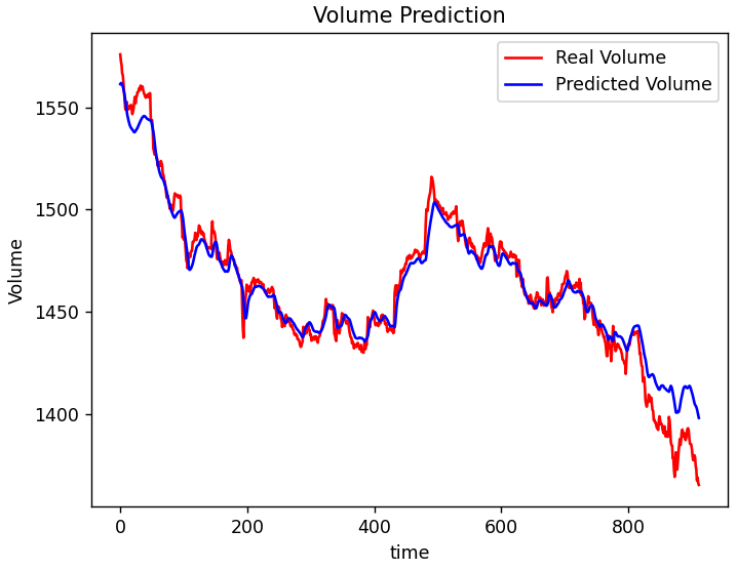


图 16 LSTM 对收盘价在 2022.1.4-2022.1.28 的预测值与真实值的比较

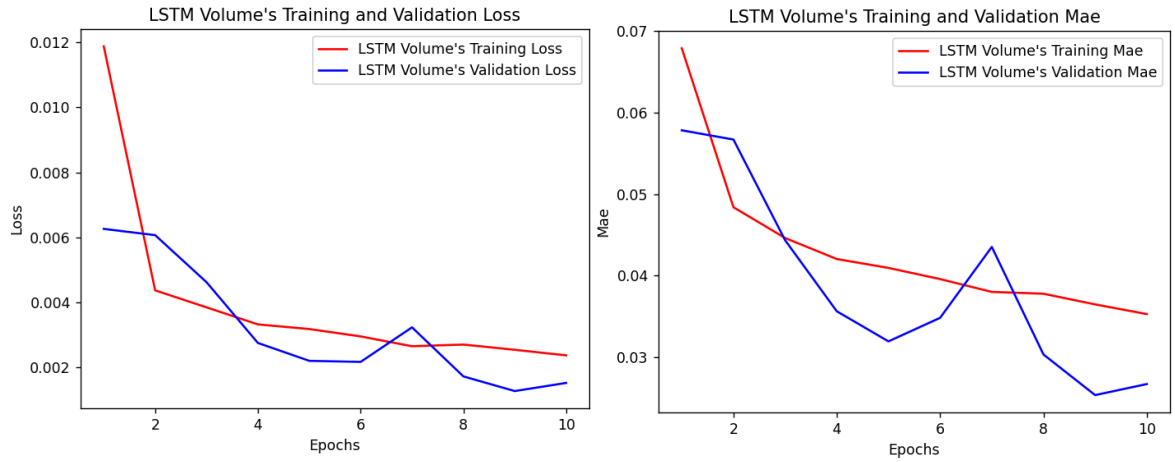


图 17 LSTM（早停）对成交量的训练数据和测试数据的验证损失和精度

图 16 中，红色表示收盘价的真实值，蓝色表示收盘价的预测值。在趋势上，LSTM 预测的数值和真实值整体相同，拟合程度较好。但在接近 2022 年 1 月 28 号的一段时间内模型的预测结果较差，这是由于 2022 年 1 月末股市出现大跌的现象，该现象在模型中属于异常值，因此模型难以预测这段时间的数值。

图 17 中显示，在设置早停的 LSTM 模型中，模型在第十轮自动停止。拟合过程中，MSE 和 MAE 呈较明显的递减趋势。当神经网络拟合结束后，训练集的 MSE 为

0.0024，MAE 为 0.0352，测试集的 MSE 为 0.0015，MAE 为 0.0266，显然，LSTM 的预测值和真实值贴合度较高，预测效果较好。

八、 问题四的求解

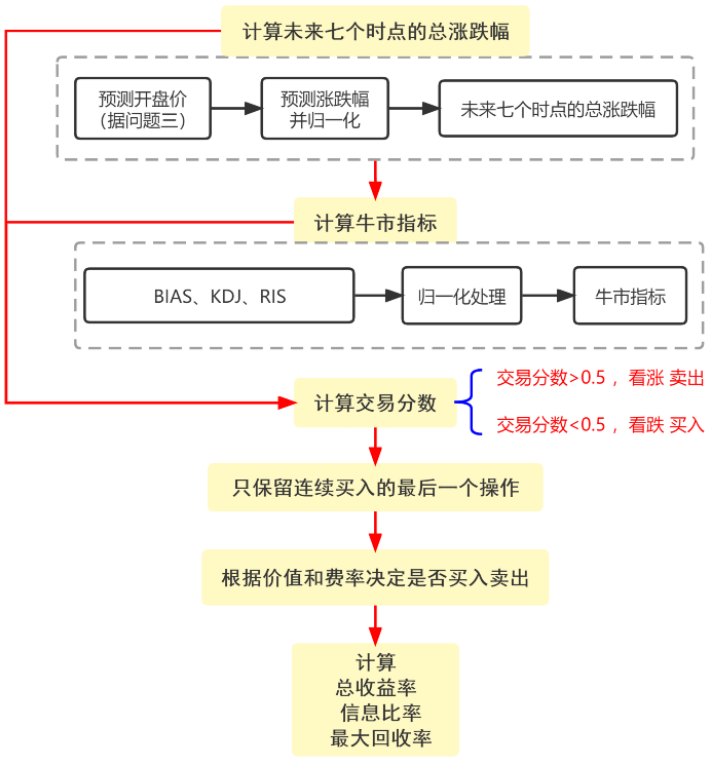


图 18 问题四的流程图

8.1 交易策略指标

基于第三问的预测结果，以及附表中表“技术指标”中的数据，我们设置如下交易策略指标。

- **实际收盘价 Y**

  以附录中 2022 年 1 月 4 日到 2022 年 1 月 28 日收盘价的真实值作为实际收盘价，用Y表示。每一个收盘价的观测值用y_i表示。

- **实际涨跌幅 C**

  根据实际收盘价Y计算实际涨跌幅，设第i个收盘价为y_i，第$i+1$个收盘价为y_{i+1}，则设实际涨跌幅C为：

$$c_i = \frac{y_{i+1} - y_i}{y_i} \tag{13}$$

- **预测收盘价 $\hat{Y}$**

  以第三问对收盘价的预测结果作为预测收盘价，用$\hat{Y}$表示。其中，每一个预测收盘价的观测值为$\hat{y}_i$。

- **预测涨跌幅 $\hat{C}$**

与实际涨跌相同，设第i个预测收盘价为$\hat{y}_i$，第$i+1$个预测收盘价为$\hat{y}_{i+1}$，则预测涨跌幅$\hat{C}$为：

$$\hat{c}_i = \frac{\hat{y}_{i+1} - \hat{y}_i}{\hat{y}_i} \tag{14}$$

- **未来七个时点的总涨跌幅 R_i**

设第i个预测涨跌幅为$\hat{c}_i$，并设$\hat{c}_{i+1}$为第$i+1$个预测的涨跌幅，以此类推。则未来七个时点的涨跌幅为：

$$R_i = \prod_{i=1}^{7}(1 + \hat{c}_i) \tag{15}$$

- **BIAS**

BIAS（乖离率）又称偏离率，即收盘价与某条移动平均线之间的差距百分比。BIAS 越高，股价与其移动平均数的偏离程度越高。

- **KDJ**

KDJ 又称随机指标，即根据一个周期内的最高价、最低价、收盘价（最后一个计算周期的）以及三者的比例关系，来计算最后一个计算周期的未成熟随机值 RSV，从而利用 RSV 得到 K 值、D 值和 J 值，以反映市场价格走势的强弱。

