Understanding The

Pair Trading

Arjun Balakrishnan
2024DMF07

Problem Statement

- High frequency trading algorithims make the markets and has a very large impact on the price fluctuations.

- Large banks builds alogrithims to make profit out of the retail traders(zero-sum game).

- Retail traders like us how will we compete againsts trading bots who execute trades very fast without human intervention.

Proposed Solution

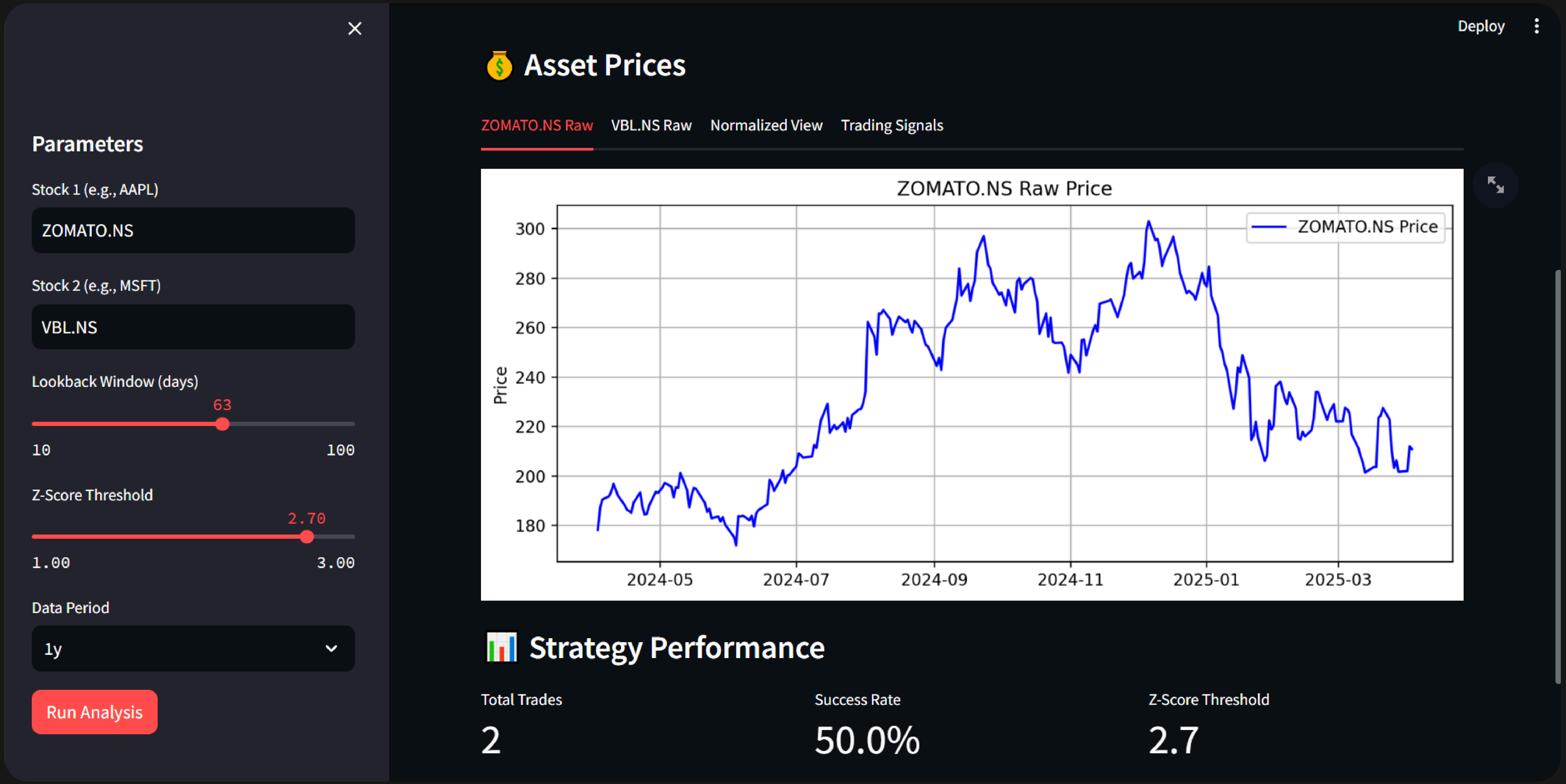
- **Level the Playing Field**: The algorithmic backtesting dashboard empowers retail traders by providing them with tools to test and refine their strategies, similar to institutional algorithms.

- **Automated Trade Signals**: Traders can input their logic and generate data-driven signals without manual analysis, helping them compete with high-frequency trading firms.

- **Cost-Efficient Strategy Testing**: Retailers can optimize their approaches risk-free through backtesting, reducing reliance on expensive proprietary systems used by big firms.

Overview

A high frequency trading strategy backtesting dashboard.



