1 Five Rings Capital DayOf Case Instructions

You will be given three stocks to model of varying difficulty: simple, medium, and hard. For each stock, you will be given a training data set to model. You'll use your model to write an allocation function, and you'll be scored based on the performance of your allocation function. We'll give you data for the following stocks for 10,000 timesteps. These stocks are labeled simple, medium, and hard, in rough order of intended difficulty to model.

1.1 Simple

You are a young investor looking to invest in the steel market, thanks to the recently imposed tariffs on foreign steel. You are looking to buy stocks from an up and coming steel company known as Real Steel. You know that Real Steel's stock prices are heavily dependent on two other factors in the stock market. One factor is the net gain in Luminous Aluminum's stock prices (represented by the variable x_1), as the aluminum industry is a big competitor to the steel industry. The other factor is the net gain in Royal Oil's stock prices (represented by the variable x_2), as Royal Oil is the main oil supplier for Real Steel.

To recap, the stock's price p_i at time i is dependent upon the following factors:

- The stock's price at the previous timestep p_{i-1}
- Two variables x_1 and x_2

You should probably try modeling linear returns $(p_i - p_{i-1})$ to get a sense of how the price changes according to values of the other variables. You might find that some type of simple polynomial model works quite nicely.

1.2 Medium

You are also looking to invest in the healthcare industry, namely the company Brontosaurus Biotechnology. You've decided to take a detailed look on how its stocks are traded to predict the future prices of the stock. Let x_1 denote the net change in the average spread of the stock, x_2 denote the net change in the quantity of bids and asks executed, and x_3 denote the net change in the fraction of orders that are limit orders.

Thus, the stock's price p_i at time i is dependent upon the following factors:

- The stock's price at the previous timestep p_{i-1}
- Three variables x_1, x_2, and x_3

Modeling log returns $(\log p_i - \log p_{i-1})$ might be a good idea for this one.

1.3 Hard

You are now looking into the telecommunications industry, specifically the company Cumulus Cloud Computing. You're now going to model the future prices of the stock by looking at the volatility of related symbols. The variable x_1 is based on the volatility of other stocks in the Cloud Computing industry, while the variable x_2 is based on the volatility of other stocks in the entire technology sector. x_3 measures the change in the beta coefficient of the company, i.e. the volatility of the company's stocks relative to the whole market.

That is, the stock's price p_i at time i is dependent upon the following factors:

- The stock's price at the previous 50 timesteps $p_{i-50}, \ldots, p_{i-1}$

- Three variables x_1, x_2, and x_3

Modeling log returns ($\log p_i - \log p_{i-1}$) might be a good idea for this one.

1.4 Allocation

You will now use your models to trade simple, medium, and hard stocks on a stock market. At each timestep, you will be able to spend at most $100M on the three stocks, and you much choose how to divide this money between these three stocks (or if to spend it at all). You cannot try to buy a negative quantity, since short positions are not allowed in this market. If you buy N stocks of a given type, and the stock changes from the price P to $P+\Delta P$ at that timestep, then your PNL for that stock on that timestep will be

$$f(N)\Delta P$$

with

$$f(N) = \left(\frac{2}{\alpha}\left(\sqrt{\alpha N + 1} - 1\right)\right)$$

for the absolute constant $\alpha = 0.0002$. This result applies to each stock individually. Note that this formula models market impact, with your marginal PNL decreasing in magnitude with higher quantity. Your net PNL at that timestep is the sum of your timestep PNLS over all three stocks. Your position will be cleared at the end of every timestep for the purpose of calculating PNL—**positions in stocks will not carry over between timesteps/testcases**.

For example, if for a particular timestep the stocks have respective prices p_1, p_2, p_3, and you allocate quantities q_1, q_2, q_3, your quantities need to be nonnegative and satisfy $q_1 p_1 + q_2 p_2 + q_3 p_3 \leq 100,000,000$. If the differences next prices minus current prices are given by $\Delta(p_1), \Delta(p_2), \Delta(p_3)$, your PNL will be computed for that timestep as

$$f(q_1)\Delta(p_1) + f(q_2)\Delta(p_2) + f(q_3)\Delta(p_3)$$

We'll provide a submit dataset that gives you the relevant features for 10,000 test cases, and you'll output allocations for these 10,000 test cases. Your score will be the sum of your PNLs over all these 10,000 cases.

