

实习总结报告

魏靖轩

目 录

CH1 选择市场.....	4
CH2 市场中.....	4
CH2.1 纪律性	4
CH2.2 基于技术指标的交易策略.....	5
CH2.2.1 技术指标.....	5
CH2.2.2 交易策略 1(基于布林通道的马丁格尔策略).....	5
(1) 策略思路	5
(2) 策略回测	6
(3) 模型局限性.....	7
(4) 风险控制与优化.....	8
(I) 限制加倍数量	8
(II) 限制卖出条件	8
(III) 与 ADX 指标结合	9
CH2.2.3 交易策略 2(双均线交易策略).....	10
(1) 策略思路	10
(2) 策略回测	11
CH2.2.4 双策略结合	12
(1) 趋势判断	12
(I) EMA 趋势判断.....	12
(II) 卡尔曼滤波器趋势判断.....	12
(2) 不同趋势的交易策略.....	15
(I) 多头交易策略	15
(II) 空头交易策略	15
(3) 策略回测	16

CH2.3 基于机器学习的交易策略.....	17
CH2.3.1 多维技术指标与 KNN 算法结合的交易策略.....	17
(1) 技术指标选取	17
(2) 特征工程	17
(3) 曼哈顿距离计算.....	18
(4) KNN 算法.....	18
(5) 策略回测	18
CH2.3.2 基于多维指标特征工程的异构集成学习预测模型.....	19
(1) 特征构建	19
(2) 特征选择	20
(3) 模型训练	21
(4) 元学习器	23
(5) 阈值优化	24
(6) 模型结果	24
CH2.4 参数动态调整思路	25
CH2.4.1 遍历优化思路.....	25
CH2.4.2 基于聚类算法的因子优化.....	26
(1) 研究思路	26
(2) 研究结果	27
(3) 进一步泛化.....	27
CH3 敬畏市场.....	28
写在最后.....	28

CH1 选择市场

在资本市场中，选择市场就是选择投资标的，不同的市场，不同的投资标的直接存在的差异是巨大的，所有的技术分析，指标，风险控制是针对单一市场的，对待不同的市场，不同的投资标的，可以保持一个交易思维，但是一定要制定不同的交易策略。

投资标的和市场的选择是个性化的，和每个人的技术能力，决策能力，抗风险能力，心理素质息息相关，有部分投资者可以在市场下跌 30% 的时候持续加仓，有部分投资者在亏损 5% 的时候就会及时止损，这就是差异所在，所以重要的不是选择最火热的，参与者最多的市场，而是根据自己的能力，心理选择最适合的交易市场。

但有一条规则适用于任何投资市场，风险越大收益越大，很多人羡慕 Crypto 市场的高收益和高年化，但很少有人经历过类似 2024 年 8 月，2025 年 1 月一天 30% 的回撤，如果真正经历过一天 30% 以上的回撤，不仅是对交易策略的考验，更多的是对人性的考验。相反，对于风险最低的投资标的，国债，极低的风险也对应不高的收益，如何想在类似这种市场实现逆天改命，是不现实的想法。

同样需要注意的是，选择不同的市场也是在选择不同的对手盘，对手盘的能力也极大影响盈利的可能性。例如，中国的市场参与者大部分集中在沪深两市，部分会投身期货市场，还有部分会参与港美股，显然，这些市场中，沪深两市拥有最多的“散户”，而相较于此，美股，期货市场有更多的“机构”，这些机构作为成熟的投资者会极大的加大在这个市场的盈利难度。

最后，最重要的一点，要选择足够了解的市场，或者加深对市场的了解。例如，大部分中国投资者绕不开的就是沪深两市，而对其的认知大概分成两派，一派是认为是骗局，他们绝大部分在市场中亏损巨大，从而认为中国股市是扶不起的阿斗；另一派则是坚定看涨派，他们普遍认为沪深市场会像美股，只要在市场的时间足够久，就能享有还不错的收益，可事实上，沪深两市具有其特殊性，既不是骗局，也不是摇钱树，它更多是一个非市场化市场，在一个巨大的经济体中扮演者一个重要的工具角色。

CH2 在市场

CH2.1 纪律性

在一切策略执行之前必须强调的东西是纪律性。所有的策略必须有的就是买入规则和卖出规则，经过了研究和回测，这些买入和卖出的点位或者指标是具有可行性的，但是对大部分人来说，交易是人为的动作，而人为的动作就会掺杂主观因素。

比如在对一个回撤中的资产设置了加仓点位 150，140，130，120，计划将现金仓

位分四次买入，当价格在跌破 150 后成功买入第一笔，但在下探到 145 的时候拐头向上，出现了可以是反转的信号，根据之前制定的规则，这显然没到加仓的点位，但对大部分市场参与者来说，他们认为出现了反转信号，便会把这视为所谓的“最后机会”，于是没有恪守客观的“加仓点位”，在 145 提前加仓，人性的贪婪让他们想获得更大的利润，但结局往往是悲观的，当价格如期下杀到 120 的时候，提前加仓，破坏纪律的选择让这部分市场参与者没有资金在最合适的点位抄底。

这也很大程度上是程序化交易比手动交易更具优势的原因，对程序化交易者来说，他们无需在一天大部分的时间盯着盘口，也无需有过多的人工干预，机器会严格按照指定的规则去做出交易选择。

CH2.2 基于技术指标的交易策略

CH2.2.1 技术指标

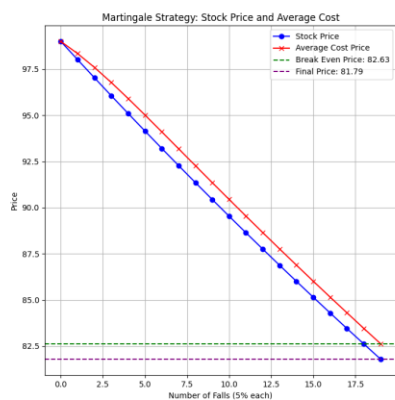
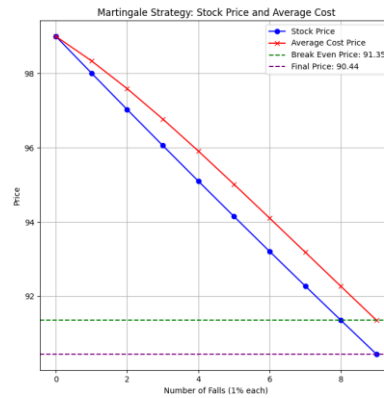
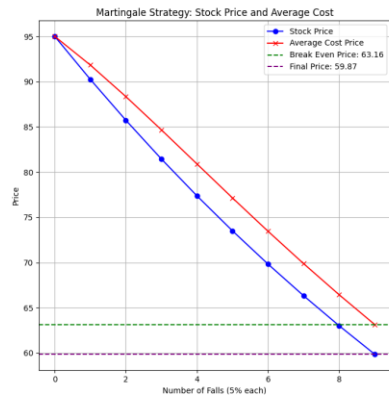
技术指标是市场分析中用于辅助交易决策的数学计算工具，主要用于分析价格走势、趋势强度、超买超卖情况以及市场动能等。在对市场的长期研究中，诞生了无数种技术指标，有的距今已有几十年的历史，不同的市场参与者对技术指标有一定的偏好，但不同的技术指标直接不存在绝对的好与绝对的差，在真正的市场中，也不存在一个万能公式，因此对每个技术指标，在应用之前都应该详细的了解其背后的原理，根据其原理在市场中进行不同的调整。

这里以 RSI 为例，RSI 是应用广泛的一种技术指标，在一般情况下，RSI 的周期是 14 天，也许某些市场来说，以 14 为 RSI 周期具有有效性，但是这不适用于全部的市场，在真正的使用中，多次尝试，多次调整，选择最适合的参数是技术指标交易必不可少的部分。

CH2.2.2 交易策略 1(基于布林通道的马丁格尔策略)

(1) 策略思路

马丁格尔策略是一种赌注策略，也被应用于金融交易中。基本原则是：每次亏损后加倍下注（或加倍交易量），直到最终获胜。这样，如果最终交易成功，累积的利润将足够弥补之前所有的亏损，并获得与初始交易量相等的利润。这里使用类似的思路进行研究。