Parameters

Stock 1 (e.g., AAPL)

ZOMATO.NS

Stock 2 (e.g., MSFT)

VBL.NS

Lookback Window (days)

63

10 100

Z-Score Threshold

2.70

1.00 3.00

Data Period

1y

Run Analysis

💰 Asset Prices

ZOMATO.NS Raw | VBL.NS Raw | Normalized View | Trading Signals

📊 Strategy Performance

Total Trades

2

Success Rate

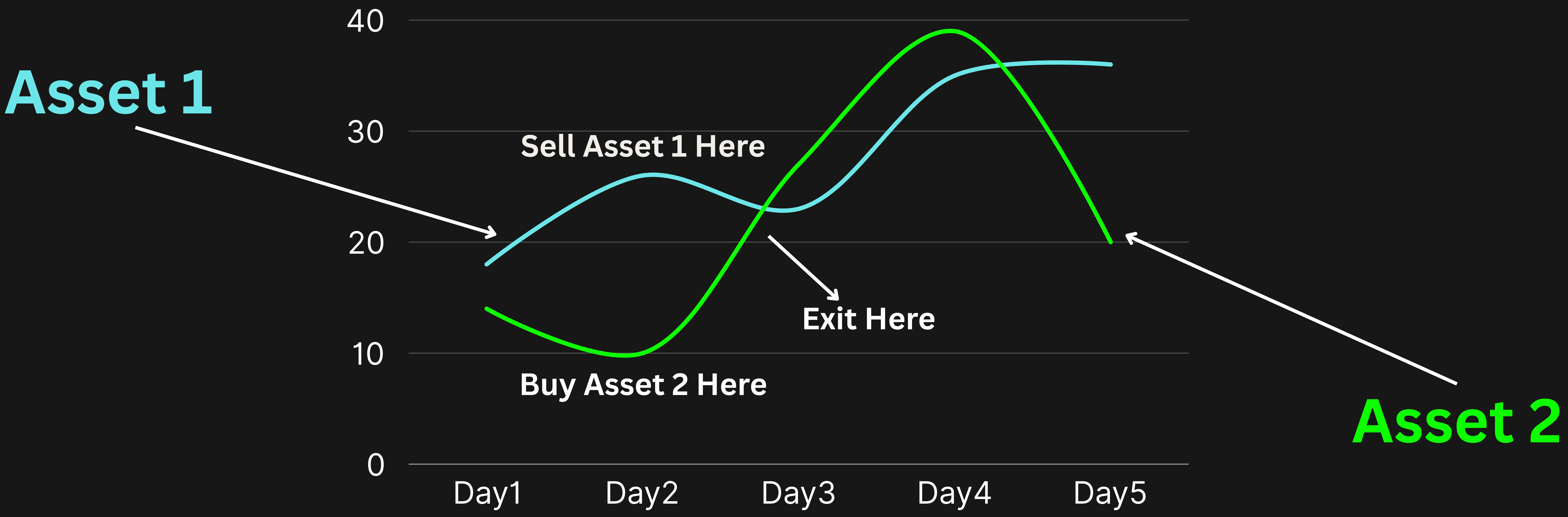
50.0%

Z-Score Threshold

2.7

Let's understand Pair Trading

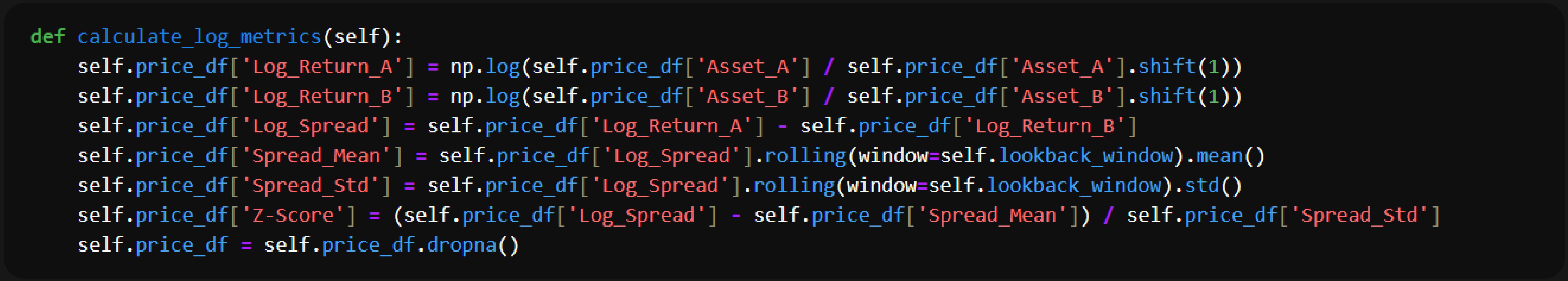
The act of buying and selling two correlated assets.



Example Chart:

Key Concepts

Z - Score Calculations



```python
def calculate_log_metrics(self):
    self.price_df['Log_Return_A'] = np.log(self.price_df['Asset_A'] / self.price_df['Asset_A'].shift(1))
    self.price_df['Log_Return_B'] = np.log(self.price_df['Asset_B'] / self.price_df['Asset_B'].shift(1))
    self.price_df['Log_Spread'] = self.price_df['Log_Return_A'] - self.price_df['Log_Return_B']
    self.price_df['Spread_Mean'] = self.price_df['Log_Spread'].rolling(window=self.lookback_window).mean()
    self.price_df['Spread_Std'] = self.price_df['Log_Spread'].rolling(window=self.lookback_window).std()
    self.price_df['Z-Score'] = (self.price_df['Log_Spread'] - self.price_df['Spread_Mean']) / self.price_df['Spread_Std']
    self.price_df = self.price_df.dropna()
```

Key Concepts

Rolling Lookback Window(Moving Average)

30 Day Lookback Period

Data Set 360 Days (1 Year)

	0 1 2 3 4 30	1 2 3 4 5 30	2 3 4 5 .... 31 32	335 336 337.... 364
Backtest:	31	32	33	365

It is always recent relative to each data point.

Every signal is genrated using only the information available at historical moment.

Uniquness

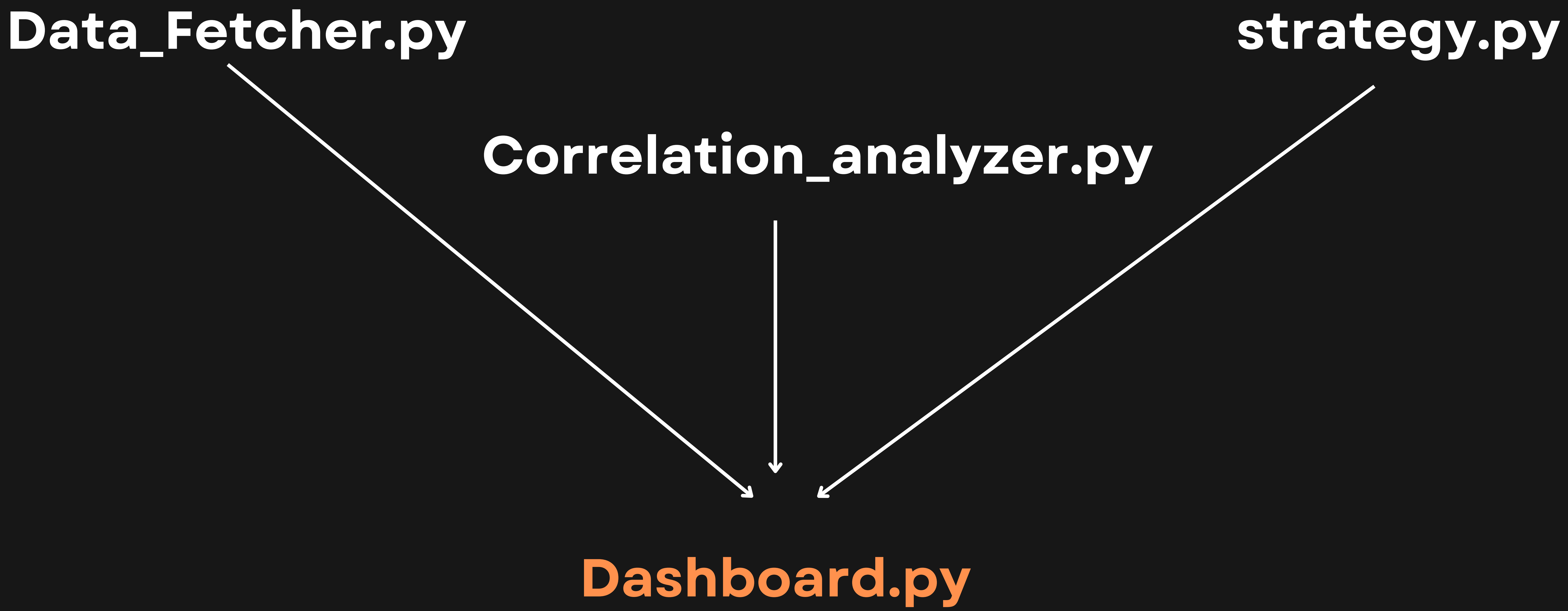
- It is unique as the strategy involves my own inputs and it is customisable according to the user inputs.

- There are other alogrithim building websites which too complex.

- In my dashboard we even show the pair taken is good to trade or bad and even tells the history of the trades according to the strategy.

✅ Simpler than other sites. ✅ Actionable insights (not just raw data).

Code Architecture



Data_Fetcher.py

Correlation_analyzer.py

strategy.py

Dashboard.py