MATH 527: Project
Reinforcement Learning for price impact

Zachary Scialom & Yi Zhong

5th of December 2022

ILLINOIS INSTITUTE OF TECHNOLOGY

Contents

List of Figures

1 Motivation

This project aims to use Reinforcement Learning in order for an investor to minimize his price impact. The focus has only been made on a liquidation perspective using market orders (MOs).

How can an agent sell a large amount of shares and yet minimize adverse price movements which are a consequence of his own trades ?

The main idea is that the agent will slice the parent order into smaller parts and try to execute each one of these child orders over a period of time.
A simplified solution of this problem has been brought and can be found here: `https://github.com/mfrdixon/ML_Finance_Codes/blob/master/Chapter9-Reinforcement-Learning/ML_in_Finance_Market_Impact.ipynb`.
We have decided to complexify and to improve this file by using real-life data and by including a new key process: order flow. The goal is to capture the price impact phenomenon thanks to the data and therefore not to assume it deterministic.
At last, we have also considered different values of risk aversion as the risk dimension is essential and interesting in this liquidation problem.

2 Contribution and improvements made

2.1 Context: optimal execution with only Market orders

The goal of this liquidation problem is the following: the agent's objective is to choose the rate at which he liquidates a certain amount of shares so that he obtains the maximum revenue from the sale. All shares must be liquidated at terminal time, T.

What makes this problem interesting is the trade-off existing between price impact (trading quicker) and price risk (complete the trades on a longer period). Indeed, trading slowly may result on an exposure towards the uncertainty of the values of the future prices. On the other hand, trading faster means more price impact and therefore a higher execution price.

Thus, the time the agent takes to space out and execute the smaller orders is genuinely important.

2.2 Including the order flow

One of the main contributions of our project is the fact that we included the order flow process in order to estimate the price impact. The latter is no longer assumed to be constant and will vary according to the volumes traded over time.
First of all, let's justify why the informative order flow has been used. Market efficiency theories tell us that the resulting price process is not predictable and so are daily asset returns. In other words, the price levels over a whole day are difficult to predict but trading activity, usually measured using volume, has a different structure. Volume displays a stronger time series persistence.

However, two other variables are related to market efficiency: volatility index and order flow.

Order flow (OF) is an ambiguous term which can be interpreted as the "sum of signed volumes" submitted thus far. It represents well the net demand for an asset. It is basically the difference between the number of shares bought and sold.
More precisely, the order flow at time t is given by:

$$F_t = \sum_{p \leq B_t} V_t(p) - \sum_{p \geq A_t} V_t(p) \tag{1}$$

where A_t , B_t are the best ask and bid prices, and Vt (p) is the volume of MOs posted at the price level p.
For the project, we considered the sum of the ten best bid volumes minus the sum of the ten ask volumes. An important property of order flow is that past and present values of the intraday order flow have predictive power for its future changes. (Centered) Order-flow displays a strong mean-reversion around its mean. It can be modeled by an AR(1) process:

$$\tilde{F}_{t+1} = \phi_1 \tilde{F}_t + \sigma \epsilon_{t+1} \tag{2}$$

Using this property enables to predict the future price. Indeed, thanks to a least-squares (LS) regression, we get:

$$\Delta S_t = \lambda \Delta F_t + \hat{\sigma} \eta_t \tag{3}$$

where S is the midprice, λ is to be estimated, and η is a zero-mean white noise with $Var(\eta_t) = 1$.
λ, is actually the price impact, which affects directly the execution price.
We chose S to be the midprice that we sampled in order to address the fact that it changes very rarely on a high-frequency scale.
Once the price impact λ and the parameter ϕ_1 are estimated (respectively with linear regression and Dickey-Fuller test), we can make predictions about the future price changes based on the past and present values of the order flow. In particular, if the present time t, then, our prediction for the price change in the interval [t, t + h] takes the form:

$$\mathbf{E}[S_{t+h} - S_t | \tilde{F}_t = \tilde{f}] = \lambda \mathbf{E}[F_{t+h} - F_t | \tilde{F}_t = \tilde{f}] = -\lambda \tilde{f}(1 - \phi_1) \tag{4}$$

To illustrate the predictive power, we have realized the prediction of future prices using the order flow (independantly from the ML file). This plot can be found in the helpul codes that we joined in the zip file:

