Project Progression Check

Brian Liu (z5592091), Tamiz Rumey Jiffrey (z5592097),
Kirk Murillo (z5592102), Vedang Purohit (z5592103)

Introduction

Our project investigates the performance of various algorithmic trading strategies in simulated stock market environments. We mainly focused on how the greediness or decision aggressiveness of algorithms affects profitability and stability within simulated markets. By comparing baselines such as random and greedy algorithms with more structured approaches like moving averages and Bollinger-band mean reversion, we aim to identify the trade-offs between responsiveness and performance.

Progress Report

In the weeks since the project proposal, we have built most of the infrastructure and framework needed to conduct analysis and implemented a few algorithms, both control and research-based.

For implementation, we first solved the problem of data collection. We used Yahoo Finance's API with the python package `yfinance`, and got daily stock prices over 10 years. As for the stocks chosen, we accessed data from the top 30 companies on the ASX, as well as a few hand-picked stocks on other exchanges, such as Apple, Microsoft, and Nvidia.

Then, we built the rest of the backtesting framework in python, using OOP to allow for modular algorithm creation. We also implemented a way to visualise the stock prices and the stock worth at every point in a line graph, using the `matplotlib` package. Later on, we added metrics (e.g. Sharpe Ratio, CAGR) to determine the effectiveness of each algorithm.

Over time, we have been implementing and testing algorithms using the backtesting framework. Out of the seven implemented so far, three of them are meant as a control, and formulated by ourselves. The rest are based on real-life algorithms found in our research. The details about each algorithm are listed below.

Algorithms

Maximally Greedy (Control)
This algorithm will sell its current stock after noting a decrease. After an increase, it will buy stock. This runs in constant time after each new data point, only considering the previous one.

Random (Control)
This algorithm makes no prediction on the future value of the stock. It will randomly choose to buy, sell, or hold its stock after each update. So it runs in constant time for each data point.

Better than the first k (Control)
This algorithm looks for the best opportunity to buy and sell stock after a grace period of k inputs has occurred. If looking to buy, it will wait for the first price to exceed the first k inputs, and vice versa for selling. This runs in constant time.

Crossover with Simple Moving Average
The simple moving average of varying length can be used as a measure of the momentum of the stock value. So a simple indicator of growing stock value is when the shorter-term average crosses over longer-term averages, indicating a buy signal, and vice versa for selling. [1] Calculating the average of n elements takes linear time, but can be reduced to constant time for each individual data point by subtracting the newly ignored stock price from the windowing sum. [2]

Crossover with Exponential Moving Average
The exponential moving average is a special case of a weighted moving average where newer stock prices are given more weight in the averaging sum. This works in a similar way to the crossover method of the simple moving average. Calculating the exponential average has a constant time calculation for each incoming data point. [3]

Mean Reversion with Bollinger Bands
This mean reversion strategy assumes that stock prices tend to fluctuate around a long-term average, with deviations from this mean eventually correcting over time. So this algorithm employs Bollinger Bands, a volatility indicator formed by plotting upper and lower bands a fixed number of standard deviations above and below a moving average. The algorithm buys when the stock price falls below the lower band and sells

when it rises above the upper band. [4] Because the moving average and standard deviation are recalculated at each step, the per-step time complexity is $O(m)$, where m is the size of the rolling window.

Proximal Policy Optimisation Model (PPO)
We use an off-the-shelf stock trading model from HuggingFace. [5] At a basic level, a number of observations of indicators on the data can be made such as moving averages, Bollinger bands, volatility, etc. For each inputted data point, these observations are sent into the model, which will output a trading action. Given that this uses Bollinger bands to make decisions and uses a neural network, we estimate that the runtime of this has a lower bound of $O(n)$.

Performance Analysis

We used 4 main financial metrics:

- Compounded Annual Growth Rate (CAGR) refers to a compound interest rate that would result in the same returns and the returns of the investment. This was used to measure the growth rate of the investment.
- Average Trade (Money made/Number of trades) - measured how much money it made on average per trade, and was used to see the algorithm's efficiency.
- Sharpe Ratio and Calmar Ratio were used to evaluate the risk adjusted reward of the algorithm, comparing the performance of the algorithm while factoring in their volatility.