- **RIS**

RSI 又称相对强弱指标，是一定时期内上涨点数和下跌点数之和的比率，常被用于测量和分析股票的升跌。

- **牛市分数 B_i**

设R_{norm}、$BIAS_{norm}$、KDJ_{norm}和RSI_{norm}分别表示四个指标归一化后的数据，由于这四个指标均能反应涨跌情况，因此我们给予它们相同的权重，计算牛市分数为

$$B_i = \frac{R_{norm} + BIAS_{\text{norm}} + KDJ_{norm} + RSI_{norm}}{4} \tag{16}$$

- **交易分数 S_i**

利用牛市分数B_i以及归一化后的R_{norm}作为买入和卖出的判断。经计算，我们给归一化后的未来七个时点的总涨跌幅更高的权重，给牛市分数较低的权重，计算出的交易分数S_i为：

$$S_{\text{i}} = \frac{2}{3}R_{norm} + \frac{1}{3}B_i \tag{17}$$

8.2 交易模型的建立、交易及相关指标的计算

根据牛市分数B_i，已经可以较好地反应涨跌情况，我们将$B_i > 0.5$的时点作为牛市，并标注到收盘价曲线上，如下所示：

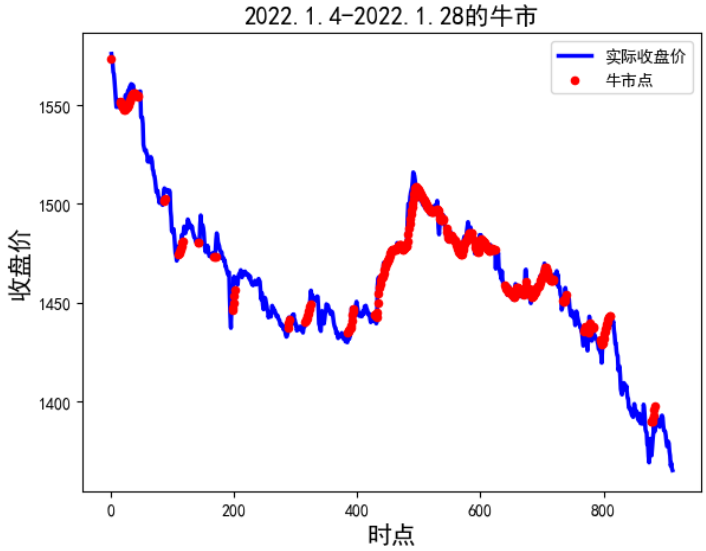


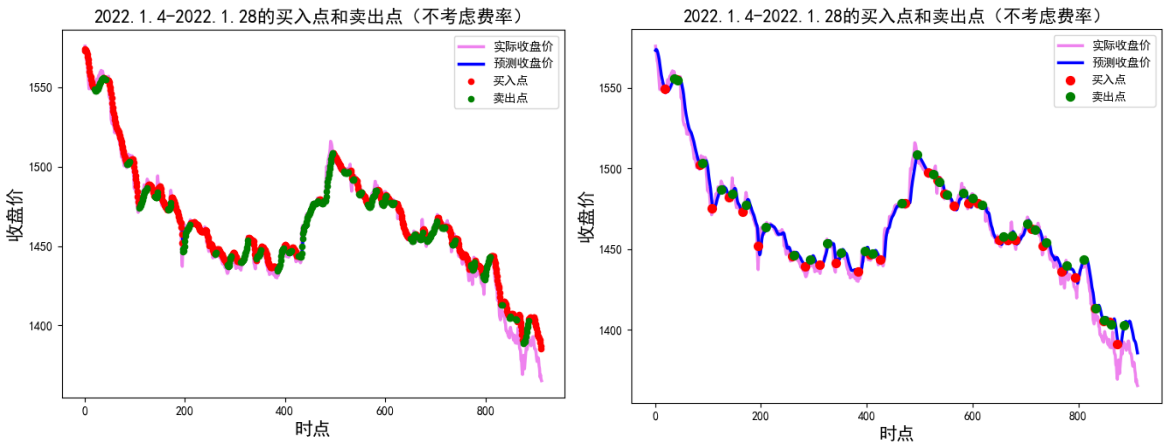
图 19 2022.1.4-2022.1.28 的牛市点

显然，单凭借牛市点来决定买入和卖出并不合理。在牛市分数B_i的技术上，根据交易分数S_i，我们对 2022 年 1 月 4 日至 2022 年 1 月 28 日共 912 个时点做出交易决策D_i。由于交易分数S_i可以较好地反应数字经济的涨跌情况，且S_i已经经过归一化，取值在[0,1]之间。当$S_i > 0.5$时，说明数字经济整体看涨，此时应当卖出持有的份额套现盈利；当$S_i < 0.5$时，说明数字经济整体看跌，此时应当买入成本较低的份额。

因此，设决策变量D_i为-1 时，选择卖出。决策变量D_i为+1 时，选择买入，则第i个时点的决策为D_i为：

$$D_i = \begin{cases} -1, S_i \geq 0.5 \\ +1, S_i < 0.5 \end{cases} \tag{18}$$

绘制共 912 个买入点和卖出点，如下所示：

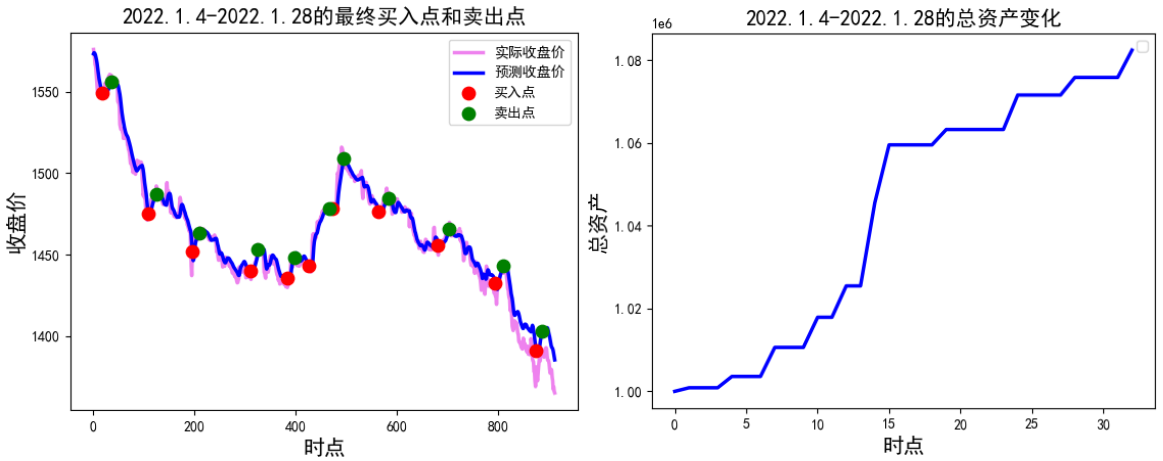


(a)连续买入卖出　　(b)剔除连续点

图 20 2022.1.4-2022.1.28 的买入点和卖出点

如上图(a)所示，红色为执行买入操作的点，绿色为执行卖出操作的点，显然并不需要操作 912 次。买入点表示看跌，卖出点表示看涨。因此，当出现连续买入或连续卖出的情况时，我们仅保留最后的买入点和卖出点，这两个点分别对应买入价格最低和卖出价格最高的情况。剔除连续的买入点和卖出点后的图像如上图(b)所示。

此时，对 32 组共 64 个买入点和卖出点进行分析。带入买入时的“收盘价”以及卖出时的“收盘价”，同时扣除交易费率，计算出每次操作后的收益。当收益大于成本时，保留本次操作，否则，剔除本次操作。最终的买入点和卖出点如下图(a)所示，总资产的变化如下图(b)所示。