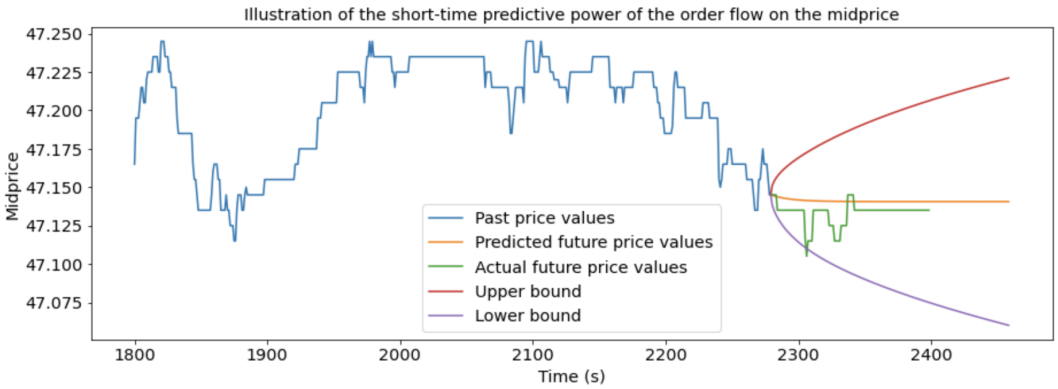

Figure 1: Illustration of the predictive power of the order flow on the future price

2.3 Work with real life data

Another main difference we brought to the initial file about market impact is the fact that we worked with real life data.
The stock prices have not been generated according to a specific model, such as a Geometric Brownian motion for instance.
A more detailed explanation of the data will be provided later. However, working with real-life data was also a challenge as we had to work with a finite MDP . Thus, we had to think about ways to keep as much information as possible without penalizing our overall analysis. The choices made about the different discretizations will be provided in the next section.

2.4 A version of Almgren-Chriss/Obizhaeva-Wang model

The whole code and Machine Learning work makes sense financially. The model and variables used constitute a version of Almgren-Chriss/Obizhaeva-Wang model. Let's detail the different variables of this model:

- ν_t represents the rate at which shares are liquidated
- Q_t^ν, is the inventory (number of shares held)
- $\tilde{S}_t$, is the unaffected true price (midprice)
- S_t^ν, is the impacted true price: given by $\tilde{S}$ adjusted by the agent's trading rate:

$$S_{t+1}^\nu - S_t^\nu = \underbrace{F(\nu_t, \nu_{t-1}, ..., \nu_1)}_{Price impact} + S_{t+1}^\nu - S_t^\nu \tag{5}$$

  The function F includes the permanent price impact as well the temporary price impact.

- X_t^{nu}, is the cash position accumulated by the agent at time t:

$$X_{t+1}^\nu - X_t^\nu = -S_{t+1}^\nu \nu_t \tag{6}$$

Obviously, we integrated those variables in our Machine Learning framework.
The goal of the next section is to detail the different parameters used for the Machine Learning treatment and to provide the Mathematical problem statement.

3 Mathematical Problem Statement

Our goal is to optimally liquidate a certain number of shares over the course of a trading window. As a result, we want to maximize the terminal wealth.

3.1 State variables and Finite MDPs

It is genuinely important to properly define the state variables for a Reinforcement Learning problem. In our case, we have three state variables: Q_t (the inventory), S_t (the midprice) and t (time considered for the trading window).
Obviously, we have to work with finite MDPs therefore it was necassary to discretize the values for the midprice and the time.
First, for the midprice, unlike the simpler version presented in class, it does make a different to trade at \$47.35 or \$ 47.60. The goal was not to take the integer associated with the midprice (for example, 48) as we would lose in accuracy. As a result, we created an array of integers (index) of the corresponding prices that could be encountered. We decided to take into account 150 values defined as followed: Then, we created an array of integers (index) associated with this array.

```
[45.75  45.85  45.95  46.05  46.15  46.25  46.35  46.45  46.55  46.65
 46.735 46.74  46.745 46.75  46.755 46.76  46.765 46.77  46.775 46.78
 46.785 46.785 46.79  46.795 46.8   46.8   46.805 46.81  46.815 46.815
 46.82  46.825 46.83  46.835 46.835 46.84  46.845 46.85  46.855 46.86
 46.865 46.87  46.875 46.88  46.885 46.89  46.895 46.9   46.905 46.91
 46.915 46.92  46.925 46.93  46.935 46.945 46.95  46.955 46.96  46.965
 46.97  46.975 46.985 46.99  46.995 47.    47.005 47.015 47.02  47.025
 47.035 47.045 47.055 47.06  47.065 47.07  47.075 47.08  47.085 47.09
 47.095 47.1   47.105 47.11  47.115 47.12  47.125 47.13  47.135 47.14
 47.145 47.15  47.155 47.16  47.165 47.17  47.175 47.18  47.185 47.19
 47.195 47.2   47.205 47.21  47.215 47.22  47.225 47.23  47.235 47.24
 47.245 47.25  47.255 47.26  47.265 47.27  47.275 47.28  47.285 47.29
 47.295 47.3   47.305 47.31  47.315 47.32  47.325 47.33  47.335 47.34
 47.345 47.35  47.355 47.36  47.365 47.37  47.375 47.38  47.385 47.39
 47.395 47.4   47.405 47.41  47.415 47.42  47.425 47.43  47.435 47.445
 47.45  47.455 47.46  47.56  47.66  47.76  47.86  47.96 ]
```