Preliminary Results

We averaged results over the 38 stocks we had long-term data on:

Algorithm	**Sharpe Ratio**	**Average Trade**	**Multiplier**	**Calmar Ratio**
Greedy	-0.207	-0.095	0.759	-0.037
Random	0.636	1.127	3.266	0.090
Best after 10	0.332	24.662	1.361	0.027
Simple MA (5, 21)	0.593	38.139	6.061	0.109
Expo MA (10, 20, 50)	0.511	94.213	4.806	0.093
Bollinger (1 STD)	0.446	0.669	1.944	0.062
Bollinger (2 STD)	0.573	4.593	2.477	0.083
PPO-ML	0.192	39713.100	98.544	0.194

Out of the algorithms we implemented (not including the off-the-shelf ML model), the moving average algorithms seemed to do best. The PPO-ML algorithm did far better than the others, although the performance was highly dependent on the stock being used. In terms of operating times, most of the algorithms were very fast at processing data, with no noticeable difference between even the constant and non-constant running times.

Challenges

- Implementation of ML PPO algorithm:
  We found an ML model - Proximal Policy Optimisation. But, this had poor documentation for the input space, causing roadblocks in making it work with our backtesting framework we had. To fix it, we studied the sample data processing script to determine the required indicators and data cleaning methods.

- Collecting data and switching from AlphaVenture to `yfinance`:
  For our backtesting framework, we required a range of required historical stock data. Initially we were collecting this stock data from AlphaVenture, but its API had strict rate limits, and lacked the range of data we desired to work with. Other platforms were also difficult to work with, so we switched to Yahoo Finance as it was simple to use and had a great range of historical data for backtesting.

- Algorithmic analysis:
  The algorithms we implemented initially did not have significant algorithmic complexity, which deviated from the original purpose of the project. Our research on simple trading algorithms also suggested that simple trading algorithms do not have heavy algorithmic complexity. We still researched and found interesting algorithms, e.g. Bollinger bands, that have linear, non-constant time complexity, and we aim to conduct more analysis on variations of algorithms.

- Categorising stocks:
  Our algorithmic analysis has not focused on the performance of algorithms in different stock conditions, as our progress was focused on creating and testing the overall performance of algorithms. We aim to solve this challenge by researching objective metrics of volatility to label stocks by their volatility, and conduct analysis on algorithmic performance on specific market scenarios.

Next Actions

In our proposal, we planned to create the backtesting framework, find data for tests, and have implemented 5 research based algorithms. We have implemented the backtesting framework, developed a solution for gathering data but we only have written 4 algorithms.

On the analysis front, we intended to evaluate performance in different stock markets as a main method of financial analysis, but we are yet to label our sets of stocks for this.

Overall, we slightly underestimated the time taken to research, implement and analyse every algorithm, and set up the infrastructure. We will take this into account in our plans going forward and scheduling only one more algorithm implementation before the final submission.

We plan to implement a trend-following/momentum-based algorithm in the coming weeks, potentially based on Relative Strength Index (RSI). On the analysis front, we will complete the labelling of stocks by volatility to conduct market type analysis.

In terms of analysis, we wish to complete the labelling of stocks so we can do market-type based analysis. Then, we have to start the writeup for the final report and poster, which should be started at least at the end of week 8. In order to allow time for this, we pushed back our "soft deadline" for these deliverables.

Project Timeline

Date	Aim	Notes
Friday, 7 Nov	RSI-based stock algorithm	
Monday, 10 Nov	Start on report	Try to use market-based analysis
Saturday, 15 Nov	Start on poster	
Friday, 21 Nov	Poster showcase	
Sunday, 23 Nov	Report submission	

Bibliography

[1] *Simple Moving Average (SMA) Explained: Definition and Calculation Formula.* Investopedia. URL: https://www.investopedia.com/terms/s/sma.asp (visited on 10/20/2025).

[2] *Moving average*. In: *Wikipedia*. Page Version ID: 1314422585. Oct. 1, 2025. URL: https://en.wikipedia.org/w/index.php?title=Moving_average&oldid=1314422585#Simple_moving_average (visited on 10/20/2025).

[3] *Exponential Moving Average (EMA): Definition, Formula, and Usage*. Investopedia. URL: https://www.investopedia.com/terms/e/ema.asp (visited on 10/21/2025).

[4] Suzanne Kvilhaug Full Bio Suzanne is a content marketer et al. *Understanding Bollinger Bands: A Key Technical Analysis Tool for Investors*. Investopedia. URL: https://www.investopedia.com/terms/b/bollingerbands.asp (visited on 10/24/2025).

[5] Adilbai. *Stock Trading RL Agent using PPO*. 2025. URL: https://huggingface.co/Adilbai/stock-trading-rl-20250704-171446.