High-Frequency Alpha Strategy Leveraging Microstructure Insights with Zero-Exposure

---

Development Environment Structure:

The project is designed to ensure modularity, scalability, and efficiency, which are critical for high-frequency trading systems. It comprises the following core files:

```
project/
├── data_processing.py  # Handles data ingestion and preprocessing
├── alpha_factors.py    # Constructs and transforms alpha signals
├── backtest.py         # Implements the backtesting logic
├── pnl_metrics.py      # Computes performance metrics and analytics
└── notebook.ipynb      # Demonstration and analysis within a Jupyter notebook
```

Model Description:

a. Underlying Ideas:

This alpha strategy leverages **market microstructure insights** and **trading behaviors** derived from one-minute intraday data. At five-minute intervals, **11 alpha signals** are generated. These signals are subsequently transformed using a custom function, `opPower`, which is designed to ensure a **net exposure of zero**. The goal is to create a position model suitable for high-frequency environments where efficiency and low-latency signal processing are critical.

---

b. Implementation Details:

1. **Data Handling:** The data is sourced from CSV files, merged, and formatted to resemble the **MMEP (Market Microstructure Event Processing)** structure. It is indexed using `didx` (representing days) and `tidx` (representing minutes), forming a **multi-indexed time-series dataset**. This structure allows for **efficient access and processing** of intraday data, which is crucial in HFT environments. After processing, the data is cached locally to further optimize speed during analysis and backtesting.
2. **Alpha Model Generation:**
    - **Alpha Signal Construction:** Using one-minute data, 11 distinct alpha signals are computed every five minutes. These signals leverage **market microstructure insights** and **trading behaviors** and capture various **market dynamics**, including volume imbalances, price momentum, VWAP deviations, and bid-ask spreads. The alpha signals are as follows:
        - **Alpha1:** Frequency of active buy volume exceeding active sell volume over five minutes.

- **Alpha2:** VWAP imbalance between bid and ask over five minutes.
- **Alpha3:** Number of instances where capital inflow exceeds capital outflow within five minutes.
- **Alpha4:** Active buy volume surpassing the five-minute rolling average.
- **Alpha5:** Active sell volume below the five-minute rolling average.
- **Alpha6:** Number of trades exceeding the five-minute rolling average.
- **Alpha7:** Price momentum over five-minute intervals.
- **Alpha8:** Volume imbalance over five-minute intervals.
- **Alpha9:** Price deviation from VWAP.
- **Alpha10:** Bid-ask spread relative to VWAP.
- **Alpha11:** Spike in trade count.

These signals provide a comprehensive view of **market behavior**, covering both liquidity and momentum, which are essential in high-frequency trading.

3. **opPower Transformation:**

   This transformation is key in ensuring the **net neutrality** of the strategy:

   - **Step 1:** Alpha values are ranked and normalized into the range [−0.5,0.5][-0.5, 0.5][−0.5,0.5].
   - **Step 2:** Exponential scaling is applied to the absolute values while maintaining the original signs, thereby amplifying stronger signals.
   - **Step 3:** Positive and negative components are scaled separately so that their sums are exactly +0.5 and -0.5, respectively. This ensures **net exposure neutrality** while preserving the strength of signals.
4. **Backtesting Framework:**

   The backtest simulates a high-frequency trading strategy with an initial capital of **10 million**. It evaluates the performance of the strategy under **realistic market conditions**, including transaction costs specific to the Hong Kong Stock Exchange (e.g., stamp duty, SFC levy, and slippage).

   - **Position Assignment and Returns Calculation:** Every five minutes, positions are recalculated based on the transformed alpha signals. The position for the current five-minute interval is used to guide the trading for the next interval to avoid forward looking problem. Returns are computed using VWAP-based price changes between intervals, weighted by the previous position, to reflect the portfolio's exposure to price movements. This ensures that trades are executed at the start of the next five-minute window, based on the positions derived from the signals in the preceding interval.
   - **Transaction Costs:** Transaction costs incorporate **stamp duty, trading fees, SFC levy, and slippage** as per Hong Kong market conventions. This ensures a realistic estimate of trading costs, which are vital for determining profitability in high-frequency strategies.
   - **Tracking PnL:**

- **Capital Updates:** After calculating returns for each interval, capital is updated and transaction costs are deducted to compute the net profit or loss.
- **Turnover:** The turnover is tracked using the absolute changes in positions, providing insight into the frequency of trading and its associated costs.
- **Long/Short Exposure:** The average long and short exposure is monitored every five minutes to evaluate the directional bias of the strategy.

---

c. Performance Metrics:

The strategy computes both **daily** and **monthly performance metrics** based on the five-minute PnL data, which is aggregated over time. These metrics include:

- **Daily/Monthly Return:** The overall return, calculated as the percentage change in capital at the end of the period relative to the beginning.
- **Daily/Monthly Turnover:** A measure of trading activity, calculated as the sum of position changes weighted by the previous capital.
- **Daily/Monthly Maximum Drawdown:** The largest peak-to-trough loss in capital over the period, expressed as a percentage of the peak capital.
- **Monthly Sharpe Ratio:** A key measure of **risk-adjusted returns**, calculated by dividing the mean monthly return by the standard deviation of daily returns and annualized using a 20-trading-day assumption.

Files Containing Generated Target Positions:

All generated target positions are stored and can be accessed in the output file generated by the backtesting framework. A pnl file is saved to serve as a **detailed record** of the positions and PnLs generated during the simulation.