Figure 2: Discretization for the midprice values

Obviously, when the values were not exactly met among those in the array, we made sure to take the one associated with the closest index. Therefore, we more precisely worked with the index associated with the midprice value to get to the next step.

For time, the process was similar. The choice of our trading window will be explained later but we decided to start trading at 1:00 pm for different reasons. Thus, we had to make sure to work with indices again. The number of time steps were around 8000 for this problem.

At last, for the inventory, we decided to proceed by blocks. Therefore, we had 40 blocks with a block size of 50 which gives a total of 2000 shares to liquidate.

Making this problem fit to real-life data was a challenge especially because we had to make certain choices in order to run the code. Indeed, those choices (number of blocks, time steps, number of actions...) have mainly been dictated by how far could we go with the memory on Google Colab without any crashes. For example, those crashes came from lines like "np.zeros()" in the Sarsa function where all the dimensions were involved.

Our goal was to keep a significant amount of time for trading and still have a good amount of values for the inventory.

3.2 Actions

There is a total of 5 actions which correspond to the number of blocks that can be sold: 0,1,2,3,4.

3.3 Reward functions

For all the time steps, the reward function included the following expression:

$$r_t = \mu - \lambda Var[S_{t+1}X_{t+1}] \tag{7}$$

However, we also added other values linked with rewards.

First, when predictions were made, we had to make sure that the statistical environment, meaning the statistical tests (Dickey-Fuller and Yule-Walker), was good enough to make predictions. To do so, we made sure the p-values were 0.01. If this condition was not verified, the agent was penalized if it took the action to sell a block whereas it was rewarded when it did not trade at all.

Moreover, we added a reward according to the behavior of the order flow.

Indeed, when the excess order flow is tilted to the buy side, the agent should slow down or not trade since it anticipates that excess buy will push the price upwards. On the other hand, if the excess order is tilted to the sell side, it should increase the trading rate as other traders might push the price further down.

3.4 SARSA

At last, for the choice of the algorithm, we decided to use SARSA as Q-Learning was surprisingly way too slow in order to get results (1 hour of execution vs 2h 30 minutes).

4 Description of the data

4.1 Data description

Thanks to the University of Toronto, we successfully got some limit order book data for algorithmic trading usage, which can perfectly meet the requirements in our own project. In the raw file, we have the entire trading day data with time stamp in each millisecond. However, our research only requires several columns from the primary data to help us rebuild and visualize the limit order book, which is displayed in figure 3.

