**技术文档：天玑300智能低吸交易系统**

## 项目概述

**1.1 项目名称**

天玑300智能低吸交易系统

## **1.2 项目简介**

在瞬息万变的股票市场中，投资者需要高效、精准的工具来把握交易时机。天玑300智能低吸交易系统应运而生，它是一款基于 Python 开发的先进股票数据分析与自动化交易管理软件。该系统深度融合了实时数据获取、布林线计算、交易逻辑、多线程管理以及 AI 实时分析等核心技术，为用户打造了一个集股票监控、数据分析、智能预测和自动交易于一体的综合性平台。系统的设计初衷是帮助投资者快速响应股票市场的动态变化，凭借科学的数据分析、智能的交易策略以及 AI 的精准预测，实现高效、精准的交易操作，从而在复杂的市场环境中提升投资收益，且该系统的设计最高可监控高达20支股票。

## 2. 系统架构

**2.1 系统组成**

系统主要由以下模块组成：

**图形用户界面（GUI）模块**

 **功能定位**： 作为用户与系统交互的桥梁，提供直观、友好的交互式界面。用户可以通过该界面输入股票代码、设置各种交易参数、实时查看股票数据、AI 分析结果，并便捷地执行交易逻辑。

 **技术实现**：基于 Tkinter 库实现， Tkinter 是 Python 内置的标准 GUI 库，具有简单易用、跨平 台等优点，能够快速搭建出稳定且美观的用户界面。

**数据处理模块**

 **功能定位**：是系统的数据核心处理单元，负责对海量的股票数据进行处理，包括实时数据的精准获取、布林线的精确计算、数据的清理和筛选等操作，为后续的交易决策和 AI 分析提供可靠的数据支持

 **技术实现**：使用 NumPy 和 Pandas 库实现高效的数据处理。 NumPy 提供了强大的多维数组对象和 各种数学函数，能够快速处理大规模数据； Pandas 则提供了灵活的数据结构和数据操作方法，方 便进行数据的清洗、分析和存储。

 **数据过滤优化策略**：鉴于同一时间的数据请求量极为庞大，为提升效率，我们摒弃了传统的串行请 求方式，转而采用更为高效的并行请求。然而，受网络延迟的影响，并行请求返回的数据不仅顺序 混乱，还存在数据缺失的问题。

为解决这一难题，我们制定了如下策略：将短时间内两次并行请求所获取的数据整合至同一个容器 中。随后，运用贪心算法对这些数据进行排列。完成排列后，将处理后的数据与前一天相同监控范 围内的股票数据进行细致对比，以此实现数据的准确排序。

**交易逻辑模块**

 **功能定位**：实现买入和卖出的核心逻辑，是系统的决策大脑。该模块结合布林线、实时数据以及 AI 分析结果，精确判断交易条件，为用户提供科学的交易建议，并通过与同花顺客户端的交互接口，实现自动化交易。

 **技术实现**：提供与同花顺客户端的交互接口，通过调用同花顺的 API 实现股票的买入和卖出操作。

**网络请求模块**

 **功能定位**：负责与外部 API 进行通信，通过网络请求获取实时行情和股票数据，确保系统数据的及 时性和准确性。

 **技术实现**：通过 requests 库实现网络请求， requests 是一个简洁而优雅的 <HTTP> 库，能够方便 地发送各种 <HTTP> 请求，处理响应数据。

**多线程管理模块**

 **功能定位**：使用多线程技术实现任务的并行处理，确保实时数据获取和交易逻辑的高效运行，避免 因单线程处理导致的性能瓶颈。

 **技术实现**：使用 ThreadPoolExecutor 和 threading 实现多线程任务管理。

ThreadPoolExecutor 提供了线程池的功能，能够自动管理线程的创建和销毁； threading 则提 供了更底层的线程操作方法，方便实现复杂的线程控制。

AI 实时分析模块

 **功能定位**：利用先进的人工智能技术，对实时股票数据进行深度分析，挖掘数据背后的潜在规律和趋势，为交易决策提供额外的参考依据。该模块能够预测股票价格的短期走势、波动幅度等关键信息，帮助用户更精准地把握交易时机。

 **技术实现**：本技术方案的实现路径为：首先，通过外部接口调用OpenAI GPT系列的API，利用其先进的自然语言处理算法。接着，在系统内部引入 JSON 文件，这一文件承担着数据载体的重要角色，将实时数据以及 “Ratio” 的详细含义以标准化、结构化的形式进行封装。随后，把这些数据 作为提示词传递给API调用，以引导模型生成与实时信息和特定概念紧密相关的回复。

**2.2 系统数据流**

以下是系统的数据流示意图：

|  |
| --- |
| 用户输入股票代码和参数 -> GUI模块 -> 数据处理模块 -> 网络请求模块 -> 实时数据获取 -> 数据筛选 和布林线计算 -> 交易逻辑模块 -> 自动买入/卖出 |

详细解释：用户通过 GUI 模块输入股票代码和交易参数，数据被传递到数据处理模块。数据处理模块调用网络请求模块，通过外部 API 获取实时股票数据。获取到的数据经过清理和筛选后，进行布林线计算。AI 实时分析模块对处理后的数据进行深度分析，预测股票的走势和波动。交易逻辑模块结合布林线、实时数据以及 AI 分析结果判断交易条件，若满足条件则触发自动买入或卖出操作。

**3. 功能说明**

**3.1 图形用户界面（GUI）**

**功能描述**

 提供友好的用户界面，用户可以通过界面操作完成股票代码输入、参数设置、实时数据查看、AI 分析结果展示和交易逻辑执行。界面设计遵循简洁易用的原则，即使是初学者也能快速上手。

