

!!Weekly Fieldwork Logs Readme!!

I completed a technical virtual internship with a financial tech company based in London working with a team member Faiza based out of Pakistan. Our boss Emauell sponsored our research into Quantitative Finance that needs capital investments. This log is more of a technical log of goals, tasks, and deliverables for the week.

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
8:00am			Weekly Meeting 1-Hours				
9:00am		Start Work		Start Work		Start Work	
1:00pm						End Work 4-Hours	
4:00pm				End Work 7-Hours			
5:00pm		End Work 8-Hours					Total 20 Hours

With much of my work being remote and solo there were not many interactions besides a weekly 30min-1hour meeting with Faiza and Emmauel during the London workday. This meant that my meeting was at 4:00pm London time or 8:00am MST.

* Prompts:

1. Concise description of what was accomplished in the fieldwork experience that week
2. Explanation of any challenges and successes during the week
3. Plans for improving your performance
4. Description of the other experiences outside of the fieldwork placement including information about the living situation, meals, transportation, excursions, cultural insights, travel information, host family, organization/company support, or any other relevant experiences (*Virtual Interns can skip this portion)
5. Description of cultural differences from host country and US culture/home culture.



****This first weeks log does not much sense****

I skipped over an email with a different invitation to a 2nd GitHub repository and only had looked at half of the project. The miscommunication was sorted out just before the meeting and the 2nd week's log is much less panicked and legible.

6/8/2025

During my first week Monday June 9th, 2025 I was tasked to summarize the current long-term investment pipeline. I needed to go over the distinct stages of data collection, processing, machine learning modeling, prediction, stock ranking, portfolio construction, and backtesting. This Python-based investment bot looks at FTSE Small-Cap stocks using XGBoost to predict each stock's potential returns and then applies and backtests it. This exploration took about 20 hours with a miscommunication about where the code was located leading to another 5 hours needed.

The data fetching and preprocessing for this program resemble the collection from our short-term prediction while having a few key differences in collecting 215 FTSE Small-Cap historical fundamentals and tickers. This is all via the same EOD Historical Data API, and data is stored in the same CSV files, but other predictive data is collected too. We get 5 file outputs by collecting net_income.csv, cash_flow.csv, balance_sheet.csv, historical_prices.csv, and shares.csv. In our next step, we calculate ratios and returns and merge files together. Our 5 files are enriched with new columns. In the end, for our ML model, we have dividends paid, historical prices (high, low, and adjusted close), shares (shares in millions), net income, and total stockholder equity (book value of equity). We end up with a merged_with_returns.csv with EPS, PE, P/B Ratio, and such.

To train the model, we set up the prediction fundamentals and feed XGBoost. We prepare data for the supervised learning by making test & train splits; the trainer creates lag features (PE_lag1..lag4, Dividend_yield_lag1..lag4) and a rolling average for each target. We then do this for returns as well. This gives us the training data set with our predictive features specified. For each ticker and target variable, an XGBoost model is trained using time-series split validation. The trainer has hyperparameter tuning with grid search and cross-validation. After training the model, its metadata and a summary of training results (MSE and R² on the test set for each model) are saved. This results in a collection of per-stock models (e.g., one model for predicting PE, one for dividend yield, and one for returns per stock).

To summarize the prediction we do two things at the same time we predict the P/E and we predict the future price. We do this in a loop to predict the next 4 quarters, predict, shift lags, predict again. Our output becomes input for the next step. We then take the average of 4 quarters to get annual predictions then predict next year's price. The end result is a file containing each ticker's projected quarter-end prices for the next year, along with expected returns and dates. These predictions then come together to rank the stocks.



This next program, [ranking.py](#), ranks stocks based on the prediction to figure out an investment portfolio. We compare the actual vs. predicted by calculating the recovery years (lower is better) from many other values. We then make a score for each stock weighed by values (dividend yield score, PE score, and recovery years score) that produce the final score.

To then make an equally weighted portfolio of the stocks to best invest in, we check the scores. With a low, medium, or high value and a beta weight, we select stocks. We can use either the equal or beta-weighted model, and they calculate the amount of stocks and which stocks to buy. The file produces a file with tickers, company names, number of shares, purchase price, and total investment per stock.

We then make a backtest to test our models and save results and comparisons. We do this by getting our constructed portfolio and, for each day, calculating the total portfolio value increase or decrease. This is also done with the S&P 500 to benchmark our model.