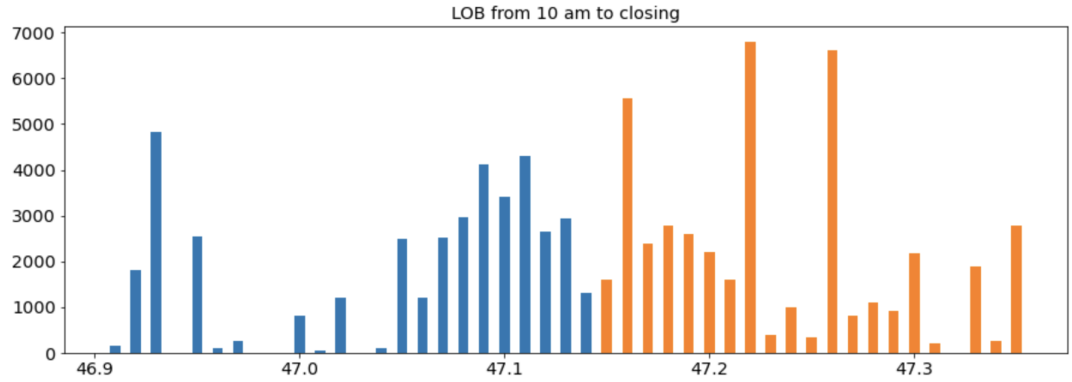

Figure 3: Limit order book (LOB) over the course of the day

In order to express our idea clearly and reduce the time for running code, we decided to set our decision time at each second. However, the time stamp from raw data is in each millisecond in the 'EventTime' column. In consequence, we needed to reduce the time steps to each second by only keeping the data every ten time stamps. To prepare the order flow data for our price impact estimation, we extracted the first 10 best bid and ask volume from 'BuyVolume' and 'SellVolume' column respectively. Moreover, we also needed the best bid and ask price from 'BuyPrice' and 'SellPrice' to prepare for the midprice, which has been further used in estimating the price impact and in predicting the execution price.
Here is displayed the evolution of the bid price and the ask price with the data chosen (which is Microsoft stock (MSFT), 3rd of November 2014).

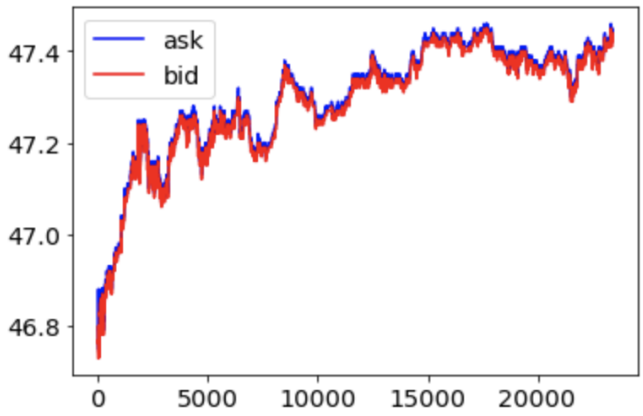

Figure 4: Evolution of the bid price and ask price over the course of the day: MSFT, 3rd of November 2014

A PDF file as well as some Jupiter notebooks have been added to the zip file in order to understand more about the data if needed.

4.2 Choice of the trading window

The trading window we chose is the following: from 1:pm to 3:30pm. Here is the evolution of the midprice over this trading window:

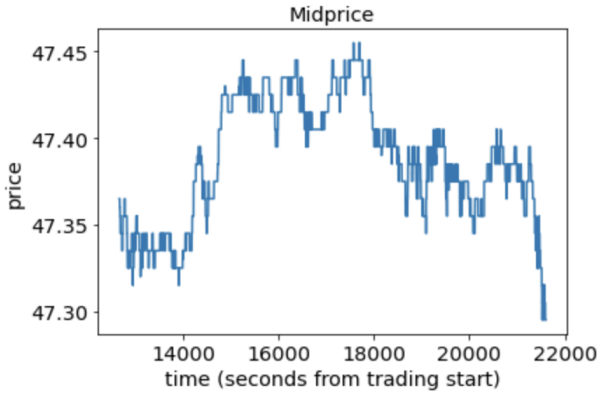

Figure 5: Evolution of the midprice over the trading window

This choice is based on two factors.
First, it is known the intraday volume displays a U-shaped curve. Indeed, it is high in the morning, then declines, and gets high again at 2:00pm. This surge at 2:00pm can be explained by the fact that there are more announcements than the norm, and they generate greater volume. Also, traders who have not been able to meet their liquidation might accelerate their execution as the market approaches its time to close.
As a result, we wanted to study a trading window that includes an increasing volume over time.

The second reason is purely technical. We wanted to observe variations in the sign of the order flow as it directly impacts our reward function.

5 Results and Performances

The results and performances were pretty convincing. As mentioned before, in such problems, it is always interesting to change the value related with risk. In our case, we decided to modify λ, the risk aversion parameter, that can be found in the penalty term of equation 7 (reward function). All the results have been obtained using Python.