**主要组件**

1. **股票代码输入框**

 用户可以输入股票代码，支持自动校验格式。例如，输入 300657.sz 时，系统会自动检查代码的 格式是否正确，若格式错误会给出相应的提示信息。

. 支持批量输入股票代码，用户可以一次性输入多个股票代码（最高可同时监控20支），方便对多只 股票进行监控和交易。

2. **令牌输入框**

 设置 API 的访问令牌，用于获取实时数据。用户需要输入有效的访问令牌，系统才能正常访问外部 API 获取数据。

 提供令牌验证功能，若输入的令牌无效，系统会提示用户重新输入。

3. **延时时间输入框**

 设置延时参数（秒），控制交易逻辑的执行频率。用户可以根据自己的交易策略和市场情况，灵活 调整延时时间。

 提供默认值，若用户未输入延时时间，系统会使用默认值进行交易逻辑的执行。

4. **比例阈值输入框**

 设置布林线的比例阈值，用于筛选股票。该阈值决定了股票价格与布林线的相对位置，是交易决策 的重要参数之一。

 提供可视化的阈值调整工具，用户可以通过滑动条等方式直观地调整比例阈值。

5. **数据展示面板**

 实时显示股票的当前价格、布林线数据以及 AI 分析结果。数据展示面板采用图表的形式，直观地展示股票价格的走势、布林线的变化情况以及 AI 预测的价格趋势。

 支持数据的实时刷新，确保用户能够及时获取最新的股票信息和AI分析结果。

 实时获取不同股票比例阈值的走势，以提醒使用者该所选股票的走势。

6. **按钮**

 **添加股票代码**：将输入的股票代码添加到列表。点击该按钮后，系统会自动检查股票代码的有效 性，若有效则将其添加到监控列表中。

 **删除选中股票代码**：删除用户选中的股票代码。用户可以在股票代码列表中选择需要删除的股票代 码，点击该按钮即可将其从列表中删除。

 **清空所有股票代码**：清空股票代码列表。点击该按钮后，系统会将股票代码列表中的所有代码清 空，方便用户重新设置监控列表。

 **执行买入逻辑**：启动买入逻辑。点击该按钮后，系统会根据用户设置的参数、AI分析结果和交易逻辑，自动筛选 满足买入条件的股票，并执行买入操作。

 **退出程序**：关闭程序。点击该按钮后，系统会保存当前的设置和数据，然后退出程序。

**3.2 数据处理模块**

**功能描述**

 提供股票数据的实时处理，包括布林线计算、数据清理和筛选。该模块能够对海量的股票数据进行 高效处理，确保数据的准确性和及时性，为 AI 分析提供高质量的数据。

**主要功能**

1. **布林线计算**

. 使用 20 个时间窗口计算股票价格的布林线。布林线是一种常用的技术分析指标，能够反映股票价 格的波动范围和趋势。

 计算公式：

 平均值（SMA）： SMA = mean(window)

 标准差（SD）： SD = std(window)

 下布林线： Lower Band = SMA - k \* SD ，其中 k 默认为 2。

 代码实现：

|  |
| --- |
| def calculate\_bollinger\_bands(self, min\_value=-9999, max\_value=9999): window\_size = 20  k = 2.0  all\_lower\_band = []  for price\_15min in self.close\_prices\_15min\_new\_:  prices = np.array(price\_15min)  sma = np.mean(prices[-window\_size:])  sd = np.std(prices[-window\_size:])  lower\_band = sma - k \* sd  all\_lower\_band.append(lower\_band)  return np.array(all\_lower\_band) |

2. **数据清理**

 删除不符合条件的股票数据。例如，删除价格为零或无效的数据，确保数据的质量。

 移除黑名单中的股票。用户可以设置黑名单，系统会自动过滤掉黑名单中的股票数据。

3. **实时数据处理**

. 每 15 分钟获取一次收盘价，处理后存储为新的数据集。通过定时任务，系统会自动从外部 API 获 取最新的收盘价数据，并进行处理和存储。

 对新获取的数据进行质量检查，若数据异常会进行相应的处理或提示用户。

**3.3 交易逻辑模块**

**功能描述**

 实现买入和卖出的核心逻辑，结合布林线和实时数据判断交易条件。该模块是系统的核心决策模 块，能够根据市场情况自动做出交易决策。

**主要功能**

1. **买入逻辑**

 检查股票价格是否低于布林线的下限。若股票价格低于布林线的下限，说明股票可能被低估，具有 投资价值。

 将满足条件的股票加入待买入列表。系统会对满足买入条件的股票进行排序，根据用户设置的优先 级和资金情况，确定买入的股票和数量。

 代码实现：

|  |
| --- |
| def buy(self):  if self.state:  print('您想同时进行多笔交易 ')  for stock\_symbol, price, amount in zip(self.stock\_symbols, self.prices, self.amounts):  self.user.buy(stock\_symbol, price, amount)  print(self.user.buy(stock\_symbol, price, amount))  self.state\_buy = True else:  print('您选择购买一只股票 ')  self.user.buy(str(self.stock\_symbols), self.prices, self.amounts)  print(self.user.buy(str(self.stock\_symbols), self.prices, self.amounts)) |

1. **卖出逻辑**

 根据用户输入的股票代码、价格和数量执行卖出操作。用户可以通过 GUI 界面输入卖出的股票代 码、价格和数量，系统会自动执行卖出操作。

 支持多笔交易同时进行。系统可以同时处理多只股票的卖出操作，提高交易效率。

 提供卖出策略建议。系统会根据股票的实时数据和布林线指标，为用户提供卖出策略建议，帮助用 户做出更合理的卖出决策。

 检查股票价格是否高于预设的止盈点，或者 AI 分析结果显示该股票短期内有下跌趋势。若满足其中一个条件，为了锁定利润或避免损失，应考虑卖出股票。

 将满足条件的股票加入待卖出列表。系统会根据用户的持仓情况和交易规则，确定卖出的股票和数量。

**3.4 网络请求模块**

**功能描述**

 通过外部 API 获取股票数据和实时行情。该模块确保系统能够及时获取最新的股票信息，为交易决 策提供数据支持。

**主要功能**

1. **获取访问令牌**

 请求 URL： <https://quantapi.51ifind.com/api/v1/get_access_token>

 请求方式： POST

 请求头：

|  |
| --- |
| {  "Content-Type": "application/json",  "refresh\_token": "用户提供的刷新令牌" } |