在理想条件下我们定义：每下跌 5%加仓两倍，使用 python 模拟操作，假设该操作重复 10 次，则此时股票价格在 59.87，而通过该策略的不断加仓，把买入价拉低到 63.16，即即使经过十次 5% 的下跌，总跌幅达到 40% 以上，只要上涨 5.49% 即可达到盈亏平衡。（如图 1）

每下跌 1% 加仓两倍，使用 python 模拟操作，假设该操作重复 10 次，则此时股票价格在 90.44，而通过该策略的不断加仓，把买入价拉低到 91.35，即即使经过十次 1% 的下跌，总跌幅达到 9% 以上，只要上涨 1% 即可达到盈亏平衡（如图 2），在操作 20 次时，这一比例同样约为 1%（如图 3），在操作 50 次时，这一比例也大约为 1%（如图 4）。

将马丁格尔策略结合布林通道构建策略：

1. 当价格触及或低于布林带的下轨时，视为超卖，发出买入信号。
2. 如果之前的买入操作亏损，下一次接触布林带下轨时，交易量会加倍。
3. 当价格触及或高于布林带的上轨时，视为超买，发出卖出信号，平仓。

(2) 策略回测

在原模型中，未设定加倍次数的上限，将每次加倍买入倍数设置为 2，标的为白银连续期货 8 月 14 日到 1 月 14 日 5 分钟 K 线，此过程为单边上涨行情+震荡行情，

在这一特定行情下，回测结果如下：

总体概要			
统计指标	所有交易	多头交易	空头交易
净利润	519,084.72	519,084.72	0.00
总盈利(毛利)	1,312,463.00	1,312,463.00	0.00
总亏损(毛损)	-793,378.63	-793,378.63	0.00
盈利因子(毛利/毛损)	1.65	1.65	0.00
收益率	51.91%	51.91%	0.00%
年化收益率	169.56%	169.56%	0.00%
最大资产回撤	-218,290.00	N/A	N/A
净利润/最大资产回撤	2.38	N/A	N/A
最大资产回撤幅度	-20.26%	N/A	N/A
年化收益率/最大资产回撤幅度	8.37	N/A	N/A
交易次数	132	132	0
盈利交易次数	112	112	0
亏损交易次数	20	20	0
胜率	84.85%	84.85%	0.00%
平均盈亏	3,932.46	3,932.46	0.00
平均盈利	11,718.42	11,718.42	0.00
平均亏损	-39,668.93	-39,668.93	0.00
盈亏比	0.30	0.30	0.00

可以发现该策略对单边上涨和震荡的行情具有很强的有效性，在这个短期收益高达 51.91%。

(3) 模型局限性

在原模型中，未设定加倍次数的上限，将每次加倍买入倍数设置为 2，标的为螺纹钢连续期货 8 月 14 日到 1 月 14 日 5 分钟 K 线，这是一个单边下跌的行情，回测结果如下：

总体概要			
统计指标	所有交易	多头交易	空头交易
净利润	-80,276.65	-80,276.65	0.00
总盈利(毛利)	407,476.94	407,476.94	0.00
总亏损(毛损)	-487,753.56	-487,753.56	0.00
盈利因子(毛利/毛损)	0.84	0.84	0.00
收益率	-8.03%	-8.03%	0.00%
年化收益率	-17.79%	-17.79%	0.00%
最大资产回撤	-247,129.81	N/A	N/A
净利润/最大资产回撤	-0.32	N/A	N/A
最大资产回撤幅度	-21.75%	N/A	N/A
年化收益率/最大资产回撤幅度	-0.82	N/A	N/A
交易次数	85	85	0
盈利交易次数	75	75	0
亏损交易次数	10	10	0
胜率	88.24%	88.24%	0.00%
平均盈亏	-944.43	-944.43	0.00
平均盈利	5,433.03	5,433.03	0.00
平均亏损	-48,775.36	-48,775.36	0.00
盈亏比	0.11	0.11	0.00

根据结果可以发现，在单边下跌行情中，该模型不具有有效性，会造成亏损，同时，在单边强烈下跌插针行情中，该策略会存在系统性风险，因此，在这里选用一些思路规避单边下跌行情中的风险。

(4) 风险控制与优化

(I) 限制加倍数量

为防止无限加仓，控制资金利用率，可以选用限制加仓数量来控制风险，通过限制最大加倍次数，防止在单边大级别下跌行情中出现大量亏损，通过这一操作，保证胜率的同时，实现正的总收益。

(II) 限制卖出条件

为了减少单边大行情中的亏损，将原先策略中“突破布林带上轨卖出”进行修改，加入限制条件，到达布林带上轨判断盈利：

```
# 布林带上轨-卖出信号
elif close[-1] >= upper[-1]:
    if portfolio.buy_quantity > 0:
        # 检查当前的收益
        if portfolio.pnl > 0:
            # 正收益卖出
            sell_close(context.sl, "market", volume=portfolio.buy_quantity, serial_id = 2)
            context.current_volume = context.initial_volume
        else:
            # 负收益不卖
            context.last_touch_lower = False
```

若收益为正卖出，收益为负，持续持有并检查，直到收益为正卖出：

```
# 直到止收益卖出
if portfolio.buy_quantity > 0 and portfolio.pnl < 0:
    if portfolio.pnl >= 0:
        sell_close(context.s1, "market", volume=portfolio.buy_quantity, serial_id = 3)
        context.last_touch_lower = False
        context.current_volume = context.initial_volume
```

对螺纹钢 5 分钟级别 k 线回测，较于之前模型有明显提升：

统计指标	所有交易	多头交易	空头交易
净利润	276,810.78	276,810.78	0.00
总盈利(毛利)	276,950.88	276,950.88	0.00
总亏损(毛损)	-140.08	-140.08	0.00
盈利因子(毛利/毛损)	1977.15	1977.15	0.00
收益率	27.68%	27.68%	0.00%
年化收益率	74.08%	74.08%	0.00%
最大资产回撤	-276,707.75	N/A	N/A
净利润/最大资产回撤	1.00	N/A	N/A
最大资产回撤幅度	-26.49%	N/A	N/A
年化收益率/最大资产回撤幅度	2.80	N/A	N/A
交易次数	65	65	0
盈利交易次数	64	64	0
亏损交易次数	1	1	0
胜率	98.46%	98.46%	0.00%

(III) 与 ADX 指标结合

加入趋势，通过 ADX 判断，在震荡区间可以使用该策略：

```
def handle_bar(context):
    high = history_bars(context.s1, context.bb_period + 30, 'self', 'high', True)
    low = history_bars(context.s1, context.bb_period + 30, 'self', 'low', True)
    close = history_bars(context.s1, context.bb_period + 30, 'self', 'close', True)
    adx = ta.ADX(high, low, close, timeperiod=context.bb_period)
    portfolio = get_portfolio(context.s1, 0)
    # ADX小于等于25时执行交易
    if adx[-1] <= 25:
        # 买入
```

总体概要			
统计指标	所有交易	多头交易	空头交易
净利润	197,155.78	197,155.78	0.00
总盈利(毛利)	402,174.91	402,174.91	0.00
总亏损(毛损)	-205,019.06	-205,019.06	0.00
盈利因子(毛利/毛损)	1.96	1.96	0.00
收益率	19.72%	19.72%	0.00%
年化收益率	50.42%	50.42%	0.00%
最大资产回撤	-199,571.19	N/A	N/A
净利润/最大资产回撤	0.99	N/A	N/A
最大资产回撤幅度	-19.03%	N/A	N/A
年化收益率/最大资产回撤幅度	2.65	N/A	N/A
交易次数	58	58	0
盈利交易次数	47	47	0
亏损交易次数	11	11	0
胜率	81.03%	81.03%	0.00%

继续修改，保证在 ADX 大于等于 25 的时候出局，进一步防止被套风险：

统计指标	所有交易	多头交易	空头交易
净利润	287,394.59	287,394.59	0.00
总盈利(毛利)	497,265.84	497,265.84	0.00
总亏损(毛损)	-209,871.36	-209,871.36	0.00
盈利因子(毛利/毛损)	2.37	2.37	0.00
收益率	28.74%	28.74%	0.00%
年化收益率	77.37%	77.37%	0.00%
最大资产回撤	-89,447.38	N/A	N/A
净利润/最大资产回撤	3.21	N/A	N/A
最大资产回撤幅度	-8.91%	N/A	N/A