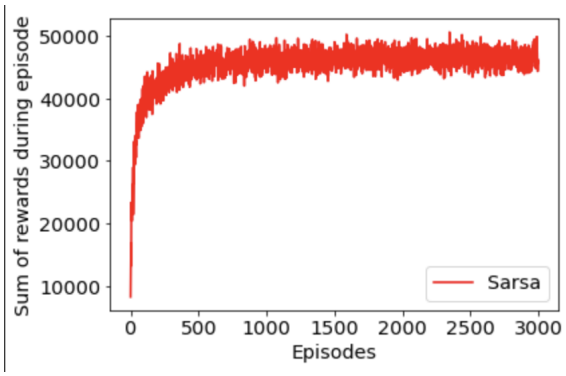

Figure 6: Sum of all rewards with $\lambda = 0.005$

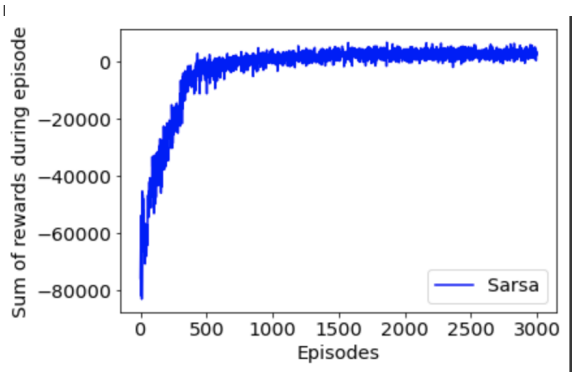

Figure 7: Sum of all rewards with $\lambda = 0.01$

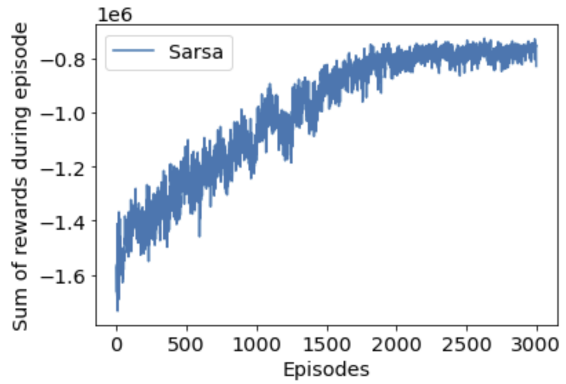

Figure 8: Sum of all rewards with $\lambda = 0.1$

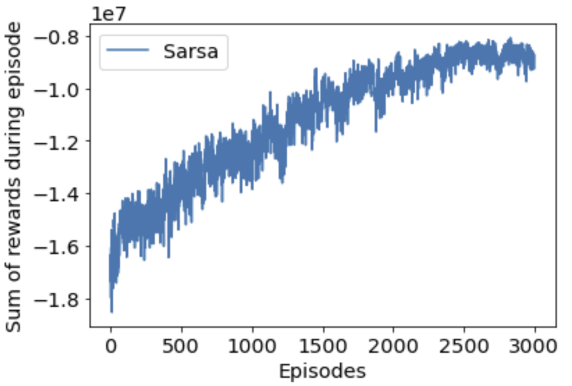

Figure 9: Sum of all rewards with $\lambda = 1$

It is interesting as this value displays a true meaning: the higher is this value, the quicker the agents wants to sell. Indeed, in order to maximize the rewards, the agent does not want to carry the penalty term if λ is high. Therefore, he trades quicker.

Nonetheless, if we look at the evolution of the order flow on the trading value, we can see that it is clearly on the buy side and we tend to penalize the agent when he trades when the order flow is positive.