 代码实现：

|  |
| --- |
| def init(self):  getAccessTokenUrl = '<https://quantapi.51ifind.com/api/v1/get_access_token>'  getAccessTokenHeader = {"Content-Type": "application/json", "refresh\_token":  self.refreshtoken} try:  getAccessTokenResponse = requests.post(url=getAccessTokenUrl, headers=getAccessTokenHeader)  getAccessTokenResponse.raise\_for\_status()  response\_data = json.loads(getAccessTokenResponse.content)  accessToken = response\_data['data']['access\_token']  self.thsHeaders = {"Content-Type": "application/json", "access\_token":  accessToken}  except requests.RequestException as e:  print(f"请求出错: {e}")  except (KeyError, json.JSONDecodeError) as e:  print(f"解析响应出错: {e}") |

2. **获取实时行情**

 请求 URL： <https://quantapi.51ifind.com/api/v1/real_time_quotation>

 请求方式： POST

 请求参数：

|  |
| --- |
| {  "codes": "股票代码",  "indicators": "需要获取的指标" } |

 提供数据缓存机制。为了减少网络请求次数，提高系统性能，系统会对获取到的实时行情数据进行 缓存，在一定时间内直接使用缓存数据，避免重复请求。

**3.5 多线程管理模块**

**功能描述**

 使用线程池管理高频任务和实时数据获取。该模块能够提高系统的并发处理能力，确保实时数据的 及时获取和交易逻辑的高效执行。

**主要功能**

1. **线程池管理**

. 创建线程池，最大线程数为 2。通过合理设置线程池的大小，避免因线程过多导致的系统资源耗 尽。

 通过 ThreadPoolExecutor 提交高频任务和实时任务。线程池会自动分配线程执行任务，提高任 务的执行效率。

 代码实现：

|  |
| --- |
| def sell\_logic(self):  self.executor.submit(self.high\_frequency, self.codes\_list)  self.executor.submit(self.real\_time, self.codes\_list, 'latest') |