由回测结果发现，加入 ADX 趋势指标后，在 ADX 值小于等于 25，即市场位于震荡区间，多空双方均没有明显动向的时候，该策略实现高收益率的同时，实现了低回撤。

CH2.2.3 交易策略 2(双均线交易策略)

(1) 策略思路

MA 均线：简单移动平均是通过计算一组数据点的算术平均值来平滑数据序列。

$$MA_t = \frac{\sum_{i=t-N+1}^t P_i}{N}$$

即如果移动均线上升，本质上是指当前的价格比选定周期 N 个 k 线之前的价格更高，相反，如果移动均线下降，本质上是指当前的价格比选定周期 N 个 k 线之前的价格更低，这种方法对趋势判断存在滞后性。

EMA 均线：指数加权移动平均是对最近的数据点赋予更高权重的一种方法，它使得计算值对近期的数据变化更加敏感。

$$EMA_t = \alpha P_t + (1 - \alpha)EMA_{t-1}$$

$$\alpha = \frac{2}{N + 1}$$

即 EMA 均线通过对最近数据点的更高赋权，可以优化一些滞后性。

所以可以认为，价格变化为趋势可能出现的最早信号，而价格的改变最先带动 EMA 的变化，EMA 可以视为第二信号，在 EMA 出现信号后，才会引起 MA 信号，因此选取 EMA 与 MA 交叉这个时间结点，视为趋势确定或结束的信号。

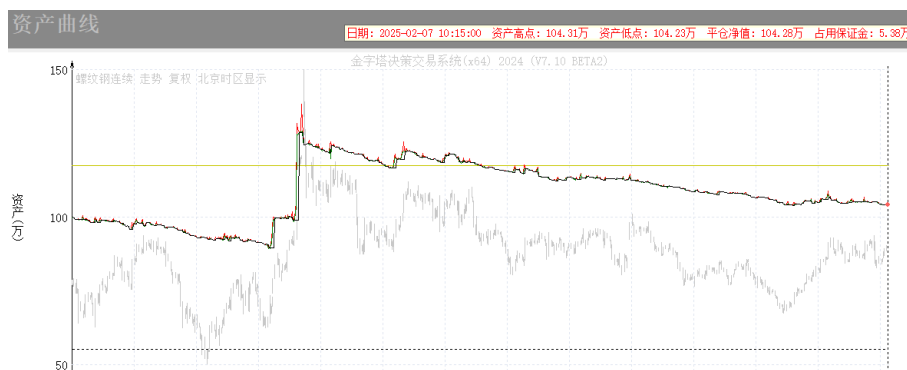


选取螺纹钢 5 分钟 k 线图，EMA&MA 参数设置为 21，可以发现通过双均线交叉判断趋势几乎可以做到和价格同步，优化了一些滞后性的影响。

(2) 策略回测

首先基于策略设计多头部分，定义 EMA 均线从下到上穿过 MA 均线，视为多头行情启动，而 EMA 均线再次回到 MA 均线下方，视为行情结束。

选取螺纹钢主力连续期货 k 线回测，将双均线参数 t 调整，当 t=5 时，在追逐快速上涨行情中有明显优势，在突然上涨的行情中实现了非常好的收益。



接下来设计策略的空头部分，定义 EMA 均线从上到下穿过 MA 均线视为空头行情启动，EMA 均线回到 MA 均线上方视为空头行情结束，同时选取 8 月 14 日至 2 月 7 日螺纹钢主力连续合约 5 分钟 k 线图，在这里将 EMA 与 MA 的 t 值设置为相同时结果均不理想，因此将对时效性反映更敏感的 EMA 的 t 值设定为 5，MA 设定为 21，得到了较好的回测结果。



CH2.2.4 双策略结合

(1) 趋势判断

(I) EMA 趋势判断

在之前的研究中以及发现，该策略在震荡区间和上涨趋势内会达到更好的收益，而在下跌行情中会导致一定的亏损，因此在多头和空头策略中，应加入对趋势的判断，确保只在合适的趋势时执行合适的策略。

这里选用均线对趋势进行判断，经过多次参数调整，尝试 SMA 与 EMA 的多个周期，最终选用使用了 300 个周期的指数移动平均线能更有效的追踪趋势。

```
ema300 = ta.EMA(numpy.array(close), timeperiod=300)
ema_slope = ema300[-1] - ema300[-8]
ema_slope_0 = ema300[-1] - ema300[-14]
```

对趋势的判断通过 EMA 斜率，计算当前 EMA 和前几个周期的 EMA 差值来判断市场趋势的强度和方向。若 EMA 斜率 > 0 ，表示市场趋势是向上或在短期震荡向上，可以执行多头策略。若 EMA 斜率 < 0 ，表示市场趋势是向上或在短期震荡向上，可以执行空头策略。

(II) 卡尔曼滤波器趋势判断

卡尔曼滤波器是一种用于估计动态系统状态的数学工具，特别适用于有噪声或不确定性的环境中。它广泛应用于信号处理、控制系统、导航、机器人等领域。卡尔曼滤波器通过递归的方式结合了观测值和预测值，使得估计更加准确，在交易市场中，可以使用卡尔曼滤波器识别噪声，从而发现真正的趋势。