Some ideas for enhancements. Refine the pipeline to user-friendly architecture and to continuously run. Getting ESG (environmental, social, and governance) metrics for the FTSE small-cap companies to rank sustainability.

6/15/2024

The week of June 16th, 2025, for my internship at Brickfort I enhanced the short-term trading strategy by optimizing code, including the ability to trade more than just the first highest estimated yield stock and using ABV and ATR to find shares, as well as updating debug testing and analyzing features. This week's deliverable includes a slide showing a mixed media format with code, text, and visuals; a GitHub commit; and this paper which all in total took about 19 hours to complete.

To explain what I accomplished this week with our project, I wanted to start with building a main testing file in a Jupyter notebook. I took all classes and imported them to a main function where I run all main functions concurrently, having all programs running sequentially with backtesting results printed and visualized in an equity graph for each bot (def record_equity). This format helped me run tests and tweak variables in the code to see changes across files in a repeatable way.

While constructing this testing framework, I ran through a few ways to improve our equity curves. The previous method included a simple ranking system based on the highest expected profit for a stock. I wanted to include the ability to buy more than one stock in a single day because being able to trade a lot when markets are volatile is vital to this program due to its high yields in highly volatile markets. I also wanted to use the ATR and volume functions already made to calculate shares.

My first attempt at implementation was a direct commit to the py file, and without a solid way to test my changes meaning it wasn't working. So I made the testing file. I also tried



including a different shares calculator, making an ATR-based position size calculator, but this showed no improvement. The final reviewed and tested code is simpler than I would like, but to be able to test how the amount of stocks traded per day affected total bot equity. There are no provable or quantifiable improvements to the equity, but I believe that with a grid search of optimized value of viable trade cutoffs to the max amount of trades per day will improve that. This is all contained in the `run_backtest` function.

In a message to me, Faiza asked me, “Can you work on picking up more tickets in one day?” So I implemented a very small function that asks for EODHB for a list of tickers instead of making a list that we pass to it. This gives us much more data, and a fine-tune needs to be done to make our bots trade with the new environment. Our trades decrease when these coins are added, but that is expected without hand-tuning.

While implementing the code, I fixed a few different bugs while also increasing our debug abilities. Our first one was: The trades with some trade didn't have an `exit_price`, making up a value. I haven't solved the problem but more got rid of it. I also added a debug feature to test the legitimacy of the trades called `debug_trades`. This records numerous things from each trade to cross-reference the trade with real life. This was able to lead me to an understanding that using a paper trade might fix some of our issues because of slippage, real commission, and other external features. This is not a code bug fix, but I think it will fix the returns being extraordinarily high. The improvement to the code's testability with the `test.ipynb` file is very large, plus the debug and equity graphs help out.

6/21/2025

For my 3rd week, June 23rd, 2025, for my internship at Brickfort, I worked on preparations to integrate a paper trader program with our short-term trading strategy. The creation on V5 bot took 15 hours to complete while exploration of financial terms and initialization took another 5.

With last week's realization that “the (high) returns we are seeing from the program are due to the fact that we're ‘playing a different game.’” and “In our current setup, we buy and sell at the same time of day, with no calculation delay, instant execution, no slippage, and none of the real-world challenges that traders face in trying to achieve a 7% return.” And with the use of a paper trader, most of these program similarities would be added. My goal this week was to help get the paper trader working.

My week started out with attempting to set up IBKR, but without data subscriptions, I couldn't use the API I wanted to. I texted Faiza, and she informed me about an issue with the paper trader not being accepted and also not having the correct permission we would need. We switched the focus of this week's goal to find another paper trader that can accomplish our goals. I attempted to find one but had no results, only finding US-based or very expensive traders.

From my previous week, I wanted to tune the trades per day and valid candidate cutoff



value, so I made a randomized search algorithm that could run indefinitely. In the process of setting this up, I was thinking about how a paper trader would interact with the program and came up with a verbalized description of the backtest -> paper trader pipeline.

The next day I worked exclusively on setting up Short Term Trading Bot v5.0. I got from data collection all through data cleaning, but without a peer review of the code and brainstorming process to understand the new trading algorithm, I stopped. I have the test.ipynb GitHub commit linked [here](#). Towards the bottom of the test file Version_5_Bot, you can find version 5.0.