2. **事件控制**

 使用 threading.Event 控制任务的启动和停止。用户可以通过 GUI 界面控制任务的启动和停止， 方便灵活地管理系统的运行。

 提供任务状态监控功能。系统会实时监控任务的运行状态，若任务出现异常会及时给出提示信息。

3. **任务调度**

. 每 4 分钟获取一次前19个15分钟K线的数值。通过定时任务，系统会定期获取高频数据，确保数据 的及时性。

 实时获取行情数据，更新布林线。系统会实时监听行情数据的变化， 一旦有新的数据更新，会立即 更新布林线指标，为交易决策提供最新的依据。

**3.6 AI 实时分析模块**

**功能描述**

 对实时股票数据进行深度分析，利用深度学习模型预测股票价格的短期走势和波动幅度，为交易决策提供额外的参考依据。

**主要功能**

1**.数据准备**

 从数据处理模块获取经过清理和筛选的股票数据，包括历史价格、成交量等信息。

 对数据进行归一化处理，将不同范围的数据统一到相同的尺度，提高模型的训练效果。

2.**模型训练**

 使用历史股票数据对 LSTM 模型进行训练，调整模型的参数以优化预测性能。

 采用交叉验证等方法评估模型的准确性和稳定性，确保模型具有良好的泛化能力。