(a)最终买入和卖出点　　(b)总资产变化

图 21 2022.1.4-2022.1.28 的最终决策和结果

将每天的 48 个时点的数据统一，计算总收益率、信息比率和最大回撤，结果如表 14 所示。所使用的所有指标的数值的结果如图 18 所示。

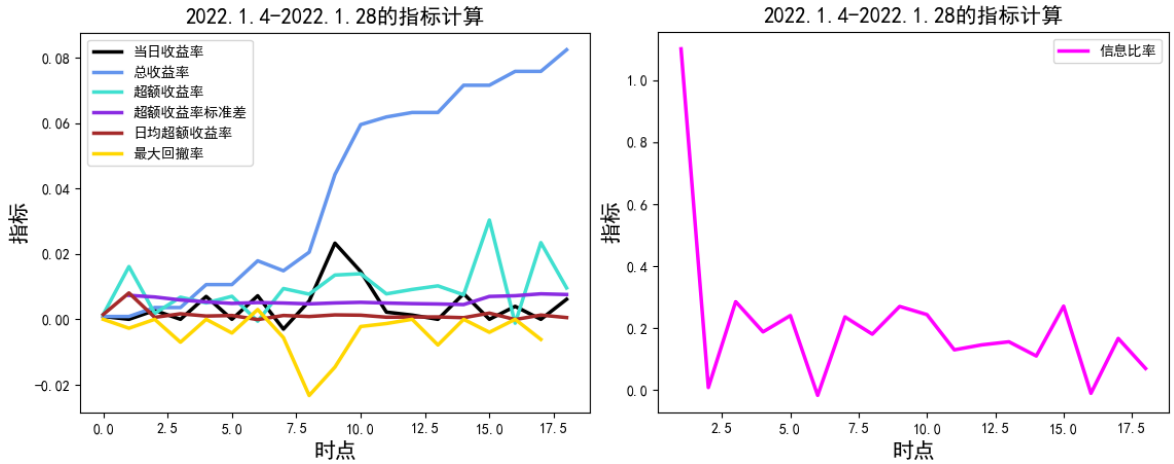


图 22 2022.1.4-2022.1.28 的各项指标

表 14 1.4-1.28 的总收益率、信息比率和最大回撤

日期	总收益率	信息比率	最大回撤
2022-01-04	0.000869	NULL	0
2022-01-05	0.000869	1.100557	-0.002729
2022-01-06	0.003600	0.0088931	0

2022-01-07	0.003600	0.285667	-0.006992
2022-01-10	0.010617	0.188294	0
2022-01-11	0.010617	0.240828	-0.004183
2022-01-12	0.017889	-0.016377	0.002991
2022-01-13	0.014845	0.236434	-0.005584
2022-01-14	0.020512	0.180863	-0.023286
2022-01-17	0.044275	0.270195	-0.014599
2022-01-18	0.059521	0.243833	-0.002204
2022-01-19	0.061856	0.13038	-0.001292
2022-01-20	0.063228	0.146312	0
2022-01-21	0.063228	0.156307	-0.00783
2022-01-24	0.071553	0.110606	0
2022-01-25	0.071553	0.271156	-0.003962
2022-01-26	0.075799	-0.009724	0
2022-01-27	0.075799	0.167159	-0.006135
2022-01-28	0.082399	0.070282	NULL

如上表所示，最后 19 次交易日的总收益率为 8.23%，年化收益率为 359.91%，收益十分可观。信息比率大于 0，说明投资产品跑赢了中证五百指数。最大回撤大部分小于 0，仅有一天大于 0，也说明投资产品的稳定性较好，风险系数较低。

九、 模型评价

9.1 模型的优点

1. 讨论变量之间的“有关”性时，不仅考虑相关性，还利用 XGBoost 算法输出特征重要性。
2. 选用的模型为集成学习模型的 XGBoost 模型和 LSTM 深度神经网络，拟合效果远好于经典统计模型。
3. 对附录的所有信息都进行了数据挖掘，提取了附录数据集所包含的大部分信息，且大部分信息在解题过程中被证实是有效的。
4. 使用带有正则项的 XGBoost 和带有 Dropout 层的早停 LSTM 模型，有效地防止过拟合问题的产生。

9.2 模型的缺点

1. 预测时只是用了最重要的变量，省略了较重要和相关的变量，可能遗漏了某些信息。

十、 参考文献

[1] 沈晨昱. XGBoost 原理及其应用[J]. 计算机产品与流通, 2019(3):1.
[2] 任君. 基于 SVM 和 LSTM 网络预测的股票量化投资模型[D]. 武汉理工大学.
[3] 蔡高远. 基于机器学习的大类资产量化投资模型研究[D]. 河南大学.
[4] 孙瑞奇. 基于 LSTM 神经网络的美股股指价格趋势预测模型的研究[D]. 首都经济贸易大学.
[5] 陆泽楠, 商玉林. 基于 LSTM 神经网络模型的钢铁价格预测[J]. 科技视界, 2017(13):2.

十一、 附录

附录 1：（SPSS）

SPSS 做柯尔莫戈洛夫-斯米诺夫正态性检验：

“分析”——“描述统计”——“探索”——“确认”

附录 2：（Python）

拉格朗日插值法即结果可视化的绘图：

```
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
from matplotlib.font_manager import FontProperties
import matplotlib.pyplot as plt
#支持在绘图中使用中文

x = [1,2,3,4,6,7,8,9]
y = [4713.07,4726.35,4659.03,4662.85,
      4577.11,4532.76,4482.73,4397.94]
def lagrange(x, y, num_points, x_test):
    l = np.zeros(shape=(num_points, ))
    for k in range(num_points):
        l[k] = 1
        for k_ in range(num_points):
            if k != k_:
                l[k] = l[k]*(x_test-x[k_])/(x[k]-x[k_])
            else:
                pass
    L = 0
    for i in range(num_points):
        L += y[i]*l[i]
    return L

x_test = [5]
y_predict = [lagrange(x, y, len(x), x_i) for x_i in x_test]
print(y_predict)

# plt.title("国际市场指标在 2022.1.17 的插值", fontsize=15)
# plt.xlabel("时点", fontsize=15)
# plt.ylabel("指数", fontsize=15)
# x = [i for i in range(1,10)]
# y1 = [36252.02 ,36290.32 ,36113.62 ,35911.81, 35674.863000000005
```

```
#           , 35368.47 ,35028.65 ,34715.39 ,34265.37]
# plt.scatter(x, y1, s = 20, color = "blue") #s 为点的大小
# plt.scatter([5], 35674.863000000005, s = 50, color = "red") #s 为点的大小
# plt.plot(x, y1, color = "royalblue", linewidth = 2.5,
#                 label = "道琼斯工业指数")

# y2 = [15153.45,15188.39,14806.81,14893.75,
# 14801.933571428575,
# 14506.90,14340.26,14154.02,13768.92]
# plt.scatter(x, y2, s = 20, color = "blueviolet") #s 为点的大小
# plt.scatter([5], 14801.933571428575, s = 50, color = "red") #s 为点的大小
# plt.plot(x, y2, color = "violet", linewidth = 2.5,
#                 label = "纳斯达克综合指数")



# y3 = [4713.07,4726.35,4659.03,4662.85,
# 4637.5610000000015,
# 4577.11,4532.76,4482.73,4397.94]
# plt.scatter(x, y3, s = 20, color = "dodgerblue") #s 为点的大小
# plt.scatter([5], 4637.5610000000015, s = 50, color = "red") #s 为点的大小
# plt.plot(x, y3, color = "cyan", linewidth = 2.5,
#                 label = "标准普尔 500 指数")

# plt.legend()
# plt.show()
```

LSTM 神经网络的拟合和预测：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import RMSprop
import tensorflow as tf

""" ====== 选择预测变量 ======
如果预测成交量，只需要成交额和其本身
如果预测收盘价，只需要开盘价和其本身
"""
# data = pd.read_csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/5min.csv")
# print(data.head())
# print(data.shape)
# df = data
```