The basis of this code was to switch from a historical data collection where we trade with the closing price of a stock to an updated version where the stock price is dynamically changed and check intraday to see if we have hit the loss or win percentages. The code is made with assumptions about the API and what data it will give and may need to be changed.

After my initial effort making the v5 Bot, I learned that our application to IBKR was denied, and we will have to change to a different paper trader. I deep-dived into UK-based trading companies and immediately found IG Markets. This looks a lot more promising than what I found at the start of the week. It is FCA certified, has FTSE SmallCap, has a paper trader with an exposed API and It is not super expensive.

6/30/2025

For my 4th week, June 30th, 2025, for my internship at Brickfort my task was to continue building a speculative live trading bot while we worked on getting a brokerage account funded and approved. The “speculative live trading bot” was built without the knowledge of what our data will look like but an understanding of trading programs to hopefully build a code base that could be refactored and integrated to work with the new API. A very important distinction between V5 bot and V4 bot was the “live paper trader” structure. This code was to simulate the same backtesting strategy on new live data meaning that this bot was to continuously work with oncoming data without look ahead bias and simulate slippage. My individual effort was put into coding our bot and any extra time was spent looking into and researching brokerage accounts which totaled up to 20 hours

My self prescribed goals were to fix bugged data cleaning and collection, add portfolio and equity stackers, pick candidates, remake the buy/ sell logic, and make a way to track returns. This function diagram I made for myself was useful to explain the process but without knowledge about what functions are called this diagram is hard to understand.

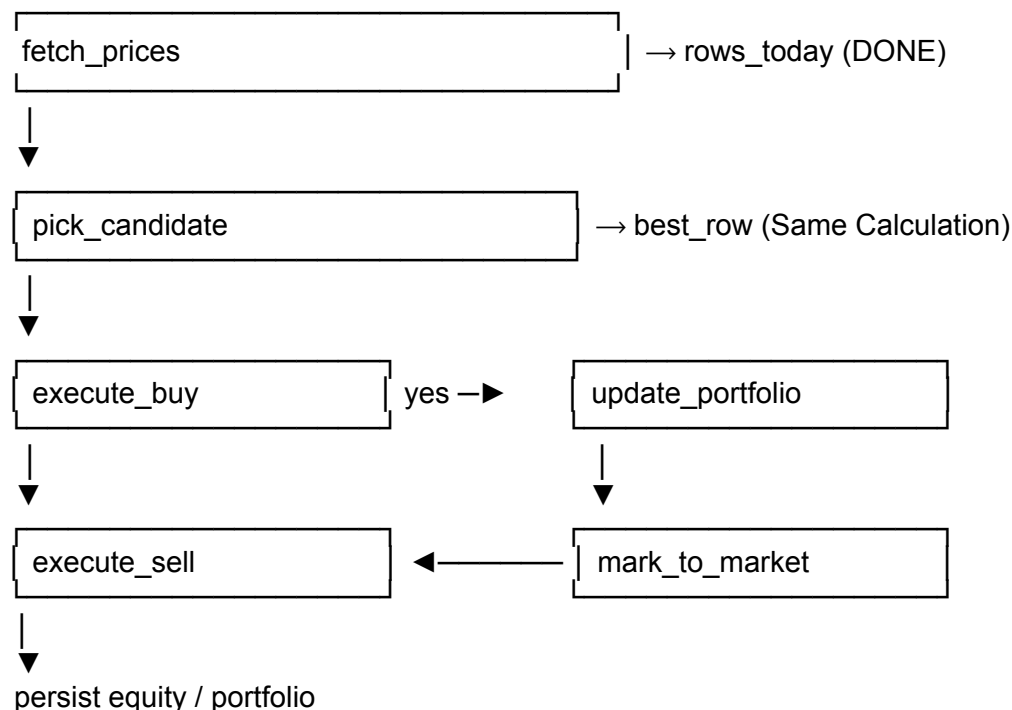


Fig 1: First step (fetch_prices) was done but not polished and would need changes throughout the process. Second step: pick the most profitable stock based on a calculation from Version 4 bot. Third Step: Buy that best stock which included tracking cash, price, shares, and slippage. From here there are two paths then a stock was bought we added it to the portfolio to track our positions and then updated the tracker. The other path is the sell logic that runs constantly checking stop loss and take profit.