3.**实时预测**

 利用训练好的模型对实时股票数据进行预测，输出股票价格的短期走势和波动幅度。

 将预测结果反馈给交易逻辑模块和 GUI 模块，为交易决策和用户展示提供支持。

**4. 使用说明**

**4.1 系统运行**

1. 启动程序，加载 GUI 界面。双击程序图标或在命令行中运行程序，系统会自动加载 GUI 界面。

2. 输入股票代码和参数。在股票代码输入框中输入需要监控的股票代码，在其他输入框中设置相应的 参数，如令牌、延时时间、比例阈值等。

3. 点击按钮执行买入或卖出逻辑。根据自己的交易需求，点击相应的按钮执行买入或卖出操作。

4. 查看实时数据和布林线。在数据展示面板中查看股票的实时价格和布林线数据，了解股票的市场情 况。

**4.2 环境依赖**

1. Python 版本： 3.8+。建议使用最新的 Python 版本，以确保系统的稳定性和性能。

2. 必要库：

 numpy ：用于高效的数值计算。

 pandas ：用于数据处理和分析。

 requests ：用于网络请求。

 tkinter ：用于创建 GUI 界面。

 easytrader ：用于与同花顺客户端进行交互。

1. **特色功能**

1. **实时数据获取与处理**：系统能够实时获取股票市场的最新数据，并进行高效处理，确保用户能够及 时了解市场动态。

2. **自动化买入与卖出逻辑**：结合布林线和实时数据，系统能够自动判断交易条件，实现自动化买入和 卖出操作，减少人为因素的干扰。

3. **支持多线程并发任务**：使用多线程技术，系统能够同时处理多个任务，提高系统的并发处理能力和 响应速度。

4. **友好的图形用户界面**：采用直观、简洁的 GUI 界面，方便用户操作和管理，即使是初学者也能轻松 上手。

5. **数据可视化展示**：通过图表等方式直观地展示股票价格的走势和布林线的变化情况，帮助用户更好 地理解市场数据。

6. **交易策略定制**：用户可以根据自己的投资偏好和风险承受能力，灵活设置交易参数和策略，实现个 性化的交易需求。

**6. 未来改进方向**

1. **增加更多技术指标支持**：除了布林线指标，还将增加更多的技术分析指标，如 MACD、 KDJ 等，为 用户提供更全面的市场分析工具。

2. **优化多线程性能，减少资源占用**：进一步优化多线程管理模块，提高线程的执行效率，减少系统资 源的占用，确保系统在高并发情况下的稳定性。