```
    # df = df.drop(["Open", "Close", "High", "Low"], axis = 1)
    # print(df.head())
    # print(df.shape)
    # n = 2

    data = pd.read_csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/5min3.csv")
    print(data.head())
    print(data.shape)
    df = data
    df = df.drop(["Volume", "Turnover"], axis = 1)
    print(df.head())
    print(df.shape)
    n = 4


    """ ====== 划分训练集和测试集 ======
    7.4-12.31 为训练集，1.4-1.28 为测试集，测试集共 912 + 48 条数据，这样可以预测 912 个数据
    """
    df_train = df[:5280]
    df_test = df[5280:]
    print(df_train.shape)
    print(df_test.shape)

    """ ====== 归一化数据 ====== """
    scaler = MinMaxScaler(feature_range = (0, 1))
    df_train_scaled = scaler.fit_transform(df_train)
    df_test_scaled = scaler.transform(df_test)
    print(df_train_scaled)
    print(df_test_scaled)

    """ ====== 拆分数据集为 XY ====== """
    def XY(data, lookback = 48):
        X = []
        Y = []
        for i in range(lookback, len(data)):
            X.append(data[i - lookback:i, 0:data.shape[1]])
            Y.append(data[i, 0])
        return np.array(X), np.array(Y)
    trainX, trainY = XY(df_train_scaled)
    testX, testY = XY(df_test_scaled)
    print(trainX.shape)
    print(trainY.shape)
    print(testX.shape)
```

```
    print(testY.shape)

    """ ====== 建立 LSTM 模型 ====== """
    model = Sequential()
    model.add(LSTM(128,
                   dropout = 0.1,
                   recurrent_dropout = 0.5,
                   return_sequences = True,
                   input_shape = (48, n)))
    model.add(LSTM(64,
                   activation = "relu",
                   dropout = 0.1,
                   recurrent_dropout = 0.5))
    model.add(Dropout(0.1))
    model.add(Dense(1))
    model.compile(optimizer = RMSprop(),
                  loss = "mse",
                  metrics = ["mae"])
    my_callbacks = [
        tf.keras.callbacks.EarlyStopping(patience=2),
        #tf.keras.callbacks.ModelCheckpoint(filepath='model.{epoch:02d}-
{val_loss:.2f}.h5'),
        #tf.keras.callbacks.TensorBoard(log_dir='./logs')
    ]
    history = model.fit(trainX, trainY,
                        steps_per_epoch = 100,
                        epochs = 20,
                        validation_data = (testX, testY),
                        callbacks = my_callbacks
                        )

    prediction = model.predict(testX)


    """ ====== 逆中心化 ====== """
    pre_array = np.repeat(prediction, n, axis = -1)
    pred = scaler.inverse_transform(np.reshape(pre_array, (len(prediction), n)))[:,0]
    ori_array = np.repeat(testY, n, axis = -1)
    ori = scaler.inverse_transform(np.reshape(ori_array, (len(testY), n)))[:,0]


    """ ====== 可视化预测结果 ====== """

    plt.plot(ori, color = "red", label = "Real Close")
```

```
plt.plot(pred, color = "blue", label = "Predicted Close")
plt.title("Close Prediction")
plt.xlabel("time")
plt.ylabel("Close")
plt.legend()
plt.show()

""" ====== 绘制验证损失和测试损失 ====== """
plt.clf()
history_dict = history.history
loss_values = history_dict["loss"]
val_loss_values = history_dict["val_loss"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, loss_values, "r", label = "LSTM Close's Training Loss")
plt.plot(epochs, val_loss_values, "b", label = "LSTM Close's Validation Loss")
plt.title("LSTM Close's Training and Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()

""" ====== 绘制 MAE ====== """
plt.clf()
mae_values = history_dict["mae"]
val_mae_values = history_dict["val_mae"]
epochs = range(1, len(loss_values) + 1)
plt.plot(epochs, mae_values, "r", label = "LSTM Close's Training Mae")
plt.plot(epochs, val_mae_values, "b", label = "LSTM Close's Validation Mae")
plt.title("LSTM Close's Training and Validation Mae")
plt.xlabel("Epochs")
plt.ylabel("Mae")
plt.legend()
plt.show()

""" ====== 写出结果 ====== """
with open("C:/Users/Lenovo/Desktop/2022 华中杯/B/Q2_LSTM.txt", "w") as f:
    for i in pred:
        file = f.writelines(str(i)+"\n")
```

投资策略模型：

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pylab import *
```

```
from matplotlib.font_manager import FontProperties
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus'] = False
#支持在绘图中使用中文

df = pd.read_csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/Q4.csv",
                         encoding = "utf-8")


buy_threshold = 0.5
sell_threshold = 0.5
bull_threshold = 0.5

x = [i for i in range(1, 913)]
bull_x = []
bull = []
buy_x = []
buy = []
sell_x = []
sell = []
for i in range(0, 912):
    if df["牛市分数"][i] >= bull_threshold:
        bull_x.append(i)
        bull.append(df["预测收盘价"][i])
    if df["买入分数"][i] <= buy_threshold:
        buy_x.append(i)
        #buy.append(df["预测收盘价"][i])
    if df["买入分数"][i] >= sell_threshold:
        sell_x.append(i)
        #sell.append(df["预测收盘价"][i])

""" ======  当连续买入没有卖出时，保留最后一个，卖出同理 ====== """
# with open("C:/Users/Lenovo/Desktop/2022 华中杯/B/Q4_buy.txt", "w") as file:
#     for i in buy_x:
#         f = file.writelines(str(i)+"\n")
# with open("C:/Users/Lenovo/Desktop/2022 华中杯/B/Q4_sell.txt", "w") as file:
#     for i in sell_x:
#         f = file.writelines(str(i)+"\n")

buy_x = [19,41,84,108,139,165,195,
         259,284,311,342,383,407,
         426,472,516,534,547,564,
         592,610,650,666,681,712,
```

```
                733,769,794,831,847,859,874]
sell_x = [37,42,90,125,146,172,210,
                262,293,326,351,397,
                412,466,495,526,537,552,
                583,601,618,659,674,703,
                719,739,778,810,833,850,
                862,886]
for i in buy_x:
    buy.append(df["预测收盘价"][i])
for i in sell_x:
    sell.append(df["预测收盘价"][i])



# plt.title("2022.1.4-2022.1.28 的牛市", fontsize=15)
# plt.xlabel("时点", fontsize=15)
# plt.ylabel("收盘价", fontsize=15)
# plt.scatter(bull_x, bull, s = 20, color = "red", label = "牛市点", zorder = 2)
# plt.plot(x, df["实际收盘价"], color = "blue", linewidth = 2.5,
#          label = "实际收盘价", zorder = 1)
# plt.legend()
# plt.show()


# plt.title("2022.1.4-2022.1.28 的买入点和卖出点（不考虑费率）", fontsize=15)
# plt.xlabel("时点", fontsize=15)
# plt.ylabel("收盘价", fontsize=15)
# plt.scatter(buy_x, buy, s = 50, color = "red", label = "买入点", zorder = 2)
# plt.scatter(sell_x, sell, s = 50, color = "green", label = "卖出点", zorder = 3)
# plt.plot(x, df["实际收盘价"], color = "violet", linewidth = 2.5,
#          label = "实际收盘价", zorder = 1)
# plt.plot(x, df["预测收盘价"], color = "blue", linewidth = 2.5,
#          label = "预测收盘价", zorder = 1)
# plt.legend()
# plt.show()

""" ======  比较 32 次操作的收益，判断是否需要操作  ====== """
P = 1000000
P_list = [1000000]
buy_true_x = []
sell_true_x = []
for i in range(0, len(buy_x)):
    index_buy = buy_x[i]
    index_sell = sell_x[i]
```

```
            buy_price = df["实际收盘价"][index_buy]
            sell_price = df["实际收盘价"][index_sell]
            share = P/(1.003*buy_price)
            P_c = share*sell_price*(0.997)
            if P_c > P:
                P = P_c
                buy_true_x.append(index_buy)
                sell_true_x.append(index_sell)
            P_list.append(P)


    buy_true = []
    sell_true = []
    for i in buy_true_x:
        buy_true.append(df["预测收盘价"][i])
    for i in sell_true_x:
        sell_true.append(df["预测收盘价"][i])


    plt.title("2022.1.4-2022.1.28 的总资产变化", fontsize=15)
    plt.xlabel("时点", fontsize=15)
    plt.ylabel("总资产", fontsize=15)
    plt.plot([i for i in range(0,33)], P_list,
                color = "blue", linewidth = 2.5)
    plt.legend()
    plt.show()


    # plt.title("2022.1.4-2022.1.28 的最终买入点和卖出点", fontsize=15)
    # plt.xlabel("时点", fontsize=15)
    # plt.ylabel("收盘价", fontsize=15)
    # plt.scatter(buy_true_x, buy_true, s = 80, color = "red", label = "买入点", zorder =
2)
    # plt.scatter(sell_true_x, sell_true, s = 80, color = "green", label = "卖出点", zorder =
3)
    # plt.plot(x, df["实际收盘价"], color = "violet", linewidth = 2.5,
    #             label = "实际收盘价", zorder = 1)
    # plt.plot(x, df["预测收盘价"], color = "blue", linewidth = 2.5,
    #             label = "预测收盘价", zorder = 1)
    # plt.legend()
    # plt.show()
```