1.5 Details

Starter code is provided in file `mymodel.py`, which will show you how to load the data. For utility, we've provided a file to check your model's mean squared error (on prices) with `python test_model.py`, but you should feel free to create your own error metrics (especially if you're predicting, say, log returns). Note in particular that `test_model.py` runs in-sample on the training data.

You should implement the `allocate` method stub in `mymodel.py` and then test it to ensure it works by running `python test_allocate.py`. In addition to testing whether your code works, the output of `test_allocate.py` will score your performance on 1000 out-of-sample test cases from `test_data`. When you're ready to output your submission csv, you can do so with `python run_allocate.py`, which will run your `allocate` function over 10,000 cases from `submit_data` and create a `output.csv` file ready for submission.

In the allocation function, we give you the current price as inputs (or the past 50 prices up to and including the current price for hard), and you decide how to allocate your money based on what you believe will be the price one timestep in the future.

You want to make sure you're running our code on `python3` to make sure everything works correctly—otherwise, there could be issues since we did not test our code on `python2`. Note that depending on the complexity of your code/model, creating your output csv with `run_allocate.py` could take 1-2 minutes to run.

1.6 Final Submission

Whatever your final `output.csv` file is, copy it to the home directory of your AWS instance before the end of the case. You can do so on command line by running `scp output.csv ec2-user@[my instance ip]:~/output.csv` from the local directory your `output.csv` is stored in.

1.7 Scoring

Each of the test cases involves running your allocation given the appropriate data for a given time step. Your final score will be your average PNL over all 10000 test-cases.

1.8 Data Details

Data specification is provided from completion, but our provided code should abstract away most of the process of loading and saving data. If you're more comfortable working in a language other than python and you're fine spending time figuring out the appropriate data formats for input and output rather than using our provided code, this section might be more relevant. Otherwise, you can (probably) safely ignore what follows.

Data is provided in three folders: train_data, test_data, submit_data. Each of these folders has three files: `simple.csv`, `medium.csv`, `hard.csv`. You may use the train and test data however you like; our template code uses train data to train your model and test data to test your model. Submit data includes the 10,000 timesteps of data that `run_allocate.py` will test your solution on.

1.8.1 Simple

The train file (10,000 data points) will contain the following columns: timestep (t), x_1, x_2, previous price (p_{i-1}), current price (p_i). The data will be provided in chronological timesteps, meaning the current price for one timestep will be the previous price for the next timestep.

The test file (1,000 data points) will contain the same columns: timestep (t), x_1, x_2, previous price (p_{i-1}), current price (p_i). Each timestep will be independent.

The submit file (10,000 data points) will contain the following columns: timestep (t), x_1, x_2, previous price (p_{i-1}).

1.8.2 Medium

The train file (10,000 data points) will contain the following columns: timestep (t), x_1, x_2, x_3, previous price (p_{i-1}), current price (p_i). The data will be provided in chronological timesteps, meaning the current price for one timestep will be the previous price for the next timestep.

The test file (1,000 data points) will contain the same columns: timestep (t), x_1, x_2, x_3, previous price (p_{i-1}), current price (p_i). Each timestep will be independent.

The submit file (10,000 data points) will contain the following columns: timestep (t), x_1, x_2, x_3, previous price (p_{i-1}). Each timestep will be independent.

1.8.3 Hard

The train file (10,000 data points) will contain the following columns: timestep (t), x_1, x_2, x_3, previous price (p_{i-1}), current price (p_i). The data will be provided in chronological timesteps, meaning the current price for one timestep will be the previous price for the next timestep.

The test file (1,000 data points) will contain the following columns: the following columns: timestep (t), x_1, x_2, x_3, price history, current price (p_i). Price history is a string of the previous 50 prices, separated by spaces. The prices are in the order $p_{i-50}, p_{i-49}, \cdots, p_{i-1}$. Each timestep will be independent.

The submit file (10,000 data points) will contain the following columns: timestep (t), x_1, x_2, x_3, previous price (p_{i-1}). Each timestep will be independent.

1.8.4 Output Csv

The output csv should have header a_1, a_2, a_3 and contain a following sequence of 10000 rows of allocations, where the ith of these rows reads N_s, N_m, N_h, with N_s the quantity of simple you wish to buy for the ith test case, and likewise for N_m and N_h.