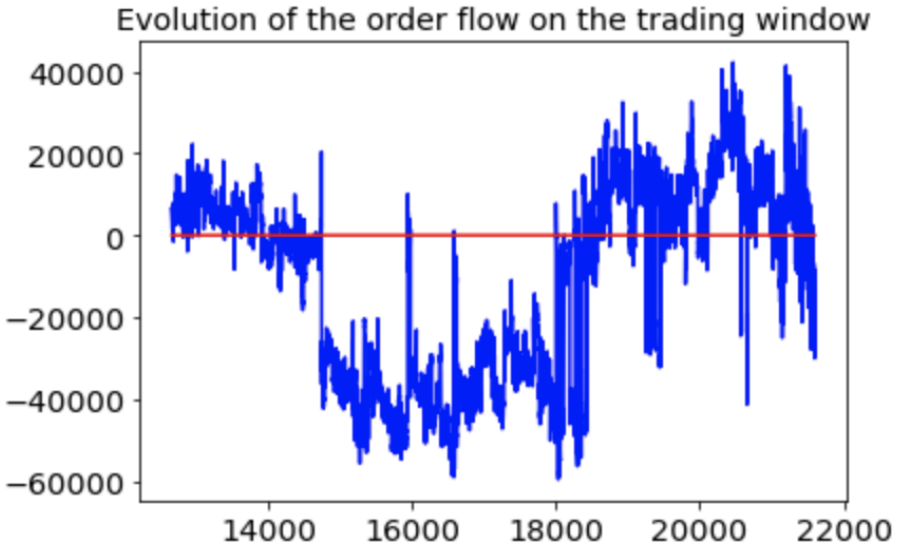

Figure 10: Evolution of the order flow on the trading window

This is why the rewards genuinely decreases when λ increases. On the other hand, a lower lambda means that the agent is willing to wait in order to make the best decisions possible.

It is also essential to compare the terminal wealth accumulated. To do so, we used the Time Weighted Average Price (TWAP) as a reference.
The name is explained by observing that the total cash received for the sale is equal to:

$$X_T = -\sum_{i=0}^{T-1} S_{i+1}^{\nu^*} = \frac{Q_0}{T} \sum_{i=0}^{T-1} S_{i+1}^{\nu^*} \tag{8}$$

which is the number of shares time the time-average of the price per share taken over the execution period. Using our agent, we are really close to the result obtained with TWAP. We were pretty satisfied by this result as TWAP is a really simplistic approach that does not assume any temporary impact, $\lambda = 0$. Thus, to compute TWAP, we directly use the true real price at time t+1 as the execution price. TWAP also offers a good balance as the trading rate is equal to $\nu^* = \frac{Q_0}{T}$ especially on a short period such as the one we chose (two hours).
Here are the results for the terminal cash:

Model considered	Terminal Wealth (in dollars)
Time Weighted Average Price (TWAP)	94,318
Our model: $\lambda = 0.005$	92,165
Our model: $\lambda = 0.01$	91,906
Our model: $\lambda = 0.1$	91,845
Our model: $\lambda = 1$	91,827

Figure 11: Terminal cash values obtained

6 Possible improvements

There are different ways to improve this project. First of all, it would be interesting to include Limit Orders (LOs) in order to be more realistic.

Secondly, it would be better to study the course of a whole day. We did not manage to do so as increasing the time steps had a direct impact with the memory offered by the servers of Google Colab.

At last, the prediction made for the price could be improved. Indeed, it would be better to work with the microprice and the LO Imbalance. The microprice can be defined as followed:

$$S = A\frac{V^b}{V^a + V^b} + B\frac{V^a}{V^a + V^b} \tag{9}$$

where A and B are the best ask and best bid prices, while V^a and V^b are the volumes (in number of shares) of LOs posted at the best ask and the best bid prices respectively. The values of microprice change more frequently as the LOs on the best bid and ask sides are updated. This makes the microprice more amendable to the standard statistical methods.
LO imbalance can be defined as followed:

$$I = \frac{V^b - V^a}{V^b + V^a} \tag{10}$$

LO imbalance measures the instantaneous imbalance between the buy and sell LOs at the present time. The most important property of the LO imbalance is that it has short-term predictive power for the signed traded volume and for the midprice.

7 Conclusion

In conclusion, the introduction of order flow helps refining the execution process by getting rid of the assumption that the price impact is constant and deterministic.

Moreover, the price impact due to our trade is also reflected in the stock price of next time period, which is one of our state variables and helps improving our decision process. The result from our reinforcement learning displays how the agent will optimally execute under a dynamic price impact and different risk aversion factors. The whole project has been tested with real-life high frequency data which was challenge to fit in the Machine Learning framework.

Our simulation with reducing risk aversion factors displays a convergence to the TWAP result, which strongly proves the success of our new model. However, the research is not complete. By introducing more parameters, such as the microprice or the LO imbalance, we may get a model with a higher predictive power and therefore a statistically more reasonable result. Introducing other order types will also make our model more realistic and optimal compared to only use market order for execution.

References

[1] Matthew Dixon, Igor Halperin, Paul Bilokon - Machine Learning in Finance: From Theory to practice

[2] Álvaro Cartea, Sebastian Jaimungal And José Penalva Frontmatter - Algorithmic and High-Frequency Trading