卡尔曼滤波器核心在于预测和更新，根据系统模型预测下一时刻的状态，使用新的观测值来修正预测，减小误差。

状态方程可以表示为：

$$x_k = A \cdot x_{k-1} + \omega_{k-1}$$

其中 ω 是过程噪声， A 是状态转移矩阵。

观测方程可以表示为：

$$z_k = H \cdot x_k + v_k$$

其中 z_k 是测量值， H 是观测矩阵， v 是观测噪声

基于预测值和实际观测值估计状态，卡尔曼增益表示为：

$$K_k = \frac{P_{k|k-1} \cdot H^T}{H \cdot P_{k|k-1} \cdot H^T + R}$$

其中 $P_{k|k-1}$ 是预测协方差矩阵， R 是测量噪声协方差矩阵。

基于以上计算更新状态估计和协方差矩阵：

$$x_k = x_{k|k-1} + K_k \cdot (z_k - H \cdot x_{k|k-1})$$

$$P_k = (I - K_k \cdot H) \cdot P_{k|k-1}$$

在这个过程中，有六个参数可以调整：

过程噪声协方差 Q ：过程噪声代表了系统模型的不确定性。它描述了系统本身的随机变化或者未知动态。

测量噪声协方差 R ：测量噪声描述了测量值的不确定性，通常由测量设备的精度、传感器的噪声等因素引起。

状态转移矩阵 A ：系统状态如何从上一时刻演化到当前时刻。

观测矩阵 H ：系统的观测值与状态的关系。

初始状态估计 x_0 ：卡尔曼滤波器开始时对系统状态的估计值。

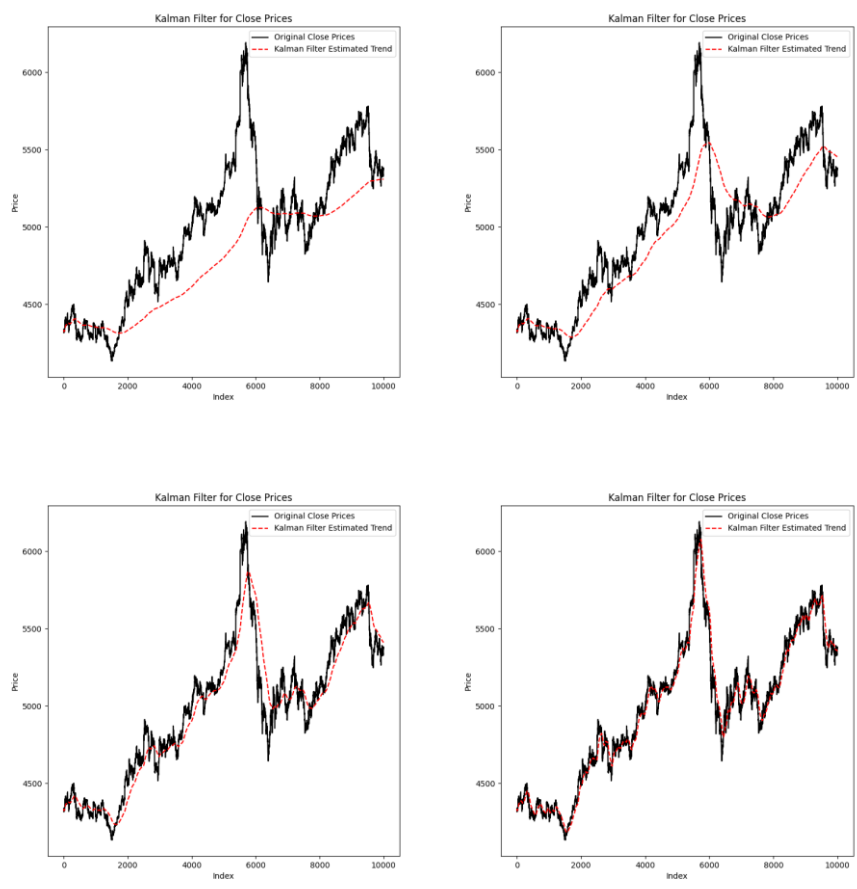
初始协方差估计 P_0 ：卡尔曼滤波器对初始状态不确定性的估计。

基于以下参数构建卡尔曼滤波器，对螺纹钢 2021 年 1 月 4 日至 2021 年 8 月 20 日数据追逐可视化：

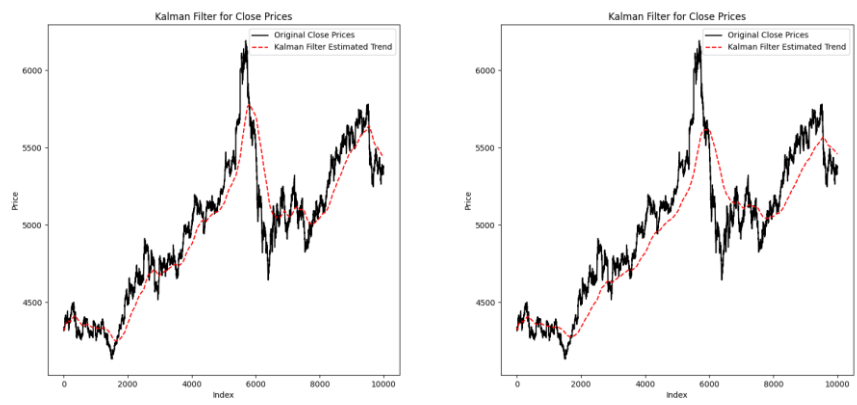
```
def kalman_filter(ts):
    n = len(ts)
    # 初始化卡尔曼滤波器参数
    x = np.zeros(n) # 估计的状态
    P = np.zeros(n) # 状态的协方差
    Q = 1e-7 # 过程噪声协方差
    R = 0.5 # 测量噪声协方差
    A = 1 # 状态转移矩阵
    H = 1 # 观测矩阵
    x[0] = ts[0] # 初始状态
    P[0] = 1.0 # 初始协方差
```

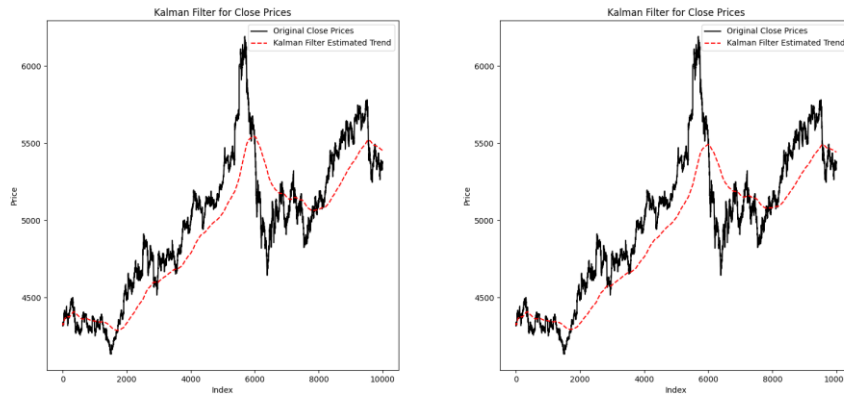
固定参数 R 不变，对 Q 参数调整，参数 Q 在 10^{-7} ， 10^{-6} ， 10^{-5} ， 10^{-4} 状态下卡

尔曼滤波器表现如下：



结合，敏感度，追踪度，此处选用 $Q = 10^{-6}$ ，并对 R 进一步优化，调整 R 参数 0.1, 0.3, 0.5, 0.7 可以认为 0.7 的情况下卡尔曼滤波器的表现最优。





因本地计算环境限制，无法迅速实时更新卡尔曼滤波器的结果，故接下来依然选用 EMA 作为趋势判断。

(2) 不同趋势的交易策略

(I) 多头交易策略

在之前的研究中，详细探索了马丁格尔策略的应用与局限性，结合多种技术指标，用来达到提高盈利同时，规避风险的目的，但始终无法避免策略本身的问题，虽然胜率达到接近 90%，但 10% 的策略失败以及市场的黑天鹅事件会对收益造成毁灭性打击，因此在这里放弃加倍买入的思路，转而使用 DCA 策略。

DCA 是一种长期投资策略，通常用于购买股票、基金等资产，其核心思想是：定期买入，降低平均成本，通过分批购买，DCA 可以平滑市场波动的影响，在长期内降低买入资产的平均成本。将原马丁格尔策略的加倍参数设置为 1，达到每次定额买入的目的。

同时，传统 DCA 策略通常不依赖市场的技术指标，而我在策略中结合了布林带和 EMA 等技术分析工具来判断买入时机。达到一个技术分析驱动 DCA，即不是在固定的时间点或价格水平买入，而是根据市场的波动和技术指标的信号来决定买入时机，通过这种操作达到收益扩大。

(II) 空头交易策略

该策略首先基于 MACD 死叉，MACD 金叉指的是 MACD 线（即短期 EMA 与长期 EMA 的差值）从下向上穿越信号线，这通常被认为是市场上涨的信号，相反，即短期 EMA 与长期 EMA 的差值从上到下穿过信号线，可以视为市场下跌的信号。

```
if context.ema5 < context.ma21 and context.last_ema5 >= context.ma21:
    if context.position >= 0:
        sell_open(context.sl, "Market", volume=10, serial_id = 1)
        context.position += 10

if context.ema5 > context.ma21 and context.last_ema5 <= context.ma21:
    if context.position > 0:
        buy_close(context.sl, "Market", volume=context.position, serial_id = 2)
        context.position = 0
```

但在实际操作中，由于长期均线和短期均线均使用 EMA，受现阶段值影响较大，因此经常会出现假突破的情况，造成交易次数的频繁，同时带来亏损，因此这里我选用双均线交易系统，对长期均线使用 SMA 用来抵消对近期波动过于敏感带来的假突破问题。

根据参数调整和多次优化，最终选用 EMA5 和 SMA21 作为组合进行交易。

(3) 策略回测

加入趋势后，可以发现，即便在大趋势上处于下跌行情，依然能正确的识别下跌行情中的小级别上涨行情，通过测策略实现多头盈利。

总体概要			
统计指标	所有交易	多头交易	空头交易
净利润	171,710.16	171,710.16	0.00
总盈利(毛利)	860,905.81	860,905.81	0.00
总亏损(毛损)	-689,195.50	-689,195.50	0.00
盈利因子(毛利/毛损)	1.25	1.25	0.00
收益率	17.17%	17.17%	0.00%
年化收益率	7.88%	7.88%	0.00%
最大资产回撤	-141,201.25	N/A	N/A
净利润/最大资产回撤	1.22	N/A	N/A
最大资产回撤幅度	-12.92%	N/A	N/A
年化收益率/最大资产回撤幅度	0.61	N/A	N/A
交易次数	236	236	0
盈利交易次数	162	162	0
亏损交易次数	74	74	0
胜率	68.64%	68.64%	0.00%

