The Dexterity Capital Trading Algorithm Project

Zian (Shawn) Cheng

Overview

The Dexterity Capital Trading Algorithm Project is about developing a trading algorithm such that the ending balance of a trading day is maximized, given the starting balance of $25,000 (USD) and 0 BTC. The algorithm should also perform reasonably well on the data from a different day, meaning that attention needs to be paid to prevent overfitting.

I developed an initial algorithm to generate Decision Set 1, resulting in an ending balance of $25701.98. It contains 24 bid attempts, 20 bid executions, 24 ask attempts and 20 ask executions. After some adjustments made to the algorithm, I generated Decision Set 2 with an ending balance of $25721.97. It contains 102 bid attempts, 21 bid executions, 102 ask attempts and 21 ask executions.

Data Exploration and Visualization

Given the three data files, I loaded them into Python Pandas data frame and started data manipulation. I converted the timestamp column to date-time. Since qty represents opposite meanings when side == “a” or “b”, I created a new column “new_qty” such that it’s equal to qty when side == “a” and it’s equal to (-1) * qty when side == “b”.

Simultaneously, I began to do some research online about price change indicators. A lot of articles mentioned that Relative Strength Index (RSI) is a great indicator in cryptocurrency market so I decided to try it first (https://medium.com/@coinloop/trading-with-rsi-2584deca18c8). However, after generating some bid attempts using RSI, I found it often lagging, meaning that when it generates the bid signal (threshold between 70 and 80), the price is already very close to the peak. I decided to give up on the RSI approach after confirming with Abraham.

After that, I created some Data Visualization using matplotlib. Figure 1 shows the time series plot of three exchanges. I found plenty of opportunities for arbitrage, the chance of bid at a low-price exchange and ask immediately at a high price on another exchange. I confirmed my idea online and it seems arbitrage is indeed a great way to trade in the cryptocurrency market (https://hackernoon.com/intricate-cryptocurrency-arbitrage-profit-like-a-ninja-fc60075cc22f). I kept this idea in mind and explored further.

Figure 1. Plot of time series of prices for Exchange 1, 2 and 3

Figure 2 shows the price, qty and new_qty plots for Exchange 1. I discovered there is usually a huge increase/drop in bid/ask volume before a huge price change. After searching online, I confirmed my hypothesis by reading a lot of articles saying demand and supply changes are significant factors that drive the cryptocurrency price changes (https://blog.hybridblock.io/8-factors-affecting-cryptocurrency-price/). Inspired by this, I decided to build a decision model to bid and ask based on recent market bid/ask volume. This also became the algorithm for generating my Decision Set 1.

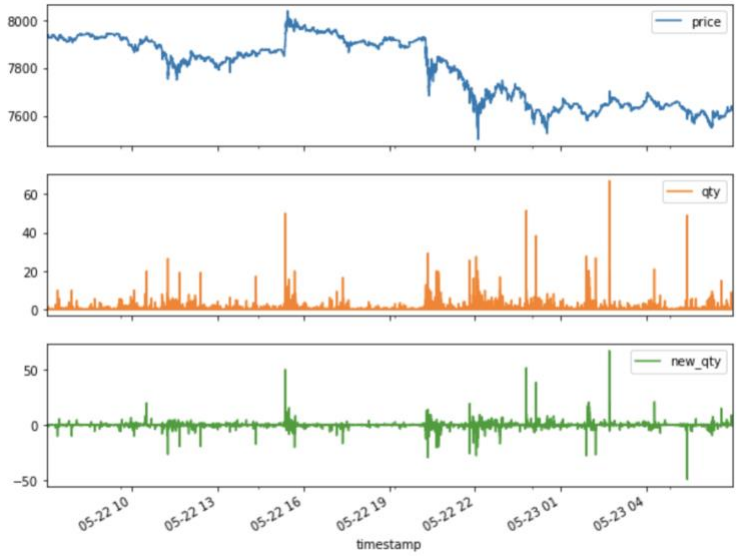

Figure 2. Plots of time series of price, qty and new_qty for Exchange 1

The idea of running Moving Average (MA) on new_qty came to my mind because it's a good measure of recent demand and supply changes. If MA is positive, there's more ask than bid in volume and if MA is negative, there's more bid than ask in volume. The next step is to select a period - I selected a period of 5 based on a comparison of plots between MA5, MA15 and MA100. I discovered that MA5 would be able to generate signals in time and it allows me to set a clear threshold for bid/ask. It could be seen from Figure 1 and 3 that a threshold in the range of [7,13] might be good for generating bid signals.