计算总收益率等指标：

```
    result_df = pd.DataFrame()
```

```
result_df["date"] = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]
x = []
y = []
z = []
a = []
b = []
c = []

z500 = [-0.000672093,-0.017918592,0.001008684,-0.007508216,0.002198413,
        -0.007818714,0.008648573,-0.01377994,-0.002341387,0.010834201,
        0.000740347,-0.006197908,-0.008706092,-0.011360683,0.000376988,
        -0.033712669,0.005726564,-0.026046209,-0.003842904]

""" 第一天 """
i = 0
P = 1000000
index_buy = buy_true_x[i]
index_sell = sell_true_x[i]
buy_price = df["实际收盘价"][index_buy]
sell_price = df["实际收盘价"][index_sell]
share = P/(1.003*buy_price)
P_c = share*sell_price*(0.997)
xi = P_c / P - 1
x.append(xi)
yi = xi - z500[0]*0.9 #第一天
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (1) # 第一天
a.append(ai)
bi = 0
b.append(bi)
ci = P_c
c.append(ci)

""" 第二天 """
xi = 0
x.append(xi)
yi = xi - z500[1]*0.9 # 第二天
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (2)  # 第二天
a.append(ai)
```

```
    bi = ai/zi
    b.append(bi)
    ci = ci
    c.append(ci)

    """ 第三天 """
    i = 1
    P = ci
    index_buy = buy_true_x[i]
    index_sell = sell_true_x[i]
    buy_price = df["实际收盘价"][index_buy]
    sell_price = df["实际收盘价"][index_sell]
    share = P/(1.003*buy_price)
    P_c = share*sell_price*(0.997)
    xi = P_c / P - 1
    x.append(xi)
    yi = xi - z500[2]*0.9 #第三天
    y.append(yi)
    zi = np.std(y)
    z.append(zi)
    ai = yi / (3) #第三天
    a.append(ai)
    bi = ai/zi
    b.append(bi)
    ci = P_c
    c.append(ci)

    """ 第四天 """
    xi = 0
    x.append(xi)
    yi = xi - z500[3]*0.9 #  第四天
    y.append(yi)
    zi = np.std(y)
    z.append(zi)
    ai = yi / (4)   #  第四天
    a.append(ai)
    bi = ai/zi
    b.append(bi)
    ci = ci
    c.append(ci)

    """ 第五天 """
    i = 2
    P = ci
```

```
index_buy = buy_true_x[i]
index_sell = sell_true_x[i]
buy_price = df["实际收盘价"][index_buy]
sell_price = df["实际收盘价"][index_sell]
share = P/(1.003*buy_price)
P_c = share*sell_price*(0.997)
xi = P_c / P - 1
x.append(xi)
yi = xi - z500[4]*0.9 #第 5 天
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (5) #第 5 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = P_c
c.append(ci)

""" 第六天-288 """
xi = 0
x.append(xi)
yi = xi - z500[5]*0.9 # 第 6 天
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (6)  # 第 6 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = ci
c.append(ci)

""" 第 7 天-336 """
i = 3
P = ci
index_buy = buy_true_x[i]
index_sell = sell_true_x[i]
buy_price = df["实际收盘价"][index_buy]
sell_price = df["实际收盘价"][index_sell]
share = P/(1.003*buy_price)
P_c = share*sell_price*(0.997)
xi = P_c / P - 1
x.append(xi)
```

```
yi = xi - z500[6]*0.9 #第 7 天-1
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (7) #第 7 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = P_c
c.append(ci)

""" 第 8 天-384
这一天只有买入，没有卖出
计算总资产应该用这一天最后的收盘价，也就是 df[383]来计算
"""
i = 4
P = ci
index_buy = buy_true_x[i]
buy_price = df["实际收盘价"][index_buy]
share = P/(1.003*buy_price)
P_c = share*df["实际收盘价"][383] # 用这一天最后的收盘价，且不需要费率
xi = P_c / ci - 1
x.append(xi)
yi = xi - z500[7]*0.9 #第 8 天-1
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (8) #第 8 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = P_c
c.append(ci)

""" 第 9 天-432
这一天有上一天的卖出，
有当日的买入
当日的卖出在次日
"""
# 先进行上一次卖出
index_sell = sell_true_x[i]
sell_price = df["实际收盘价"][index_sell]
P = share*sell_price*(0.997)
```

```
i = 5
index_buy = buy_true_x[i]
index_sell = sell_true_x[i]
buy_price = df["实际收盘价"][index_buy]
sell_price = df["实际收盘价"][index_sell]
share = P/(1.003*buy_price)
P_c = share*df["实际收盘价"][431] # 用这一天最后的收盘价，且不需要费率
xi = P_c / ci - 1
x.append(xi)
yi = xi - z500[8]*0.9 #第 9 天-1
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (9) #第 9 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = P_c
c.append(ci)

""" 第 10 天-480
这一天有上一天的卖出，
有当日的买入
"""
# 先进行上一次卖出
P = share*sell_price*(0.997)

i = 6
index_buy = buy_true_x[i]
index_sell = sell_true_x[i]
buy_price = df["实际收盘价"][index_buy]
sell_price = df["实际收盘价"][index_sell]
share = P/(1.003*buy_price)
P_c = share*df["实际收盘价"][479] # 用这一天最后的收盘价，且不需要费率
xi = P_c / ci - 1
x.append(xi)
yi = xi - z500[9]*0.9 #第 10 天-1
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (10) #第 10 天
a.append(ai)
bi = ai/zi
b.append(bi)
```

```
ci = P_c
c.append(ci)

""" 第 11 天-528
这一天有 495 的卖出
"""
# 先进行上一次卖出
P = ci
P_c = share*sell_price*(0.997)
xi = P_c / P - 1
x.append(xi)
yi = xi - z500[10]*0.9 # 第 11 天-1
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (11)  # 第 11 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = ci
c.append(ci)

""" 第 12 天-576
这一天有 564 的买入
"""
i = 7
P = ci
index_buy = buy_true_x[i]
index_sell = sell_true_x[i]
buy_price = df["实际收盘价"][index_buy]
sell_price = df["实际收盘价"][index_sell]
share = P/(1.003*buy_price)
P_c = share*df["实际收盘价"][575] # 用这一天最后的收盘价，且不需要费率
xi = P_c / P - 1
x.append(xi)
yi = xi - z500[11]*0.9 #第 12 天-1
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (12) #第 12 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = P_c
```

```
c.append(ci)


""" 第 13 天-624
这一天有 583 的卖出
"""
P = ci
P_c = share*sell_price*(0.997)
xi = P_c / P - 1
x.append(xi)
yi = xi - z500[12]*0.9 # 第 13 天-1
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (13)  # 第 13 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = ci
c.append(ci)

""" 第 14 天-672
无操作
"""
xi = 0
x.append(xi)
yi = xi - z500[13]*0.9 # 第 14 天
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (14)  # 第 14 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = ci
c.append(ci)

""" 第 15 天-720
买入卖出
"""
i = 8
P = ci
index_buy = buy_true_x[i]
index_sell = sell_true_x[i]
```

```
buy_price = df["实际收盘价"][index_buy]
sell_price = df["实际收盘价"][index_sell]
share = P/(1.003*buy_price)
P_c = share*sell_price*(0.997)
xi = P_c / P - 1
x.append(xi)
yi = xi - z500[14]*0.9 #第 15 天
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (15) #  第 15 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = P_c
c.append(ci)

""" 第 16 天-768
无
"""
xi = 0
x.append(xi)
yi = xi - z500[15]*0.9 #  第 16 天
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (16)   #  第 16 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = ci
c.append(ci)

""" 第 17 天-816
买入卖出
"""
i = 9
P = ci
index_buy = buy_true_x[i]
index_sell = sell_true_x[i]
buy_price = df["实际收盘价"][index_buy]
sell_price = df["实际收盘价"][index_sell]
share = P/(1.003*buy_price)
P_c = share*sell_price*(0.997)
```

```
xi = P_c / P - 1
x.append(xi)
yi = xi - z500[16]*0.9 #第 17 天
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (17) #  第 17 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = P_c
c.append(ci)

"""  第 18 天-864
无
"""
xi = 0
x.append(xi)
yi = xi - z500[17]*0.9 #  第 18 天
y.append(yi)
zi = np.std(y)
z.append(zi)
ai = yi / (18)   #  第 18 天
a.append(ai)
bi = ai/zi
b.append(bi)
ci = ci
c.append(ci)

"""  第 18 天-912
买入卖出
"""
i = 10
P = ci
index_buy = buy_true_x[i]
index_sell = sell_true_x[i]
buy_price = df["实际收盘价"][index_buy]
sell_price = df["实际收盘价"][index_sell]
share = P/(1.003*buy_price)
P_c = share*sell_price*(0.997)
xi = P_c / P - 1
x.append(xi)
yi = xi - z500[18]*0.9 #第 19 天
y.append(yi)
```

```
    zi = np.std(y)
    z.append(zi)
    ai = yi / (18) #  第 19 天
    a.append(ai)
    bi = ai/zi
    b.append(bi)
    ci = P_c
    c.append(ci)

    result_df["当日收益率"] = x
    result_df["超额收益率"] = y
    result_df["超额收益率标准差"] = z
    result_df["日均超额收益率"] = a
    result_df["信息比率"] = b
    result_df["产品净值"] = c
```

Q4 结果的可视化：

```
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from pylab import *
    from matplotlib.font_manager import FontProperties
    import matplotlib.pyplot as plt
    plt.rcParams['font.sans-serif']=['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    df = pd.read_csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/Q4_result.csv",
                                encoding = "utf-8")

    x = [i for i in range(0, 19)]
    plt.title("2022.1.4-2022.1.28 的指标计算", fontsize=15)
    plt.xlabel("时点", fontsize=15)
    plt.ylabel("指标", fontsize=15)
    # plt.plot(x, df["当日收益率"], color = "black", linewidth = 2.5,label = "当日收益率
")
    # plt.plot(x, df["总收益率"], color = "cornflowerblue", linewidth = 2.5,label = "总收
益率")
    # plt.plot(x, df["超额收益率"], color = "turquoise", linewidth = 2.5,label = "超额收
益率")
    # plt.plot([i for i in range(1, 19)], df["超额收益率标准差"][1:19], color =
"blueviolet", linewidth = 2.5,label = "超额收益率标准差")
    # plt.plot(x, df["日均超额收益率"], color = "brown", linewidth = 2.5,label = "日均
超额收益率")
    # plt.plot([i for i in range(0, 18)], df["最大回撤率"][0:18], color = "gold", linewidth
= 2.5,label = "最大回撤率")
```

```
    plt.plot([i for i in range(1, 19)], df["信息比率"][1:19], color = "fuchsia", linewidth = 2.5,label = "信息比率")
    plt.legend()
    plt.show()
```

建立监督学习数据集以及归一化：

```
    from pandas import DataFrame
    from pandas import concat
    import pandas as pd
    import numpy as np
    from sklearn.preprocessing import MinMaxScaler
    import matplotlib.pyplot as plt

    def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):

        n_vars = 1 if type(data) is list else data.shape[1]
        df = DataFrame(data)
        cols, names = list(), list()
        for i in range(n_in, 0, -1):
            cols.append(df.shift(i))
            names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
        for i in range(0, n_out):
            cols.append(df.shift(-i))
            if i == 0:
                names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
            else:
                names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
        agg = concat(cols, axis=1)
        agg.columns = names
        if dropnan:
            agg.dropna(inplace=True)
        return agg


    df = pd.read_csv('5min.csv')
    Q2df = df.drop(["Open","Close","High", "Low"], axis = 1)
    Q3df = df.drop(["Volume","Turnover"], axis = 1)


    """ ====== 归一化数据 ====== """
    scaler = MinMaxScaler(feature_range = (0, 1))
    Q2df_scaled = scaler.fit_transform(Q2df)
    volume = []
    turnover = []
```

```
for i in Q2df_scaled:
    volume.append(i[0])
    turnover.append(i[1])
Q2df_scaleddf = pd.DataFrame()
Q2df_scaleddf["Volume"] = volume
Q2df_scaleddf["Turnover"] = turnover
Q2 = series_to_supervised(Q2df, n_in = 48)
#Q2.to_csv('Q2_xgboost.csv')
Q2_s = series_to_supervised(Q2df_scaleddf, n_in = 48)
#Q2_s.to_csv('Q2_xgboost_s.csv')

Q2_res = pd.read_csv('Q2_xgboost_s_pre.csv')
Q2_result = pd.DataFrame()
Q2_result["Volume"] = Q2_res["var1.t."]
Q2_result["Turnover"] = Q2_res["var2.t."]
Q2_result_np = np.array(Q2_result)
pred = scaler.inverse_transform(Q2_result_np)[:,0][-912:]
ori = Q2df["Volume"][-912:]

plt.plot([i for i in range(0, 912)], ori, color = "red", label = "Real Volume")
plt.plot([i for i in range(0, 912)], pred, color = "blue", label = "Predicted Volume")
plt.title("Volume Prediction")
plt.xlabel("time")
plt.ylabel("Volume")
plt.legend()
plt.show()

""" ====== 绘制验证损失 ====== """
plt.clf()
msedf = pd.read_csv('XGBoost_MSE.csv')
msedf["Q2"]

plt.plot([i for i in range(0, 200)], msedf["Q2"])
plt.title("XGBoost Volume's Training Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()



""" ====== 归一化数据 ====== """
scaler = MinMaxScaler(feature_range = (0, 1))
Q3df_scaled = scaler.fit_transform(Q3df)
```

```
Open = []
Close = []
High = []
Low = []
for i in Q3df_scaled:
    Open.append(i[0])
    Close.append(i[1])
    High.append(i[2])
    Low.append(i[3])
Q3df_scaleddf = pd.DataFrame()
Q3df_scaleddf["Open"] = Open
Q3df_scaleddf["Close"] = Close
Q3df_scaleddf["High"] = High
Q3df_scaleddf["Low"] = Low
Q3 = series_to_supervised(Q3df, n_in = 48)
#Q3.to_csv('Q3_xgboost.csv')
Q3_s = series_to_supervised(Q3df_scaleddf, n_in = 48)
#Q3_s.to_csv('Q3_xgboost_s.csv')

Q3_res = pd.read_csv('Q3_xgboost_s_pre.csv')
Q3_result = pd.DataFrame()
Q3_result["var1.t."] = Q3_res["var1.t."]
Q3_result["var2.t."] = Q3_res["var2.t."]
Q3_result["var3.t."] = Q3_res["var3.t."]
Q3_result["var4.t."] = Q3_res["var4.t."]

Q3_result_np = np.array(Q3_result)
pred = scaler.inverse_transform(Q3_result_np)[:,1][-912:]
ori = Q3df["Close"][-912:]

plt.plot([i for i in range(0, 912)], ori, color = "red", label = "Real Close")
plt.plot([i for i in range(0, 912)], pred, color = "blue", label = "Predicted Close")
plt.title("Close Prediction")
plt.xlabel("time")
plt.ylabel("Close")
plt.legend()
plt.show()

""" ====== 绘制验证损失 ====== """
plt.clf()
msedf = pd.read_csv('XGBoost_MSE.csv')
msedf["Q3"]

plt.plot([i for i in range(0, 200)], msedf["Q2"])
```

```
plt.title("XGBoost Close's Training Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```

附录 3：（R）

斯皮尔曼相关系数的计算及可视化绘图：