在空头策略中，通过双均线策略，同样可以实现稳定盈利。

总体概要			
统计指标	所有交易	多头交易	空头交易
净利润	156,550.47	0.00	156,550.47
总盈利(毛利)	663,824.69	0.00	663,824.69
总亏损(毛损)	-507,274.56	0.00	-507,274.56
盈利因子(毛利/毛损)	1.31	0.00	1.31
收益率	15.66%	0.00%	15.66%
年化收益率	7.21%	0.00%	7.21%
最大资产回撤	-58,808.25	N/A	N/A
净利润/最大资产回撤	2.66	N/A	N/A
最大资产回撤幅度	-5.29%	N/A	N/A
年化收益率/最大资产回撤幅度	1.36	N/A	N/A
交易次数	367	0	367
盈利交易次数	124	0	124
亏损交易次数	243	0	243
胜率	33.79%	0.00%	33.79%

在两年数据的回测中，可以发现这两种策略结合在市场中的有效性。

总体概要			
统计指标	所有交易	多头交易	空头交易
净利润	474,428.34	291,695.84	182,733.14
总盈利(毛利)	2,589,758.25	1,912,521.13	677,236.25
总亏损(毛损)	-2,115,329.25	-1,620,826.75	-494,503.25
盈利因子(毛利/毛损)	1.22	1.18	1.37
收益率	47.44%	29.17%	18.27%
年化收益率	20.43%	13.03%	8.37%
最大资产回撤	-298,641.25	N/A	N/A
净利润/最大资产回撤	1.59	N/A	N/A
最大资产回撤幅度	-17.72%	N/A	N/A
年化收益率/最大资产回撤幅度	1.15	N/A	N/A
交易次数	820	456	364
盈利交易次数	456	331	125
亏损交易次数	364	125	239
胜率	55.61%	72.59%	34.34%

CH2.3 基于机器学习的交易策略

CH2.3.1 多维技术指标与 KNN 算法结合的交易策略

(1) 技术指标选取

识别趋势的关键是识别价格的偏离，因此选取五个与价格偏离相关的技术指标构建模型：RSI，WT，CCI，ADX，ATR。

```
# 计算技术指标
data['RSI'] = ta.RSI(*args: data['收盘'], timeperiod=14)
data['ADX'] = ta.ADX(*args: data['最高'], data['最低'], data['收盘'], timeperiod=14)
data['WT'] = ta.WILLR(*args: data['最高'], data['最低'], data['收盘'], timeperiod=14)
data['CCI'] = ta.CCI(*args: data['最高'], data['最低'], data['收盘'], timeperiod=14)
data['ATR'] = ta.ATR(*args: data['最高'], data['最低'], data['收盘'], timeperiod=14)
```

RSI、WT 和 CCI 主要集中在市场的超买超卖状态和价格的极端偏离，这三个指标在判断市场的反转时起着关键作用；而 ADX 则注重趋势的强度，帮助交易者识别当前市场是否有持续性趋势，因此能有效避免在震荡市场中的错误操作；最后，ATR 通过衡量市场的波动性，辅助风险管理，帮助调整交易策略的参数。

(2) 特征工程

基于已选的五技术指标进行建模，将这五个技术指标作为特征构建特征数组。

```
# 将技术指标转化为特征数组
features = ['RSI', 'ADX', 'WT', 'CCI', 'ATR']
feature_matrix = data[features].dropna().values
```

构建特征数组后，设置目标变量，这里目标变量为价格的涨跌，如果未来的收盘价高于当前收盘价，视为 1，如果未来的收盘价低于当前收盘价视为 0。

```
# 计算收盘价格的未来涨跌行为
data['future_close'] = data['收盘'].shift(-1)
data['price_change'] = np.where(data['future_close'] > data['收盘'], 1, 0)
```

(3) 曼哈顿距离计算

目前已经设定好特征数组和目标变量，在 KNN 算法之前需要对各个样本进行度量，在度量这一部分，常见的距离度量有欧氏距离，闵可夫斯基距离，考虑到使用技术指标时，各个指标之间可能存在非线性的关系。曼哈顿距离对每个特征值的绝对差值进行求和，较为简单且能够处理指标间的非线性关系。尤其在金融市场数据中，特征之间常常存在复杂的相互作用，而曼哈顿距离相对能够更好地捕捉到这种相互关系，因此选用曼哈顿距离进行计算。

```
# 计算特征矩阵的洛伦兹距离
lorentzian_distance = cdist(feature_matrix, feature_matrix, metric='cityblock')
knn.fit(lorentzian_distance, data['price_change'])
```

(4) KNN 算法

使用曼哈顿距离完成对样本度量后，使用 KNN 算法计算按照以下流程计算：

选择参数 K：选择在预测时考虑的最近邻居数量。一个事先设定的超参数，通常通过交叉验证来选择最合适的值。

计算距离：通过上一部分计算的曼哈顿距离，计算待预测样本与训练集中所有样本之间的距离。

选择最近的 K 个邻居：根据距离选择出离待预测样本最近的 K 个样本。

投票：对于分类任务，KNN 会根据这 K 个邻居的标签进行投票，选择出现频率最高的标签作为预测结果。预测的标签是 1（上涨）或 0（下跌）。

```
# 使用 KNN 进行市场行为预测
knn = KNeighborsClassifier(n_neighbors=5, metric='precomputed')
# 计算特征矩阵的洛伦兹距离
lorentzian_distance = cdist(feature_matrix, feature_matrix, metric='cityblock')
knn.fit(lorentzian_distance, data['price_change'])

# 进行预测
predictions = knn.predict(lorentzian_distance)
```

(5) 策略回测

在实际应用中，样本的大小会很大程度影响结果的准确性，受限于本机内存大

小，选取 20000 条 K 线回测，在这里随机可视化 100 个 k 线的策略结果如下，可以看到这种结合多维指标的 KNN 算法基本能有效做到分类。



CH2.3.2 基于多维指标特征工程的异构集成学习预测模型

(1) 特征构建

首先读取数据，基于已有数据构建特征，选取 14 个指标群：

基础价格特征，RSI 指标族，趋势指标，威廉指标和 CCI，ATR 波动率指标，

MACD 指标，布林带指标，KDJ 指标，移动平均线，均线差值和交叉特征，成交量指标，动量和波动率特征，价格形态特征以及综合特征。

构建所有特征后，将所有特征定义成特征列表，合并所有的特征：

```
# 定义特征列表
price_features = ['log_return', 'price_range']
momentum_features = ['RSI', 'RSI_7', 'RSI_21', 'RSI_divergence', 'CCI', 'CCI_trend', 'MOM', 'ROC', 'WT']
trend_features = ['ADX', 'ADX_strong_trend', 'MACD', 'MACD_signal', 'MACD_hist', 'MACD_hist_change', 'MA5_10_diff',
                  'MA10_20_diff', 'MA20_60_diff', 'MA5_10_cross', 'MA10_20_cross', 'trend_strength']
volatility_features = ['ATR', 'ATR_ratio', 'BB_width', 'BB_position', 'volatility_5', 'volatility_10']
volume_features = ['OBV', 'OBV_ratio']
pattern_features = ['KDJ_cross', 'swing_high', 'swing_low', 'breakout_up', 'breakout_down']
other_features = ['K', 'D', 'J', 'price_pct_change', 'price_pct_change_5', 'price_pct_change_10']

# 合并所有特征
all_features = (price_features + momentum_features + trend_features + volatility_features +
               volume_features + pattern_features + other_features)
```

特征类别	示例特征	作用
价格特征	log_return, price_range	反映价格变动和波动范围
动量特征	RSI, RSI_divergence, CCI, MOM, ROC, WT	衡量价格趋势的强度和持续性
趋势特征	ADX, MACD, MA5_10_diff, MA10_20_cross	识别市场趋势和趋势反转点
波动率特征	ATR, BB_width, volatility_5	量化市场波动性
成交量特征	OBV, OBV_ratio	评估市场资金流向
价格形态特征	KDJ_cross, swing_high, breakout_up	识别特殊价格结构和突破
其他特征	K, D, J, price_pct_change	额外补充的分析指标