3. **提供交易日志功能，记录用户操作**：增加交易日志功能，详细记录用户的交易操作和系统的运行情 况，方便用户进行交易回顾和分析。

4. **增加用户权限管理功能**：实现用户权限管理，不同权限的用户具有不同的操作权限，提高系统的安 全性和数据的保密性。

5. **支持更多交易平台**：除了同花顺客户端，还将支持其他主流的交易平台，为用户提供更多的选择。

6. **引入机器学习算法**：利用机器学习算法对历史数据进行分析和预测，为用户提供更精准的交易建议。

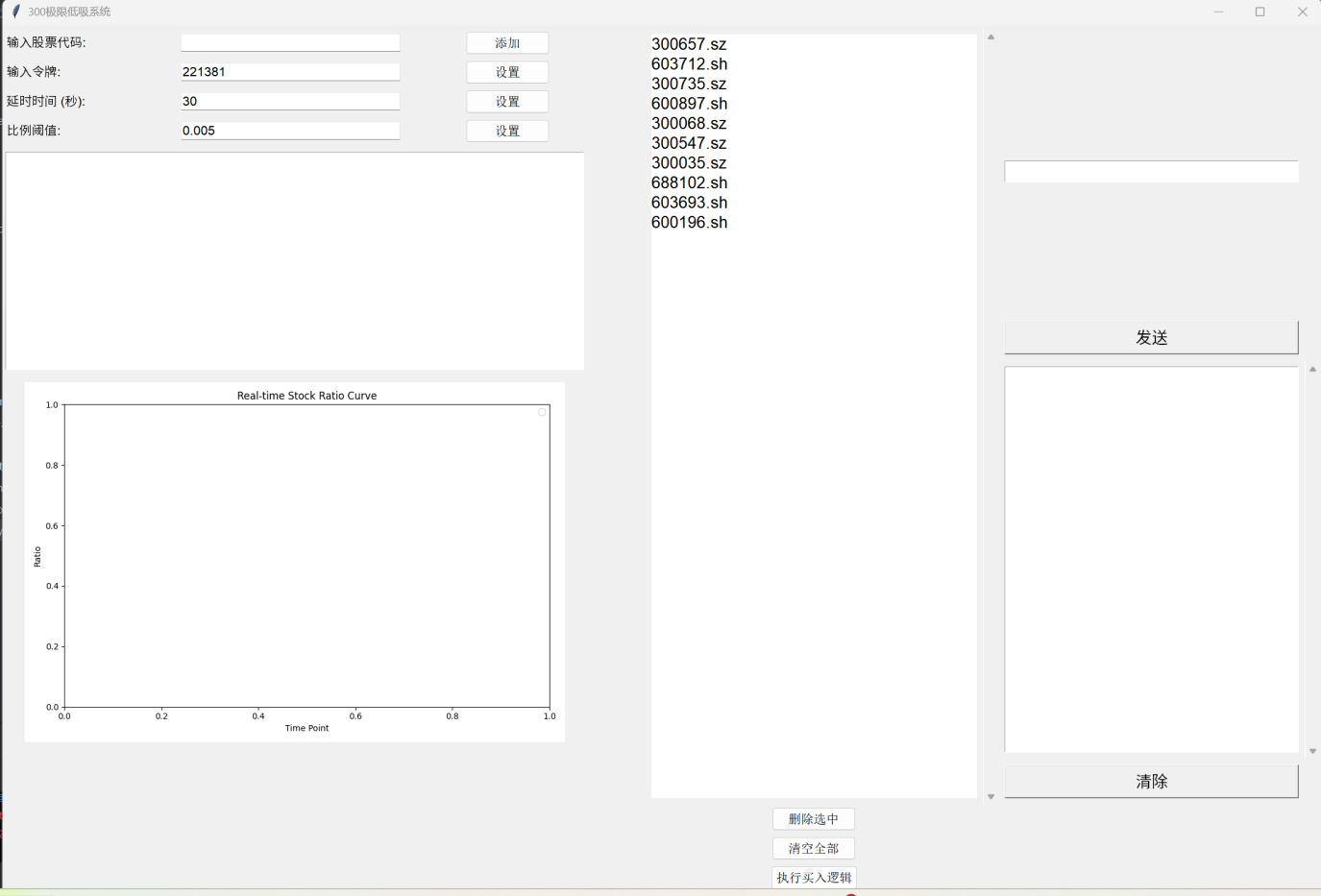
**7. 版权声明**

本系统为原创开发，版权归作者所有。未经授权，不得用于商业用途。任何单位或个人如需使用本系 统，必须获得作者的书面许可。

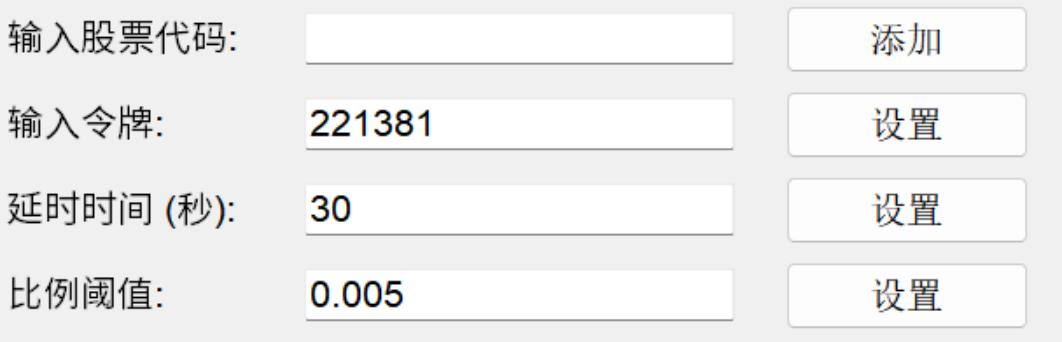
**8.1 系统运行截图**

1. **GUI 界面截图**：

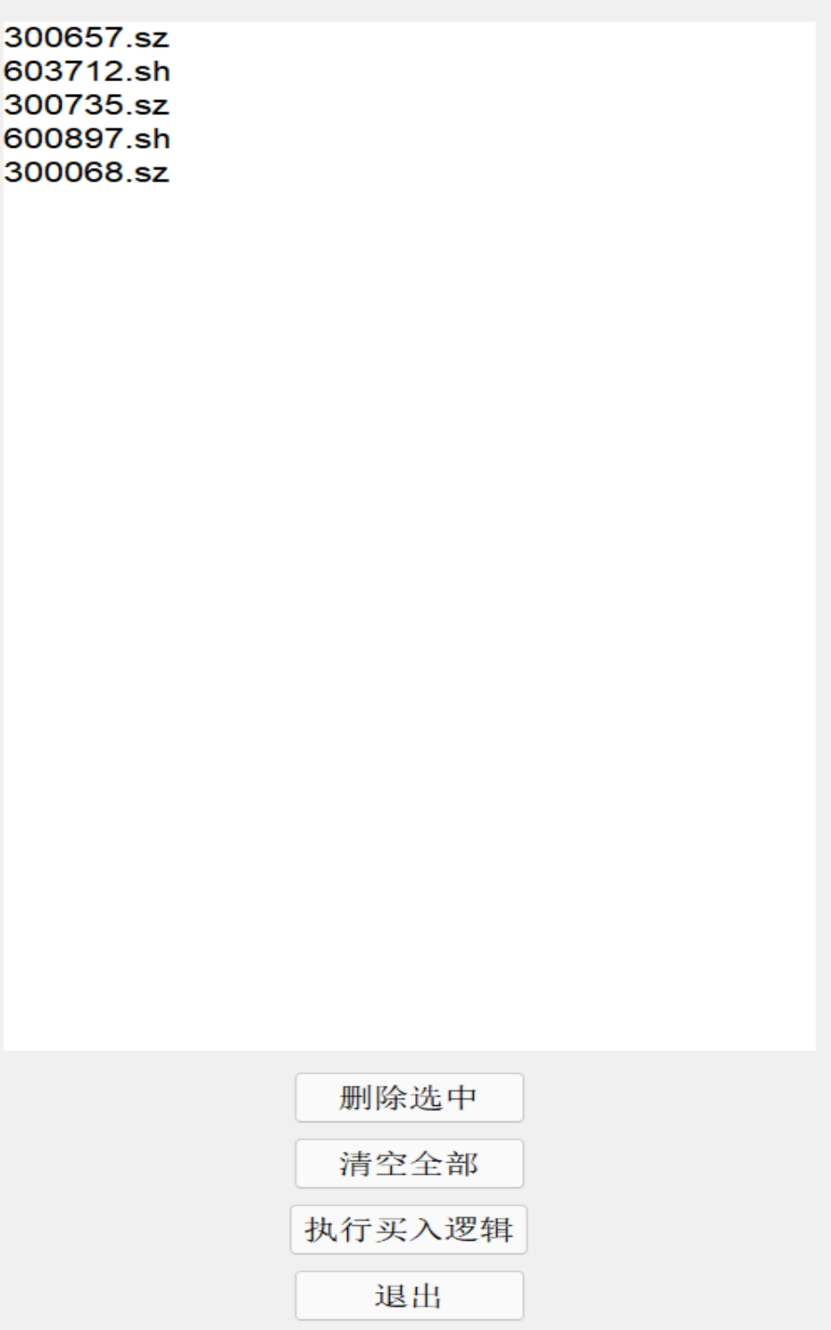
系统的图形完整用户界面：



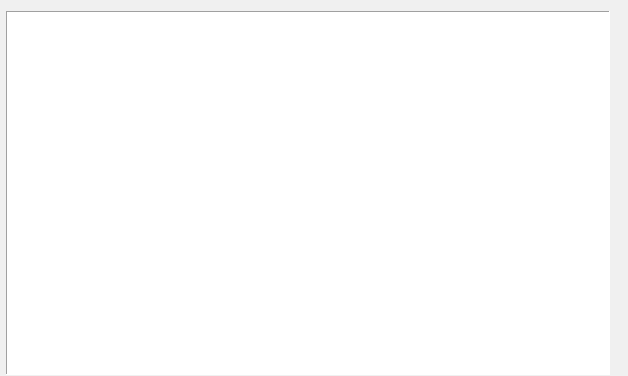
参数设置部分：



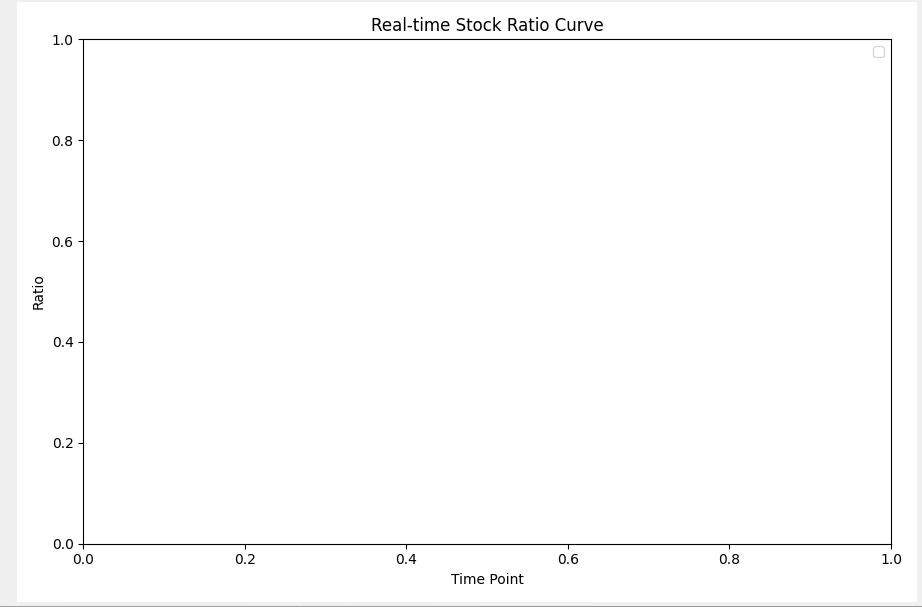