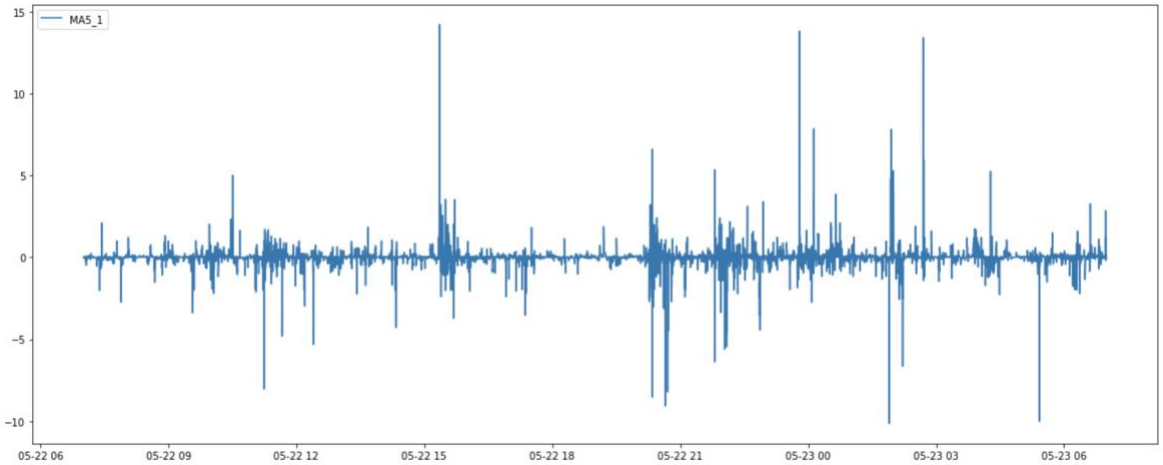

Figure 3. Plot of time series of MA5 for Exchange 1

Initial Algorithm and Decision Set 1

I combined the three data frames and sorted the result using timestamp. The combined data frame is then used to build the algorithm. Here is an overview of how the initial algorithm works:

- Loop through the combined data frame row by row
- Check if MA5 >= MA5_threshold; if yes, bid as much as the balance allows using the current lowest price, record this price as bid_price
- Check if price - bid_price >= price_difference_threshold or time is within 1 minute of end time; If yes, ask as much as the BTC allows using the current highest price

To generate Decision Set 1, I set MA5_threshold at 10 and price_difference_threshold at 50 after testing some combinations of the parameters. It's the best result so far, with an ending balance of $25701.98. A lot of assumptions and simplifications were made in this algorithm. For example, I assumed all bid/ask attempts would be successful immediately to easily keep track of balance and BTC. It's important to pick good parameters (MA5_threshold, price_difference_threshold) in this case. If MA5_threshold is too low, too many bad bid attempts would be made and if MA5_threshold is too high, good opportunities of bid would have been gone. If price_difference_threshold is too low, opportunities of winning more money would be wasted and if price_difference_threshold is too high, there's a risk of never being able to ask until last minute.

To find the best parameters, I tried to code an optimization tool to help me pick the parameters that would generate the highest ending balance. However, to prevent overfitting, I stopped the development of this optimization tool as suggested by Abraham.

Improved Algorithm and Decision Set 2

Moving Average is a simple and naïve approach to measure the volume of new_qty since it weighs the past data equally, ignoring inhomogeneous sampling. I searched online about good methods to deal with inhomogeneous sampling and decided to try Exponential Moving Average (EMA) because it weighs the recent data point a lot more significantly than far data points. This

aligns with the need of a good signal metric since the metric needs to generate signals fast and accurately after the big/ask volume changes.

Furthermore, I incorporated the arbitrage opportunities into the algorithm and here's how it works:

- Loop through the combined data frame row by row
- Check if EMA5 >= EMA5_threshold or current highest price - current lowest price > price_difference_threshold; if yes, bid as much as the balance allows using the current lowest price, record this price as bid_price
- Check if price - bid_price >= price_difference_threshold or timestamp is within 1 minute of end time; If yes, ask as much as the BTC allows using the current highest price

Compared to the initial algorithm, the improved algorithm would bid not only when EMA reaches the threshold but also when there's an arbitrage opportunity. As expected, the bid/ask attempts significantly increased and resulted in an ending balance of $25721.97. It is also worth noticing that the successful attempt rate decreased. This is mostly likely because this algorithm would still make bid/ask attempts even when previous attempts have not been executed yet. A lot of the time, the price hasn't achieved the price_difference_threshold yet and I still have BTCs on hand while more arbitrage opportunities come in simultaneously. Since there are plenty of arbitrage opportunities, it would require fast and successful buy and sell to prevent this kind of situations from happening. To prevent excessive bid/ask attempts, I limited the number of total bid attempts to be under 100 and generated the current result.

Further Improvements

My second algorithm incurs a lot of bids and asks but the success rate is low. Therefore, approaches should be taken to evaluate whether there are pending attempts and to decide on if it's appropriate to make new attempts. If given more time, I would also like to discover the potential of bidding/asking using a portion of the balance/BTC instead of full balance/BTC.

Furthermore, I would also explore more about the signal used to generate ask attempts. In the two algorithms, I used pre-selected thresholds to generate ask attempts and potentially one could use another appropriate metric to generate ask attempts.

Challenges and Milestones

The biggest challenge I encountered was understanding the logic behind cryptocurrency trading. Although I had plenty of experiences in time-series data and Data Analytics in general, it was always challenging to get started with a project in a new field, where I need to do extensive research about the background before coding. It felt like walking in the darkness in the beginning but thanks to Abraham and the whole Dexterity team, I could follow the guidance, study the materials and grasp the project gradually. For example, I was completely unsure about which metric to use for generating bid/ask signals. It was through extensive research and Data Visualization that I found bid and ask volume could be good sources for signals. This discovery also became one of the greatest milestones in this project and I was super excited when I made the discovery.

Another big challenge in the project was the coding of the algorithm. It required me to have a deep understanding of the trading logic and it took me a lot of thinking to build the algorithm from scratch. In the beginning, I tried to search online for similar trading algorithms but barely found anything. I guess most people want to keep their trading algorithms a secret so that they have a better chance of winning against the market. I developed the algorithm gradually and after hours of work, I was finally able to see the algorithm generating some profit. That was when I realized my algorithm really worked and I felt a great sense of accomplishment.

Conclusion

To sum up, this was a very challenging and fun project to work on. I would like to thank the whole Dexterity team again for creating this project and providing me with guidance throughout the process. I developed my knowledge in cryptocurrency trading drastically within a few days of working on this project. It has become a valuable part of my Financial Data Analytics experience and I enjoyed the project from beginning to end.