```
mydata = read.csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/数字经济板块信息.csv",
                  header = TRUE, sep = ",", encoding="UTF-8")
data = mydata[,4:9]
data = data[1:6240,]
names(data) = c("开盘价","收盘价","最高价","最低价","成交量","成交额")
library(corrgram)
cordata = cor(data, method = "spearman")
corrgram(cordata,
         type = "corr",
         order=TRUE,
         lower.panel=panel.shade,
         upper.panel=panel.pie,
         text.panel=panel.txt,
         main="数字经济板块（五分钟）的数据的 spearman 相关系数")


mydata = read.csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/数字经济板块信息（日）.csv",
                  header = TRUE, sep = ",", encoding="UTF-8")
data = mydata[,4:9]
names(data) = c("开盘价","收盘价","最高价","最低价","成交量","成交额")
cordata = cor(data, method = "spearman")
corrgram(cordata,
         type = "corr",
         order=TRUE,
         lower.panel=panel.shade,
         upper.panel=panel.pie,
         text.panel=panel.txt,
         main="数字经济板块（每日）的数据的 spearman 相关系数")


mydata = read.csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/1day.csv",
                  header = TRUE, sep = ",", encoding="UTF-8")
```

```
data = mydata[,4:53]
library(corrgram)
cordata = cor(data, method = "spearman")
corrgram(cordata,
         type = "corr",
         order=TRUE,
         lower.panel=panel.shade,
         upper.panel=panel.pie,
         text.panel=panel.txt,
         main="以天为时间频率的数据的 spearman 相关系数")

write.table(cordata,"C:/Users/Lenovo/Desktop/2022 华中杯/B/以天为时间频率的数据的斯皮尔曼相关系数.csv",row.names=T,col.names=T,sep=",")
```

以 XGBoost 做特征重要性检测：

```
#五分钟
library(xgboost)
library(caret)
library(Matrix)
data = read.csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/5min.csv",
                header = TRUE, sep = ",", encoding="UTF-8")

#以 Turnover 为例，展示 xgboost 的拟合度
trainset = data[1:5616, ]
testset = data[5617:6240, ]

#标准化
mean_Open = mean(trainset$Open)
std_Open = sd(trainset$Open)
mean_Close = mean(trainset$Close)
std_Close = sd(trainset$Close)
mean_High = mean(trainset$High)
std_High = sd(trainset$High)
mean_Low = mean(trainset$Low)
std_Low = sd(trainset$Low)
mean_Volume = mean(trainset$Volume)
std_Volume = sd(trainset$Volume)
mean_Turnover = mean(trainset$Turnover)
std_Turnover = sd(trainset$Turnover)

dataa = data
dataa$Open = (dataa$Open - mean_Open) / std_Open
dataa$Close = (dataa$Close - mean_Close) / std_Close
dataa$High = (dataa$High - mean_High) / std_High
```

```
dataa$Low = (dataa$Low - mean_Low) / std_Low
dataa$Volume = (dataa$Volume - mean_Volume) / std_Volume
dataa$Turnover = (dataa$Turnover - mean_Turnover) / std_Turnover

trainset = dataa[1:5616, ]
testset = dataa[5617:6240, ]



traindata1 <- data.matrix(trainset[,c(1,2,3,5,6)])
traindata2 <- Matrix(traindata1,sparse=T)
train_y <- as.numeric(trainset[,4])
traindata <- list(data=traindata2,label=train_y)
dtrain <- xgb.DMatrix(data = traindata$data, label = traindata$label)

testset1 <- data.matrix(testset[,c(1,2,3,5,6)])
testset2 <- Matrix(testset1,sparse=T)
test_y <- as.numeric(testset[,4])
testset <- list(data=testset2,label=test_y)
dtest <- xgb.DMatrix(data = testset$data, label = testset$label)

model_xgb <- xgboost(data = dtrain,
                     booster='gbtree',
                     max_depth=100,
                     eta=0.1,
                     objective='reg:squarederror',
                     nround=300)
pre = predict(model_xgb, newdata = dtest)

library(ggplot2)
xx = 1:624
ggplot()+
  geom_line(aes(x = xx, y = pre), color = "red", size = 1, show.legend = "预测值")
+
  geom_line(aes(x = xx, y = dataa[5617:6240, 6]), color = "blue", size = 1,
show.legend = "真实值") +
  labs(title = "利用 XGBoost 以开盘价、收盘价、最高价、最低价、成交量对后
10%成交额进行预测",
       x = "时点",y = "成交额")

importance = xgb.importance(model = model_xgb)
xgb.ggplot.importance(importance)



#一天
```

```
library(xgboost)
library(caret)
library(Matrix)
library(reshape)
data = read.csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/1day_std.csv",
                header = TRUE, sep = ",", encoding="UTF-8")
data = rename(data, c(X.U.FEFF.开盘价  = '开盘价'))

data_Open = data[,c(1,2,3,4,9,10,11,14,15,16,17,18,19,20)]
data_Close = data[,c(2,1,3,4,9,10,11,14,15,16,17,18,19,20)]
data_High = data[,c(3,1,2,4,9,10,11,14,15,16,17,18,19,20)]
data_Low = data[,c(4,1,2,3,9,10,11,14,15,16,17,18,19,20)]
data_Volume = data[,c(5,6,12,13,22,23,24,25,26)]
data_Turnover = data[,c(6,5,7,8,12,13,21,22,23,24)]

#分别拟合
data = data_Open
trainset = data[1:117, ]
testset = data[118:130, ]

traindata1 <- data.matrix(trainset[,c(2:14)])
traindata2 <- Matrix(traindata1,sparse=T)
train_y <- as.numeric(trainset[,1])
traindata <- list(data=traindata2,label=train_y)
dtrain <- xgb.DMatrix(data = traindata$data, label = traindata$label)

testset1 <- data.matrix(testset[,c(2:14)])
testset2 <- Matrix(testset1,sparse=T)
test_y <- as.numeric(testset[,1])
testset <- list(data=testset2,label=test_y)
dtest <- xgb.DMatrix(data = testset$data, label = testset$label)

model_xgb <- xgboost(data = dtrain,
                     booster='gbtree',
                     max_depth=100,
                     eta=0.1,
                     objective='reg:squarederror',
                     nround=200)
pre = predict(model_xgb, newdata = dtest)

library(ggplot2)
xx = 1:13
ggplot()+
  geom_line(aes(x = xx, y = pre), color = "red", size = 1, show.legend = "预测值")
```