程序执行部分：



数据显示部分：



比例可视化部分：

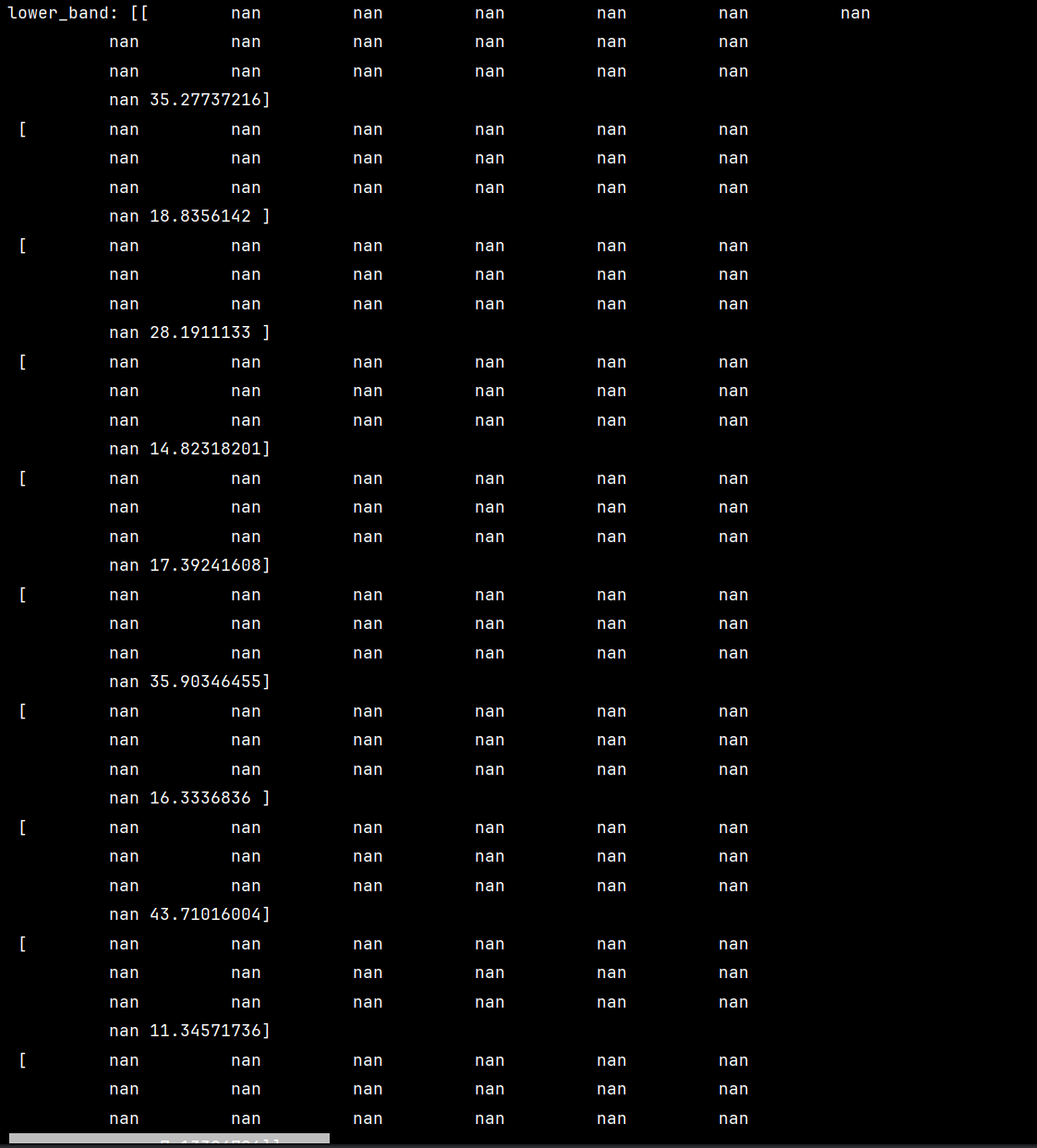


AI问答部分：



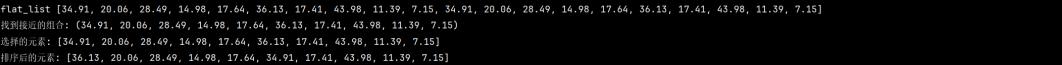
2.**数据处理结果截图：**

**下布林线计算展示：**

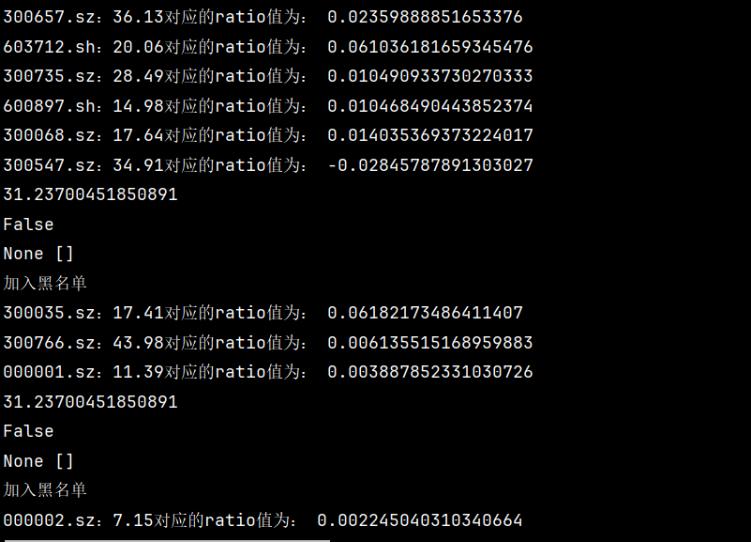
****实实时GUI界面展示：



实时价格选择算法展示：

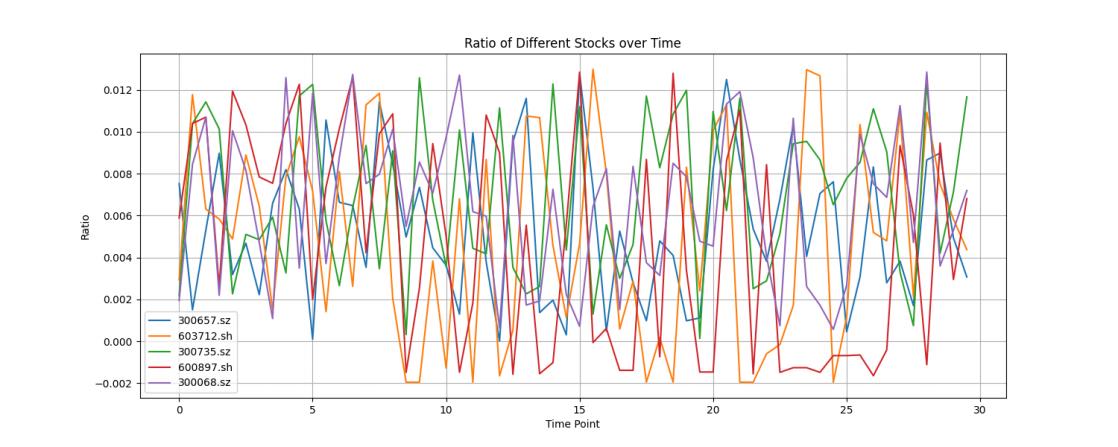


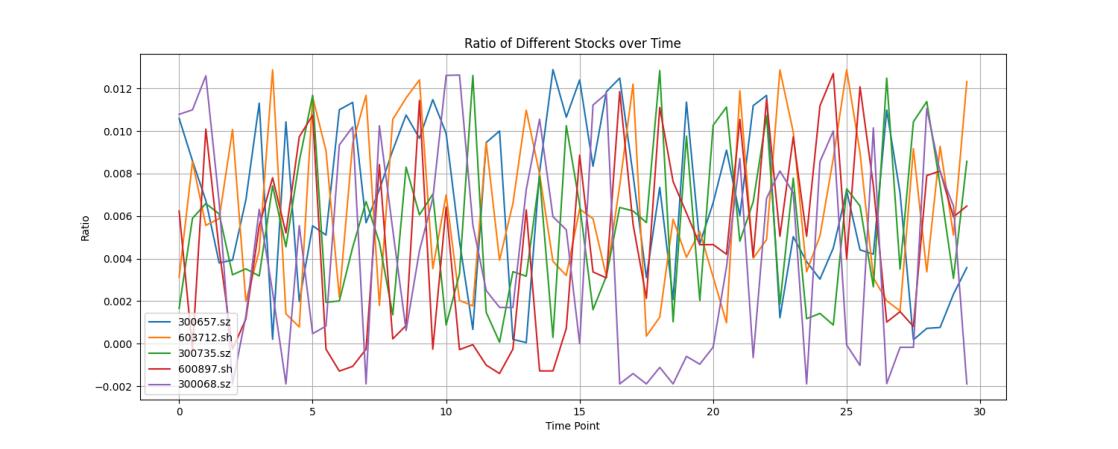
实时阈值获取展示：

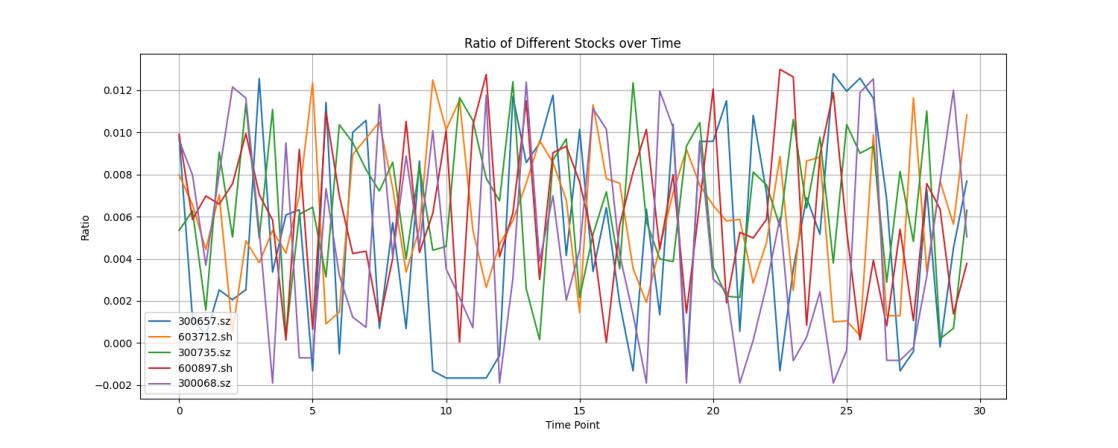


实时比例值可视化展示：

我们展示了三个时间点的Ratio值：

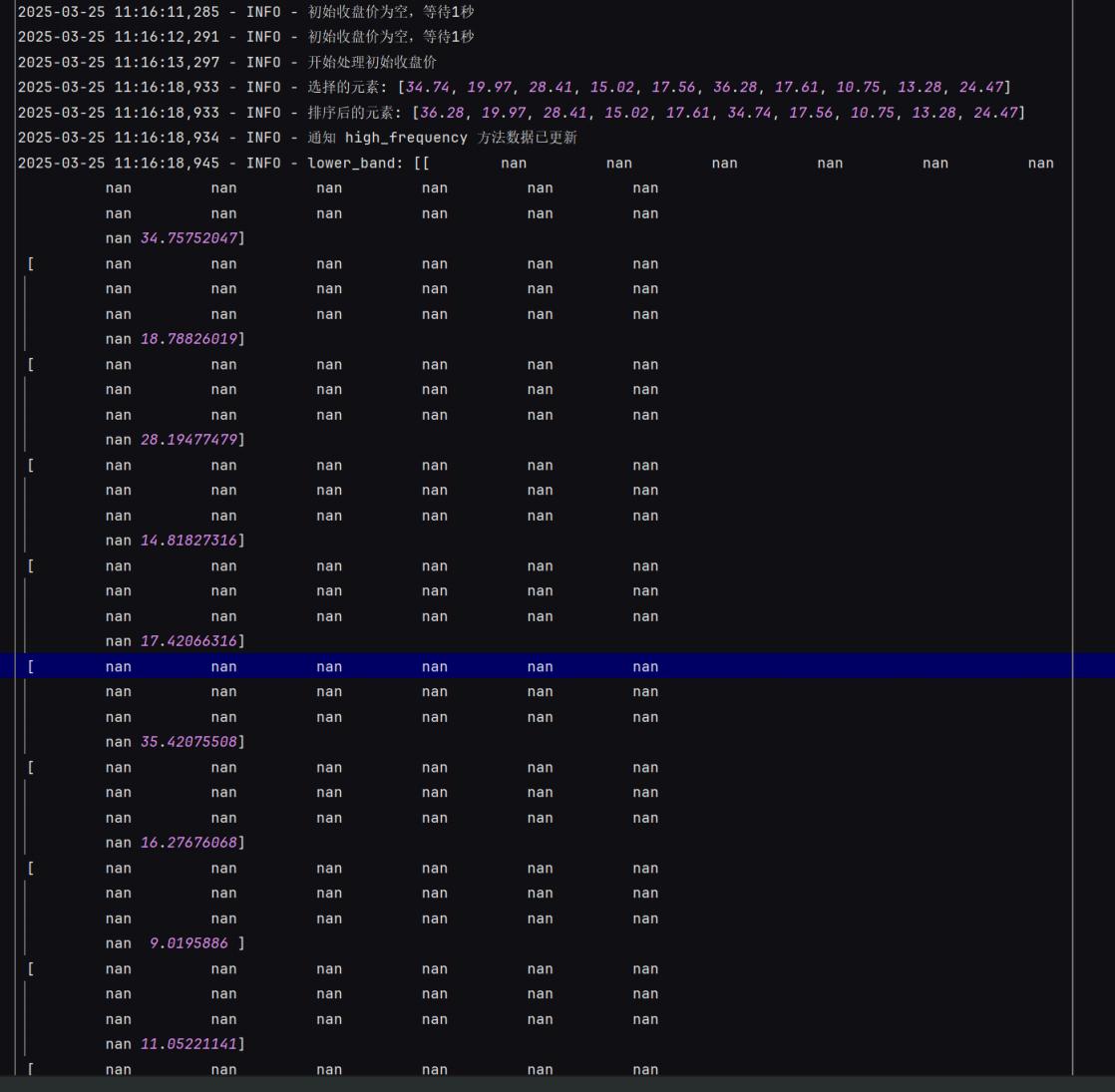


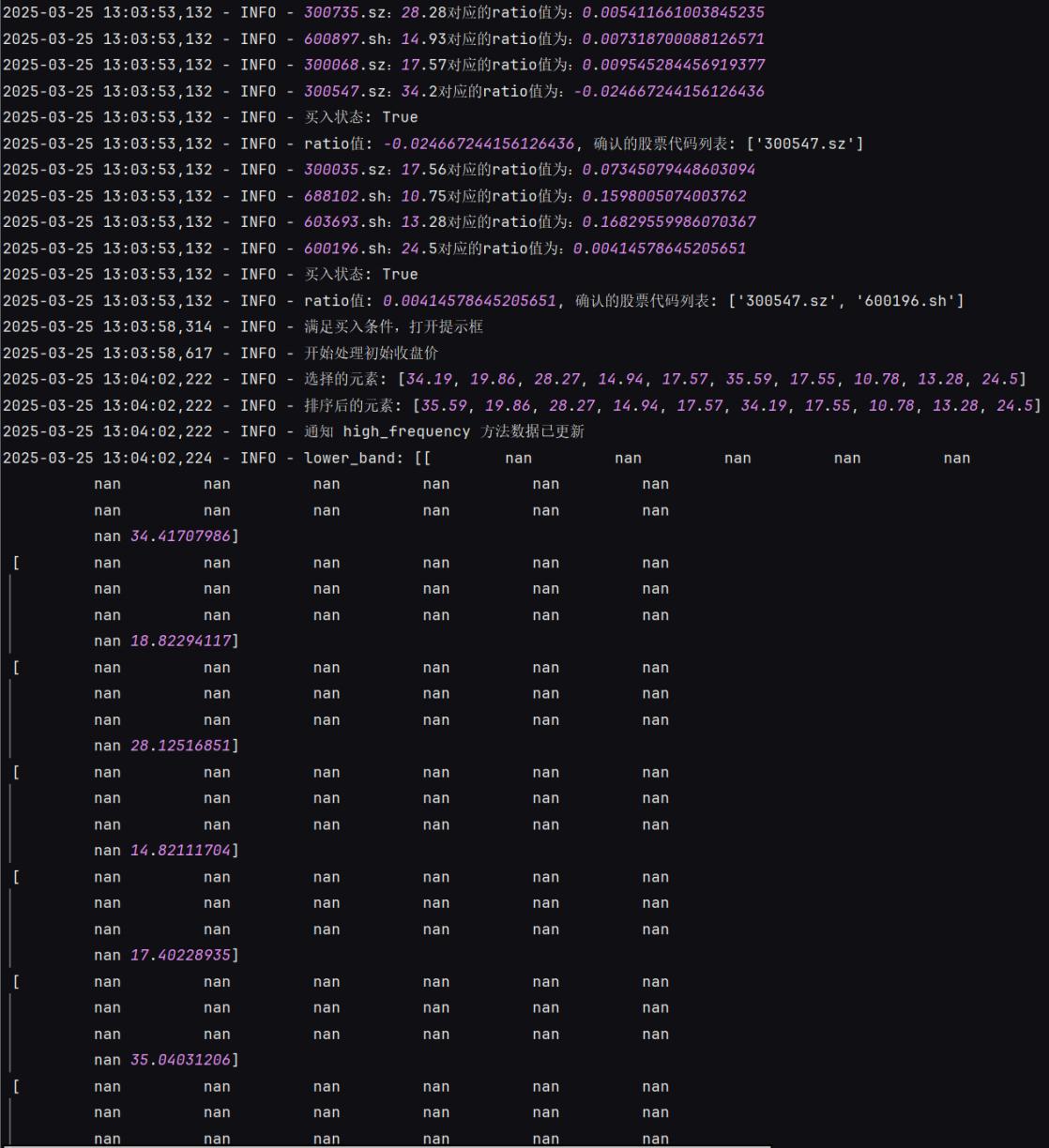




3.**实时日志保存结果**：

我们展示了2个时间点的实时日志：





5.实时AI问答效果展示:

