TRADING HACKATHON CHALLENGE PROPOSAL

J.P. Morgan AI Research
email@jpmchase.com

ABSTRACT

This hackathon aims to let participants build an equity trading algorithm that maximizes profit by buying and selling stocks, like managing an investment portfolio. Participants will be provided access to a certain number of years of historical data for a group of up to 500 stocks. Participants' algorithms should review the daily price data and generate orders to buy or sell stocks.

1 About The Data

The data for this hackathon is a synthetic version of the S&P 500, created by AI Research's Synthetic Data group. It is intended to be a synthetic market version that can be used for testing trading algorithms. It is modeled after real data, with accurate similarities and correlations between groups of stocks. The data traders use to inform their trading is classified as "financial data" and "market data." Financial data includes information about a company's financial health, industry, and macroeconomic factors. Market data focuses on historical price and volume information. Traders analyze these data to make informed decisions about buying and selling stocks.

1.1 Market Data

The market data for this project is provided as a comma-separated-values (CSV) file, where each row represents the data for one day. An example looks like this:

Date	AAPL	IBM	JPM	TSLA
2005-01-03	52.2773	86.8520	60.4727	2.6500
2005-01-04	51.7295	85.7908	60.4137	2.6015
2005-01-05	52.0064	85.1988	60.3900	2.5867
2005-01-06	52.0417	85.6319	60.4313	2.5740
2005-01-07	52.1596	85.5092	60.4137	2.5570
2005-01-10	52.2420	85.9135	60.3841	2.5660
2005-01-11	52.5484	85.3215	60.4668	2.5888
2005-01-12	52.6191	85.6031	60.5553	2.5810

Table 1: A market data example with five tickers

The column headers indicate the ticker or symbol for each stock, as shown in Table 1. Each row starts with a time stamp (date) and then prices for each stock for that day. These prices represent closing prices for the stocks on that day. The executed prices for all trading orders are defined as the closing prices on that day.

1.2 Financial Data

The financial data used by traders for investing generally falls into four categories: Fundamental, Technical, Sentiment, and Macroeconomic:

- **Fundamental Analysis:** This approach involves evaluating a company's financial health, competitive position, and overall economic environment.
- **Technical Analysis:** This approach involves analyzing historical price and volume data to identify patterns and trends suggesting future price movements.

- **Market Sentiment:** Stock traders analyze market sentiment data, such as investor surveys, put-call ratios, and short interest, to gauge investor sentiment and identify potential contrarian trading opportunities. News feeds may also provide sentiment information.
- **Macroeconomic Indicators:** Stock traders consider the overall economic environment by analyzing GDP growth, inflation, interest rates, and employment figures to identify how these factors may impact a company's performance.

For instance, when developing trading strategies at a hedge fund, all this incoming data is analyzed, condensed, and summarized into daily values for each stock. These numbers are called "features," "indicators," or "signals." Then, all this data is provided to the trading algorithm for determining trades. To simplify the project, we will precompute the values for five indicators and provide them to participants for each stock in a CSV file named for that stock. For instance, the file providing information about IBM is named IBM.CSV, and it looks like this:

Date	indicator01	indicator02	indicator03	indicator04	indicator05
2005-01-03	0.2773	1.8520	0.4727	1.6500	0.2773
2005-01-04	0.7295	1.7908	0.4137	0.6015	1.7908
2005-01-05	0.0064	1.1988	0.3900	0.5867	0.3900
2005-01-06	0.0417	-2.6319	-0.4313	-0.5740	-2.6319
2005-01-07	-0.1596	1.5092	0.4137	-1.5570	-0.1596
2005-01-10	1.2420	1.9135	0.3841	0.5660	1.9135
2005-01-11	0.5484	1.3215	-0.4668	1.5888	-0.4668
2005-01-12	0.6191	1.6031	0.5553	2.5810	1.6031

Table 2: A financial data example with five indicators

The values for the indicators are always numbers, and they may be positive or negative. We're not providing more information for the participants about these indicators or their meaning except that if the participants' algorithms are good, they can extract actionable information and make a certain profit.

2 The Task For Participants

Participants' trading algorithms should operate in the following way and follow these rules:

- The program should read over 500 files, as follows:
  - The market data – one file.
  - the financial data – about 500 files.
  - A strategy file that describes how the participants' algorithm should behave. The format and structure of this file can take any form.
- Assume the participants start with $\$1,000,000$ in cash.
- Participants' program should output a trades file, named "trades.csv", that lists the trades it would like to make for each day.

The "trades.csv" file should be formatted like this:

Date	AAPL	IBM	JPM	TSLA
2005-01-03	0	0	0	0
2005-01-04	100	0	0	0
2005-01-05	0	0	0	-100
2005-01-06	0	0	0	0
2005-01-07	0	0	0	0
2005-01-10	-100	0	0	0
2005-01-11	0	0	0	0
2005-01-12	0	0	0	100

Table 3: A "trades.csv" file format example

The numbers on each row for each stock indicate how many shares the participants would like to trade in that stock on that day. Participants will get the closing price for the stock on that day. A "0" indicates no trading on that day, positive numbers indicate a "BUY" order with that number of shares, and negative numbers indicate a "SELL" order.

3 Rules And Evaluation

We will provide a market simulator to read participants' trade and market data files and simulate the trades with our testing algorithm. Some metrics for evaluating the trading performance will be provided. We will evaluate each algorithm according to its **Sharpe ratio** and **total return**. We will also provide awards and rankings on each of those metrics. We also set certain rules for participants' trading algorithms:

- Portfolio leverage should not exceed 1.0. For instance, at the beginning of the simulation, participants can only buy $1M in stocks with the $1M cash. Any requested trades that lead to a violation of the leverage restriction will be ignored.
- Shorting is allowed, which means participants can "sell" stocks before buying them.

We use the following equation to determine how leveraged it is. In general, using leverage means a participant borrows money to increase the size of the portfolio:

$$L = (lv + abs(sv))/(lv + cash - abs(sv)) \tag{1}$$

- L is portfolio leverage.
- lv is the total value of all the long positions.
- sv is the total value of all the short positions.
- $cash$ is the value of cash in the portfolio.

Examples:

- If a participant starts with $100 and purchases $100 of stock, the leverage is 1.0.
- If a participant starts with $100 and purchases $100 of stock and shorts $100 of stock, the leverage is 2.0.

To simplify this problem, we have made the following assumptions that may not hold in the real world:

- No transaction costs – we don't charge a trading fee.
- No market impact – participants' trades don't affect the rest of the market.
- Participants can observe the market close price and execute on it.

4 Hackathon Files And Procedure

Participants will be provided with 20 folders of data. The folders are in two groups: "training" and "testing", e.g., training01 to training10 and testing01 to testing10. Participants can use the data in the training folders to train or develop trading algorithms. Each folder will contain about 501 files (1 market and 500 financial data). The data in each folder covers the same period, from January 2022 to December 2022.

Participants' algorithms must follow these rules:

- No peeking into the future – the algorithm should step through the data daily and make trading decisions only on the data up to and including that day. It should not, for instance, look ahead to a future price to make a trade in anticipation of that future price.
- If the trading algorithm is learning or adaptive, the participant should reset it before running it on each test set. For instance, the algorithm should not learn from data in testing01 in anticipation of running on data in testing02.
- Deterministic – given the same data, the algorithm should always output the same trades. Participants can use a random number generator, which should be initialized with the same seed each time.
- Use of data – participants should NOT allow their trading algorithms to look at the data in the testing folders until the trading strategies or algorithms have been finalized.
- If participants choose to revise or improve the trading algorithm, for instance, by learning or retraining, they should ONLY use the training data for that purpose. Participants should never train on the testing data.
- Participants should run the same trading algorithm on all the testing data. Don't optimize trading algorithms for each folder.