(2) 特征选择

特征选取的目标是从完整特征池中筛选出最有助于预测未来价格方向的特征。不同的模型使用了不同的特征选取方法，主要包括：基于特征重要性的选取（XGBoost、LightGBM、随机森林），基于信息增益的选取（朴素贝叶斯），基于时序相关性的选取（LSTM）。

对 XGBoost 和 LightGBM 模型，使用 GBDT 选取特征，GBDT 采用了梯度提升的方式选取特征：先训练第一棵树，计算第一棵树的预测误差，并让第二棵树学习这个误差，依次重复，建立多个决策树，每棵树都在减少上一棵树的误差，所有树的加权组合最终给出预测结果。

```
# 1. XGBoost特征选择
xgb_selector = XGBClassifier(n_estimators=100, max_depth=5, learning_rate=0.1)
xgb_selector.fit(X_train, y_train)
xgb_importance = xgb_selector.feature_importances_
xgb_importance_df = pd.DataFrame({'feature': all_features, 'importance': xgb_importance})
xgb_importance_df = xgb_importance_df.sort_values(by: 'importance', ascending=False)
xgb_features = xgb_importance_df['feature'][:25].tolist() # XGBoost选择前25个特征

# 2. LightGBM特征选择
lgbm_selector = LGBMClassifier(n_estimators=100, max_depth=5, learning_rate=0.1, verbosity=-1)
lgbm_selector.fit(X_train, y_train)
lgbm_importance = lgbm_selector.feature_importances_
lgbm_importance_df = pd.DataFrame({'feature': all_features, 'importance': lgbm_importance})
lgbm_importance_df = lgbm_importance_df.sort_values(by: 'importance', ascending=False)
lgbm_features = lgbm_importance_df['feature'][:20].tolist() # LightGBM选择前20个特征
```

对随机森林模型，特征选取基于决策树，通过计算特征对目标变量的重要性来筛选最重要的特征，具体来看，本策略中使用基尼重要性，基尼系数用于衡量数据纯度，决策树的每个节点都会基于某个特征进行分裂，减少不纯度。特征的重要性由该特征分裂节点所带来的基尼指数减少量决定。

```
# 3. 随机森林特征选择
rf_selector = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
rf_selector.fit(X_train, y_train)
rf_importance = rf_selector.feature_importances_
rf_importance_df = pd.DataFrame({'feature': all_features, 'importance': rf_importance})
rf_importance_df = rf_importance_df.sort_values(by: 'importance', ascending=False)
rf_features = rf_importance_df['feature'][:25].tolist() # 随机森林选择前25个特征
```

与以上模型不同，朴素贝叶斯假设特征之间是相互独立的，因此需要选择那些对目标变量 单独 贡献最大的信息的特征。在这里我们使用互信息方法来筛选特征，衡量特征与目标变量之间相关性。

```
# 4. 朴素贝叶斯特征选择 - 使用信息增益(互信息)
# 对于朴素贝叶斯，选择信息性强的特征
mi_selector = SelectKBest(mutual_info_classif, k=15) # 选择前15个特征
mi_selector.fit(X_train, y_train)
mi_scores = mi_selector.scores_
mi_df = pd.DataFrame({'feature': all_features, 'mi_score': mi_scores})
mi_df = mi_df.sort_values(by: 'mi_score', ascending=False)
nb_features = mi_df['feature'][:15].tolist() # 朴素贝叶斯选择前15个互信息最高的特征
```

对 LSTM 模型，代码选择了与时间序列高度相关的特征，包括：价格特征，波动率特征，动量特征，趋势特征，交易量相关特征。

```
# 5. LSTM特征选择 - 时序相关性(自相关)和波动性特征
# LSTM通常对时序相关性和波动性特征敏感
lstm_features = []
# 添加价格和波动率特征
lstm_features.extend(price_features)
lstm_features.extend(volatility_features)
# 添加动量和趋势特征的子集
lstm_features.extend([f for f in momentum_features if '趋势' in f or 'trend' in f or 'RSI' in f or 'MOM' in f])
lstm_features.extend([f for f in trend_features if 'MACD' in f or 'MA' in f or 'ADX' in f])
# 保留最多30个特征
lstm_features = list(set(lstm_features))[:30]
```

(3) 模型训练

XGBoost 基于梯度提升树的模型，通过多棵决策树组合进行预测，每棵树学习前一棵树的残差，随后计算每个特征的信息增益选择最优分裂点，自动处理缺失值，并支持 L1/L2 正则化来防止过拟合，同时使用 logloss 作为优化目标。

```
# XGBoost模型
xgb_model = XGBClassifier(
    n_estimators=200,
    max_depth=5,
    learning_rate=0.01,
    subsample=0.8,
    colsample_bytree=0.8,
    min_child_weight=5,
    gamma=0.1,
    reg_alpha=0.1,
    reg_lambda=1.0,
    scale_pos_weight=(1 - y_train.mean()) / y_train.mean(),
    eval_metric='logloss'
)
```

LightGBM 训练比 XGBoost 更快，采用叶子优先生长的方式训练决策树。适合大规模数据，支持类别变量直接训练，同时有较低的内存占用，更适合高维数据。

```
# LightGBM模型
lgbm_model = LGBMClassifier(
    n_estimators=200,
    max_depth=5,
    learning_rate=0.01,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_alpha=0.1,
    reg_lambda=1.0,
    min_child_samples=20,
    min_split_gain=0.01,
    scale_pos_weight=(1 - y_train.mean()) / y_train.mean(),
    verbosity=1
)
```

随机森林训练使用 Bagging 方法，训练多棵独立的决策树，每棵树使用不同的随机样本和特征。通过投票机制或平均机制来生成最终预测结果，这种算法训练速度较快，对异常值不敏感。在本策略中训练 200 棵独立决策树，降低方差，提高泛化能力，同时针对数据集类别不平衡情况自动调整权重。

```
# 随机森林模型
rf_model = RandomForestClassifier(
    n_estimators=200,
    max_depth=10,
    min_samples_split=10,
    min_samples_leaf=5,
    class_weight='balanced',
    random_state=42
)
```

朴素贝叶斯模型基于概率的分类模型，适用于特征独立假设，通过计算每个特征与目标变量的条件概率，并根据贝叶斯公式进行预测，适用于多维数据。

```
# 朴素贝叶斯模型 - 处理特征不同，使用高斯朴素贝叶斯
nb_model = GaussianNB()
```

LSTM 训练适用于时间序列预测，能够捕捉短期趋势和长期趋势，通过 LSTM 层记住长期信息，并使用全连接层进行最终分类，这里使用 Dropout 防止过拟合。

```

# LSTM模型 - 保持原有架构
lstm_model = Sequential([
    LSTM(64, return_sequences=True, input_shape=(time_steps, len(lstm_features))),
    Dropout(0.2),
    LSTM(32),
    Dropout(0.2),
    Dense(units=16, activation='relu'),
    Dense(units=1, activation='sigmoid')
])

lstm_model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)

```

模型	选取特征数	训练方式
XGBoost	25	梯度提升树，学习残差
LightGBM	20	GBDT，叶子优先生长
随机森林	25	Bagging，投票机制
朴素贝叶斯	15	互信息特征选择，计算概率
LSTM	30	时间序列预测，记忆长期信息

(4) 元学习器

单个模型可能存在以下问题：XGBoost / LightGBM 可能对趋势特征敏感，但容易忽略短期波动；随机森林不擅长时间序列数据，容易受特征选择影响；朴素贝叶斯 计算高效，但假设特征相互独立，无法捕捉复杂关系；LSTM 能捕捉时序关系，但可能过拟合短期模式。

元学习器是一个用于融合多个基础模型预测结果的神经网络，使用 Stacking 方法实现。它可以收集多个基础模型的预测概率并学习它们之间的关系，找到最优的组合方式，提升整体预测性能，最终输出优化后的预测结果，减少单个模型的偏差，提高稳定性。