```
+
        geom_line(aes(x = xx, y = data[118:130, 1]), color = "blue", size = 1, show.legend
= "真实值") +
        labs(title = "时间间隔为日的数据对后 10%开盘价进行预测",
             x = "时点",y = "开盘价")

    importance = xgb.importance(model = model_xgb)
    xgb.ggplot.importance(importance)

    data = data_Close
    trainset = data[1:117, ]
    testset = data[118:130, ]
    traindata1 <- data.matrix(trainset[,c(2:14)])
    traindata2 <- Matrix(traindata1,sparse=T)
    train_y <- as.numeric(trainset[,1])
    traindata <- list(data=traindata2,label=train_y)
    dtrain <- xgb.DMatrix(data = traindata$data, label = traindata$label)
    testset1 <- data.matrix(testset[,c(2:14)])
    testset2 <- Matrix(testset1,sparse=T)
    test_y <- as.numeric(testset[,1])
    testset <- list(data=testset2,label=test_y)
    dtest <- xgb.DMatrix(data = testset$data, label = testset$label)
    model_xgb <- xgboost(data = dtrain,
                         booster='gbtree',
                         max_depth=100,
                         eta=0.1,
                         objective='reg:squarederror',
                         nround=200)
    importance = xgb.importance(model = model_xgb)
    xgb.ggplot.importance(importance)

    data = data_High
    trainset = data[1:117, ]
    testset = data[118:130, ]
    traindata1 <- data.matrix(trainset[,c(2:14)])
    traindata2 <- Matrix(traindata1,sparse=T)
    train_y <- as.numeric(trainset[,1])
    traindata <- list(data=traindata2,label=train_y)
    dtrain <- xgb.DMatrix(data = traindata$data, label = traindata$label)
    testset1 <- data.matrix(testset[,c(2:14)])
    testset2 <- Matrix(testset1,sparse=T)
    test_y <- as.numeric(testset[,1])
    testset <- list(data=testset2,label=test_y)
    dtest <- xgb.DMatrix(data = testset$data, label = testset$label)
```

```
model_xgb <- xgboost(data = dtrain,
                     booster='gbtree',
                     max_depth=100,
                     eta=0.1,
                     objective='reg:squarederror',
                     nround=200)
importance = xgb.importance(model = model_xgb)
xgb.ggplot.importance(importance)

data = data_Low
trainset = data[1:117, ]
testset = data[118:130, ]
traindata1 <- data.matrix(trainset[,c(2:14)])
traindata2 <- Matrix(traindata1,sparse=T)
train_y <- as.numeric(trainset[,1])
traindata <- list(data=traindata2,label=train_y)
dtrain <- xgb.DMatrix(data = traindata$data, label = traindata$label)
testset1 <- data.matrix(testset[,c(2:14)])
testset2 <- Matrix(testset1,sparse=T)
test_y <- as.numeric(testset[,1])
testset <- list(data=testset2,label=test_y)
dtest <- xgb.DMatrix(data = testset$data, label = testset$label)
model_xgb <- xgboost(data = dtrain,
                     booster='gbtree',
                     max_depth=100,
                     eta=0.1,
                     objective='reg:squarederror',
                     nround=200)
importance = xgb.importance(model = model_xgb)
xgb.ggplot.importance(importance)

data = data_Volume
trainset = data[1:117, ]
testset = data[118:130, ]
traindata1 <- data.matrix(trainset[,c(2:9)])
traindata2 <- Matrix(traindata1,sparse=T)
train_y <- as.numeric(trainset[,1])
traindata <- list(data=traindata2,label=train_y)
dtrain <- xgb.DMatrix(data = traindata$data, label = traindata$label)
testset1 <- data.matrix(testset[,c(2:9)])
testset2 <- Matrix(testset1,sparse=T)
test_y <- as.numeric(testset[,1])
testset <- list(data=testset2,label=test_y)
dtest <- xgb.DMatrix(data = testset$data, label = testset$label)
```

```
model_xgb <- xgboost(data = dtrain,
                     booster='gbtree',
                     max_depth=100,
                     eta=0.1,
                     objective='reg:squarederror',
                     nround=200)
importance = xgb.importance(model = model_xgb)
xgb.ggplot.importance(importance)

data = data_Turnover
trainset = data[1:117, ]
testset = data[118:130, ]
traindata1 <- data.matrix(trainset[,c(2:10)])
traindata2 <- Matrix(traindata1,sparse=T)
train_y <- as.numeric(trainset[,1])
traindata <- list(data=traindata2,label=train_y)
dtrain <- xgb.DMatrix(data = traindata$data, label = traindata$label)
testset1 <- data.matrix(testset[,c(2:10)])
testset2 <- Matrix(testset1,sparse=T)
test_y <- as.numeric(testset[,1])
testset <- list(data=testset2,label=test_y)
dtest <- xgb.DMatrix(data = testset$data, label = testset$label)
model_xgb <- xgboost(data = dtrain,
                     booster='gbtree',
                     max_depth=100,
                     eta=0.1,
                     objective='reg:squarederror',
                     nround=200)
importance = xgb.importance(model = model_xgb)
xgb.ggplot.importance(importance)
```

XGBoost 做时间序列分析：

```
library(xgboost)
library(caret)
library(Matrix)
data = read.csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/Q2_xgboost_s.csv",
                header = TRUE, sep = ",", encoding="UTF-8")

trainset = data[1:5280, ]
testset = data[5281:6192, ]

x = 1:96
x = append(x,98)
traindata1 <- data.matrix(trainset[,x])
```

```
traindata2 <- Matrix(traindata1,sparse=T)
train_y <- as.numeric(trainset[,97])
traindata <- list(data=traindata2,label=train_y)
dtrain <- xgb.DMatrix(data = traindata$data, label = traindata$label)

testset1 <- data.matrix(testset[,c(x)])
testset2 <- Matrix(testset1,sparse=T)
test_y <- as.numeric(testset[,97])
testset <- list(data=testset2,label=test_y)
dtest <- xgb.DMatrix(data = testset$data, label = testset$label)

model_xgb <- xgboost(data = dtrain,
                     booster='gbtree',
                     max_depth=100,
                     eta=0.1,
                     objective='reg:squarederror',
                     nround=200)
testpre = predict(model_xgb, newdata = dtest)
trainpre = predict(model_xgb, newdata = dtrain)

library(ggplot2)
xx = 1:6192
ggplot()+
  geom_line(aes(x = xx, y = c(trainpre, testpre)), color = "blue", size = 1,
show.legend = "预测值") +
  geom_line(aes(x = xx, y = data[, 97]), color = "red", size = 1, show.legend = "真实
值") +
  labs(title = "XGBoost's Volume Prediction",
       x = "Time",y = "Volume")

result = c(trainpre, testpre)
res = data.frame(data, result)
write.csv(res, "C:/Users/Lenovo/Desktop/2022 华中杯/B/Q2_xgboost_s_pre.csv")









data = read.csv("C:/Users/Lenovo/Desktop/2022 华中杯/B/Q3_xgboost_s.csv",
                header = TRUE, sep = ",", encoding="UTF-8")

trainset = data[1:5280, ]
testset = data[5281:6192, ]
```

```
x = 1:193
x = append(x,195)
x = append(x,196)
traindata1 <- data.matrix(trainset[,x])
traindata2 <- Matrix(traindata1,sparse=T)
train_y <- as.numeric(trainset[,194])
traindata <- list(data=traindata2,label=train_y)
dtrain <- xgb.DMatrix(data = traindata$data, label = traindata$label)

testset1 <- data.matrix(testset[,x])
testset2 <- Matrix(testset1,sparse=T)
test_y <- as.numeric(testset[,194])
testset <- list(data=testset2,label=test_y)
dtest <- xgb.DMatrix(data = testset$data, label = testset$label)

model_xgb <- xgboost(data = dtrain,
                     booster='gbtree',
                     max_depth=100,
                     eta=0.1,
                     objective='reg:squarederror',
                     nround=200)
testpre = predict(model_xgb, newdata = dtest)
trainpre = predict(model_xgb, newdata = dtrain)

library(ggplot2)
xx = 1:6192
ggplot()+
  geom_line(aes(x = xx, y = c(trainpre, testpre)), color = "blue", size = 1, show.legend = "预测值") +
  geom_line(aes(x = xx, y = data[, 97]), color = "red", size = 1, show.legend = "真实值") +
  labs(title = "XGBoost's Volume Prediction",
       x = "Time",y = "Volume")

result = c(trainpre, testpre)
res = data.frame(data, result)
write.csv(res, "C:/Users/Lenovo/Desktop/2022 华中杯/B/Q3_xgboost_s_pre.csv")
```