During the rest of the week I was able to get a working program and bought 52 stocks of solaris for \$19,050 showing a +0.26% profit over one day. This FAKE paper trader that used fake money was a proof of concept that improved our team's knowledge base well.

7/7/2025

For my 5th week, July 7th, 2025, for my internship at Brickfort, after Emmanuel's attempts at funding a brokerage account we attempted to set up all of the brokerage accounts for Faiza's and I computers. This week also marked when I had to leave for a family reunion in Canada. Keeping a log of what happened is very difficult because of many meetings that were necessary and also our meeting had got pushed back till Tuesday every week so my time spent on these projects were over the span of 2 weeks. In total there was much more than 20 hours spent this week because faiza asked for me to send a commit during my vacation and their misunderstanding of when i was gone.

The first half of this week was spent working on researching the brokerage accounts that would be a good option for the work we wanted to complete. I found a UK company called IG Markets. They are FCA regulated; it's said that they have FTSE 100/250/SmallCap live data,

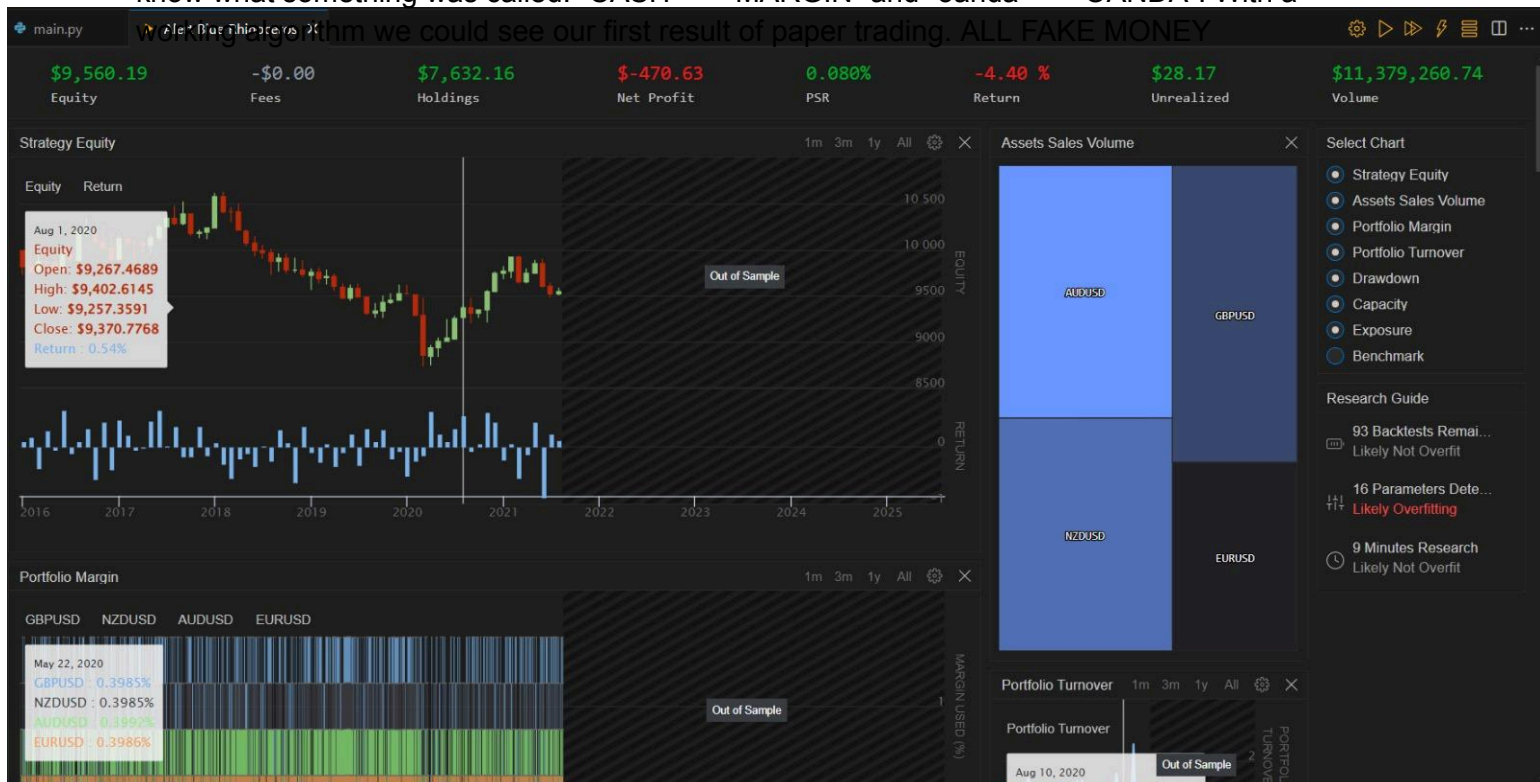


and they have a paper trader. But there were two issues. First, equities are mostly CFDs, not ISA-eligible and lastly, the program seems not future-proof, but I don't know much about this and I think it might be a perfect place to start. During meetings we decided what accounts we would set up and then we also made sure everyone could log into the accounts with 2 factor authentication. The process took quite a long time and had many issues like captcha test, 2FA, ip address locations, and slow brokerage acceptance/ paper work.

When I came back from my trip to Canada I would say the 2nd half of the project started due to having a funded live and historical data stream and environment to work with. Our diction landed on QuantConnect which is “an open-source, cloud-based algorithmic trading platform for equities, FX, futures, options, derivatives and cryptocurrencies”. QuantConnect is set up exactly for the work that we are trying to accomplish. It has built- in functions, data flow, Dockerized VS Code environment, sync with other brokerage accounts, and cloud based algorithm running. With built- in paper trading functions and live paper traders the code made last week became useless due to that functionally being directly built into prices.

The first day was spent reading documentation, watching tutorials, understanding LEAN and working on Faiza code. We both were new to this and took some time to download the necessary dependencies and get the local execution running.

My goal was to understand the LEAN Engine well enough to fix faiza code. I spent the whole day reading every page of documentation, set up tutorials and looking at Fiazia's simple trading algorithm to fix the bugs. I made comments documenting what I believed were the issues we were facing and much of it was incorrect end point declaration meaning she didn't know what something was called. “CASH” → “MARGIN” and “oanda” → “OANDA”. With a