本策略中，元学习器是 MLP，多层感知机，它的输入是基础模型的预测概率，输出最终的分类预测。

```

# 创建元学习器 - 使用加权融合的神经网络
from tensorflow.keras.layers import Input, Dense, Concatenate
from tensorflow.keras.models import Model

# 元学习器结构优化 - 增加层数和特征交互
meta_input = Input(shape=(5,)) # 5个基模型的预测概率
meta_hidden1 = Dense(units=12, activation='relu')(meta_input)
meta_hidden2 = Dense(units=8, activation='relu')(meta_hidden1)
meta_output = Dense(units=1, activation='sigmoid')(meta_hidden2)
meta_model = Model(inputs=meta_input, outputs=meta_output)

meta_model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

```

基于已训练的元学习器，在测试集上进行最终预测，确保集成模型的准确率和泛化能力。

```
# 使用元学习器进行最终预测
ensemble_test_proba = meta_model.predict(meta_X_test).flatten()

# 评估各个基础模型和集成模型在验证集上的性能
xgb_val_preds = (xgb_val_proba_aligned >= 0.5).astype(int)
lgbm_val_preds = (lgbm_val_proba_aligned >= 0.5).astype(int)
rf_val_preds = (rf_val_proba_aligned >= 0.5).astype(int)
nb_val_preds = (nb_val_proba_aligned >= 0.5).astype(int)
lstm_val_preds = (lstm_val_proba >= 0.5).astype(int)
meta_val_preds = (meta_model.predict(meta_X_val).flatten() >= 0.5).astype(int)
```

(5) 阈值优化

在二分类任务中，默认的分类阈值 0.5 并不总是最优的，特别是在类别不平衡或错误分类成本不均衡的情况下。在本策略中，遍历 0.3 到 0.7 之间的多个阈值基于最大化准确率和最大化 F1-score 优化阈值。

```
def find_optimal_threshold_for_meta(meta_proba, y_true, criterion='accuracy'):
    """
    寻找异构集成模型的最优预测阈值
    """
    # 限制阈值范围在0.3-0.7之间，避免极端值
    thresholds = np.arange(0.3, 0.7, 0.01)
    scores = []

    for threshold in thresholds:
        y_pred = (meta_proba >= threshold).astype(int)

        if criterion == 'accuracy':
            # 准确率
            accuracy = np.mean(y_pred == y_true)
            scores.append(accuracy)
        elif criterion == 'f1':
            # F1分数
            precision = np.sum((y_pred == 1) & (y_true == 1)) / (np.sum(y_pred == 1) + 1e-10)
            recall = np.sum((y_pred == 1) & (y_true == 1)) / (np.sum(y_true == 1) + 1e-10)
            f1 = 2 * precision * recall / (precision + recall + 1e-10)
            scores.append(f1)
        # 其他标准可以根据需要添加...

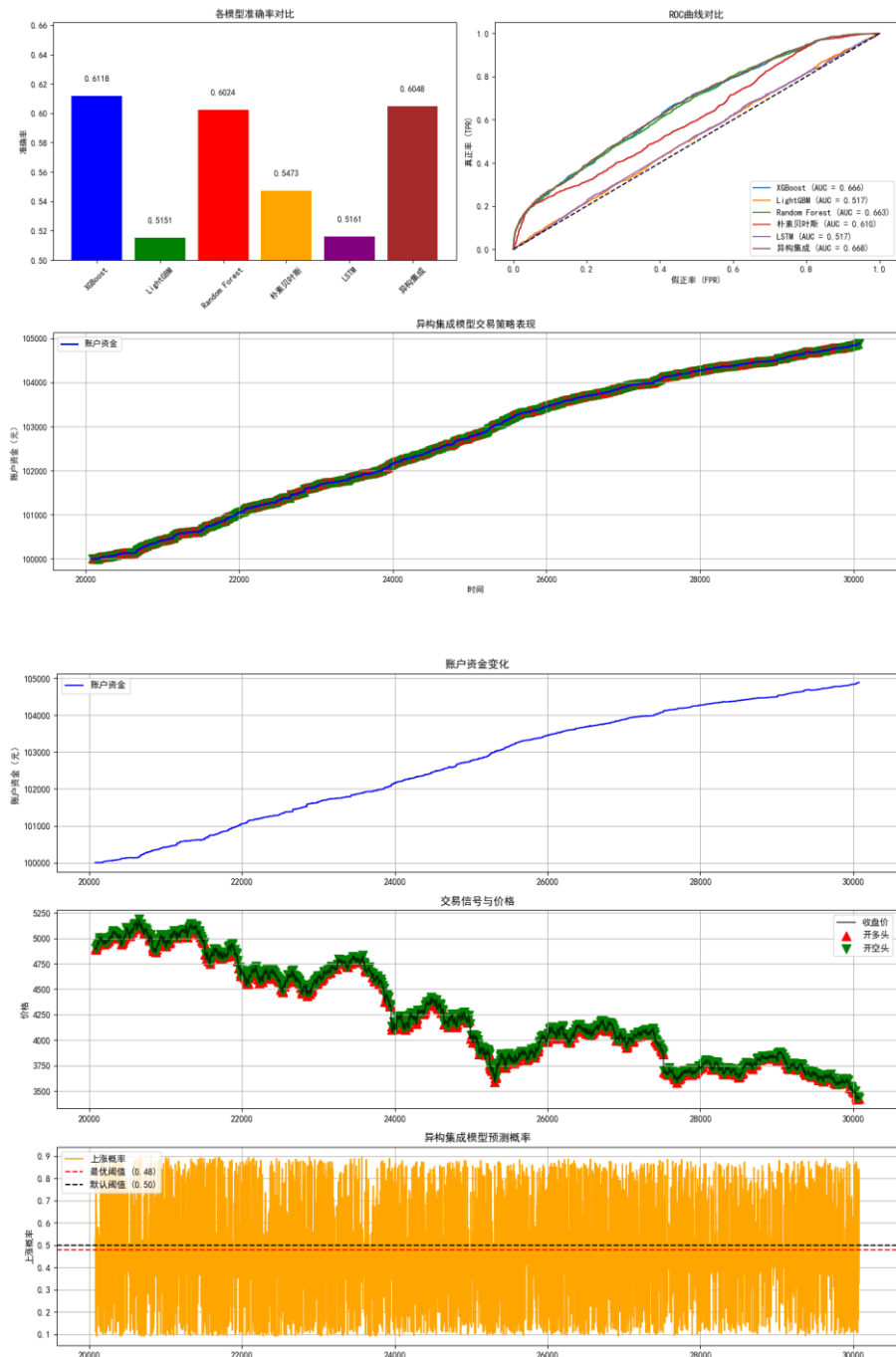
    # 找到最高分数对应的阈值
    optimal_threshold = thresholds[np.argmax(scores)]
    return optimal_threshold
```

(6) 模型结果

选取螺纹钢主力连续期货 5 分钟 k 线，2021 年至 2025 年数据回测，将前 20000 行作为训练集，20000-30000 行作为测试集进行模型训练，训练结果如下：

初始资金: 100000 最终资金: 104887.55 最大回撤: 0.02%

总交易次数: 2669 盈利交易次数: 1882 交易胜率: 70.51%



CH2.4 参数动态调整思路

CH2.4.1 遍历优化思路

通过对多个参数组合进行遍历，找到最适合当前市场参数，并实时更新优化。这里采用这个思路对布林通道优化，使用两层循环来遍历所有可能的 period（周期）和 dev（标准差倍数）的组合。对每一组参数组合进行模拟交易，通过模拟交易选取最优参数并实时更新。

```
def optimize_bb_parameters(close_prices, min_period=10, max_period=30, min_dev=0.5, max_dev=2.0):
    best_profit = float('-inf')
    optimal_period = 20
    optimal_dev = 1.0
    # 使用最近的500个数据点进行优化
    test_data = close_prices[-500:] if len(close_prices) > 500 else close_prices

    for period in range(min_period, max_period + 1, 2): # 步长为2
        for dev in numpy.arange(min_dev, max_dev + 0.1, 0.1): # 步长为0.1
            upper, middle, lower = ta.BBANDS(
                test_data,
                timeperiod=period,
                nbdevup=dev,
                nbdevdn=dev,
                matype=0
            )

            # 模拟交易
            profit = 0
            position = 0
            entry_price = 0

            for i in range(len(test_data)):
                if numpy.isnan(upper[i]) or numpy.isnan(lower[i]):
                    continue

                if test_data[i] <= lower[i] and position == 0:
                    # 买入信号
                    position = 1
                    entry_price = test_data[i]
                elif test_data[i] >= upper[i] and position == 1:
                    # 卖出信号
                    profit += test_data[i] - entry_price
                    position = 0

            if profit > best_profit:
                best_profit = profit
                optimal_period = period
                optimal_dev = dev

    return optimal_period, optimal_dev
```