7/30/2025

For my 6th week, July 28th, 2025, at my internship at Brickfort my goal was to research and understand trend following strategies and to create some ideas on how to build a logic to shift strategy between different trends (upwards, downwards, sideways). The short term bot only used the mean reversion strategy and succeeded in times of trend reversal but not following trends. So we had to identify ways/instances when we should shift between strategies meaning reversion, trend following. This task took 20 hours of work but plus writing 12 pages for SA408 took much more than that to complete.

This week combined work on the backtester for our trading strategies and research of a classification model. To summarize the work on our trading algorithm I attempted to implement a few things. First ATR based stop loss and take profit margins. I also tested using it to find out position size but my efforts did not reveal anything worthwhile. The other effort I made was to switch from Forex trading to crypto trading. Multi asset trading in quant connect is quite difficult and needs its own structure. Not having a brokerage account to trade crypto led me to only start the process of making multi asset trading work.

With both of these strategies not working, faiza gave me the goal of researching a trend classification bot. This task asks me to understand why stocks move, rebound and trend. Then also understanding how people can algorithmically classify up, down and flat trends. This will be then used to switch trading strategy from trend following to moving average. There are many ways to classify trends, some quite complex and some simple. The plan that i made and gave to faiza was the follows:

“My plan for the classification algorithm is to use a simple moving-average baseline that relies on the exponential moving-average crossover, defined as $EMAC(16, 64) = EMA(16) - EMA(64)$, to classify the trend. I want to be able to have a base line to compare my predictions against. After that, I'll build a Hidden Markov Model whose hidden states are uptrend, downtrend, and flat, using returns as the observed variable. If that step is finished early, or if we're not satisfied with the results, I'd like to add an RNN-based next-day trend prediction to verify the results. These idea comes from two Ritvik Math's video who makes data science videos, an article that shows implementation of HMM in QStrader, and a 2020 paper”

8/5/2025

For my 7th week, August 4th, 2025, at my internship at birckfort my goal was to work on building the trend classification bot. This week is not finished and I have one more after it because of the vacation I took in the middle of the internship. There was a lot of progress made so far on a Hidden Markov Model and Exponential moving average. I encountered many problems my first day while trying to get a research node working and spend hours downloading dependencies.