CH2.4.2 基于聚类算法的因子优化

(1) 研究思路

在实际交易中，为了加入根据实际情况动态调整的部分，此处展开因子驱动的趋势聚类研究。

```
# 计算超级趋势表现值 (perf)
1 个用法
def calculate_supertrend(df, factor):
    df['supertrend_upper'] = (df['最高'] + df['最低']) / 2 + df['atr'] * factor
    df['supertrend_lower'] = (df['最高'] + df['最低']) / 2 - df['atr'] * factor
    df['supertrend'] = np.where(df['收盘'] > df['supertrend_upper'], 1,
                                np.where(df['收盘'] < df['supertrend_lower'], 0, np.nan))
    df['supertrend'] = df['supertrend'].ffill() # 使用前向填充
    df['perf'] = df['收盘'].diff() * df['supertrend'].shift(1) # 计算表现值
    return df
```

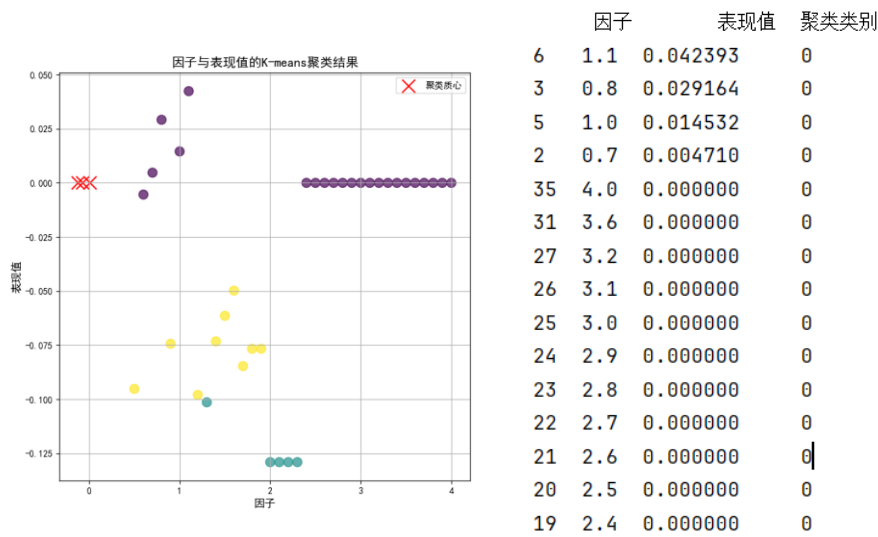
为了方便进一步研究，此处选用超级趋势指标，超级趋势是一个趋势跟踪指标，基于市场的平均真实波幅和一个调整因子来计算上下边界，它帮助识别市场当前的趋势：上涨、下跌或震荡。通过调整因子，计算表现值，表现值根据趋势方向与收盘价的变化之间的关系来衡量的。

```
# 进行K-means聚类
kmeans = KMeans(n_clusters=3, random_state=0) # 设置为3个聚类
kmeans.fit(perf_values)
```

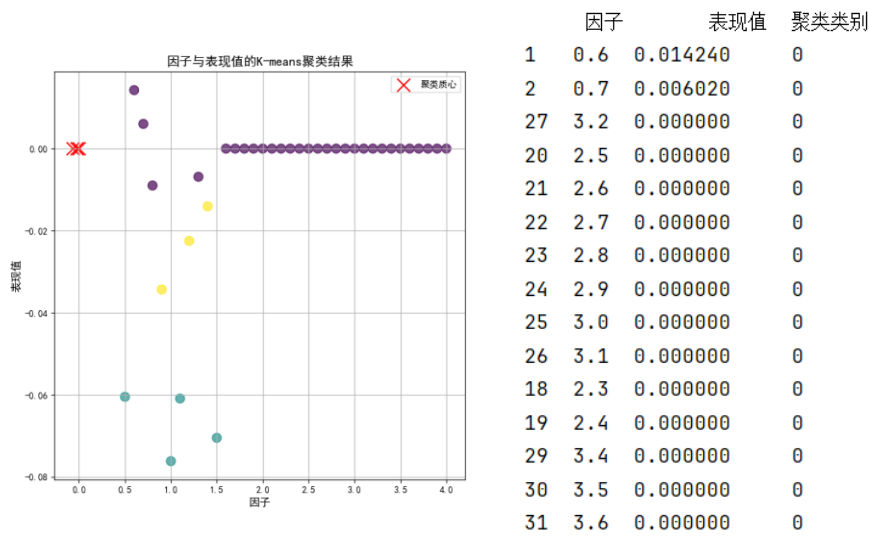
在计算表现值后，使用 K-means 聚类算法将不同因子下的表现值分为 3 个类别（最优、平均和最差）。聚类分析帮助识别出哪些因子在当前市场条件下表现得最好，哪些因子表现较差。

(2) 研究结果

研究样本选取螺纹钢 2021 年 1 月 4 日至 2025 年 1 月 27 日 5 分钟 k 线数据，此处对因子选取最小值为 0.5，最大值为 4，步长设置为 0.1，选取 10000-20000 行数据应用模型结果如下：



选取 30000-40000 行数据应用模型结果如下：



(3) 进一步泛化

从理论层面可以在实盘中对其进行动态调整，但受限于算力和速度，很难做到及时更新，同时对于因子，应采用一些方法对因子平滑，避免因子的多次连续变换影响交易。

此处仅对 ATR 因子进行研究，在进一步的研究中，可以借助类似思想对均线日期，布林带标准差倍数等运用 kmean 算法更新最优参数。

CH3 敬畏市场

在任何市场中，并不存在一个永久盈利的“印钞机”模型，因为在任何时刻，市场的行为是不可预测的，市场是动态且不可预测的。消费者的需求、技术的发展、政策的变化以及外部环境的波动都能对市场产生深远影响。作为一个普通市场参与者，我们必须时刻保持敏感，洞察市场的变化并做出及时的调整。忽视这些变化，盲目自信，往往会导致市场的失利。因此，敬畏市场意味着时刻保持谨慎、谦逊，意识到自己的决策并非万能，必须根据市场的反馈不断修正和优化。

同时，在每一次交易中，我们不可能做到想象中所谓“买在最低点”，“卖在最高点”，或者说同样也没人知道这个标的会“涨到哪里”，“跌到哪里”，我们也无需听信别人的所谓“铁底”，“见顶”，按照自己的思路，在市场波动中不断质疑自己，不断优化自己，可以踏空，不要去赌。

此外，市场的牛市虽然令人兴奋，但也充满了短期的波动和过度的乐观情绪，这种市场气氛往往会让人忽视长期的风险。而在熊市中，许多人会因为眼前的亏损而放弃掉原本具备潜力的机会。真正成功的市场参与者往往拥有较为长远的视角，他们能看到市场的周期性波动，而不是仅仅局限于当下的盈亏。真正成功的市场参与者往往不止经历牛市，而是在牛市和熊市中都保持正确的心态，保持坚定的信仰。

写在最后

在完成这些内容的时候恰逢两会召开，美股回调，加密货币市场巨大波动，所有市场都是这样，有时候表面风平浪静，实际上暗潮汹涌，有时候看起来风险巨大，实际上却是机会，很幸运能在 20 岁接触到市场中的一些理论，认识一些成功的市场参与者，人生海海，翘首以盼。

最后感谢成都正福金融工程研究院的老师，YouTube 频道“區塊鏈日報 Blockchain Daily”，YouTube 频道“韭菜翻轉日誌”，YouTube 频道“LEI”对我个人交易思想的影响。