Iceberg Orders and Shape of the Limit Order Book

by Benyamin Azoulay

ICEBERG ORDERS AND SHAPE OF THE LIMIT ORDER BOOK

by

Benyamin Azoulay (CID: 01573890)

Department of Mathematics
Imperial College London
London SW7 2AZ
United Kingdom

Thesis submitted as part of the requirements for the award of the
MSc in Mathematics and Finance, Imperial College London, 2018-2019

IMPERIAL COLLEGE LONDON

THESIS DECLARATION FORM

(for candidates whose degree will be awarded by **Imperial College**)

Declaration: I hereby confirm that the work contained in this thesis is my own work unless other wises stated.

Name Benyamin Azoulay **CID** 01573890

Title of Thesis ICEBERG ORDERS AND SHAPE OF THE LIMIT ORDER BOOK

Month and year of submission for examination September 2019

Date 10/09/201

Signature of Candidate [illegible]

Acknowledgements

I would like to thank Deutsche Bank's Electronic Equity Trading team for providing data, high-speed computation facilities and guidance for my project. Especially Lars Boyde, Torsten Schoeneborn, Yuan Shen, Sacha Drevet and Adam Shannon have been a great help through insightful comments and discussions. Also, I would like to express my gratitude to my project supervisor Aitor Muguruza providing guidance whenever needed.

Abstract

This thesis is a general study of the limit order book. In particular we analyse the detection and prediction of iceberg orders and we study the limit order book dynamic using machine learning methods. We start by a review of basic linear methods and machine learning methods related to validation methods. We will give an overview of the limit order book and its mechanism, with a particular attention to how iceberg orders work. After that, we will describe the data and the algorithm used to detect iceberg orders, and we show that there are some complications due to the data. Then we will introduce and use a model to compute the price impact of order book events using a linear regression. The last part is a general application of machine learning methods, using many features to predict targets related to the shape of the order book.

Contents

1 Introduction

1.1 Motivation

The theory of Supply and Demand tries to explain how the relationship between the availability of a particular product and the desire (or demand) for that product has on its price. Progress in information technology and the increasing deregulation of exchanges have led to a wide adoption of the limit order book for price formation (in simple words the list of prices at which one can transact for a given security). Knowledge of forecasting methods and their application to the dynamic of the limit order book, could vastly improve PnL (profit and loss) through optimized trading costs.

1.2 Limitation

The initial goal of this thesis was to detect and predict the volume of iceberg orders, but due to issues with the data and the difficulty in obtaining results, the thesis became a general investigation of the limit order book, using quotes of the limit order book in a high frequency setting. In order to do this we use simple linear regression methods.

1.3 Approach

Due to the different vocabularies used to refer to the shape of the limit order book, we will first give in the first section a fast overview of the most general terms used. The definition and mechanisms of iceberg order, more specifically, vary from one exchange to another. For iceberg orders, we will define them more carefully and explain how their mechanisms work as clearly as possible for the exchanges of interest, as we will need a good understanding of their mechanisms in order to detect them.

Even if we met lots of issues when doing our analysis of iceberg orders, we will still present in Section 3 the detection algorithm that we have implemented and the results we get from it, emphasizing the issues encountered all along our analysis.

Linear regression is perhaps one of the most well-known and well understood algorithms in statistics and machine learning : even if it is a simple algorithm, it often gives very good results on different financial problems (which tend to include large and noisy data). This is why we will not use other models in our thesis and try to show the variety of results we can get with such a model. We will instead concentrate our attention on the data, the consistency of our features and the robustness of our models through time and stocks. In our description of models in Section 4, we will start out by reviewing OLS and its extensions Ridge and Lasso where L^1 and L^2 penalties are used to regularize the linear regression. We will consider these for both predictive modelling and feature selection, as especially Lasso can give a quick estimate of feature importance.

In Section 5, we will describe and implement a model which links linearly, the price change and the imbalance of the supply and demand in the order book during a specific time interval. We will then see what could be the possible applications of this linear relationship.

Finally, the last section is a general application of Machine Learning methods, including feature engineering and feature selection, to predict targets related to the order book.

2 Limit Order Book Description and Mechanism

In this section, we describe how trading takes place on LOB markets, and we will give some precise definitions related to the limit order book. We will also have a precise look at how iceberg orders work.
More details can be found in [1].

2.1 Preliminaries

Definition 2.1 (Limit order book)**.** It is a device that the vast majority of organized electronic markets (all equity, futures and other listed derivatives markets) use to store the list of all the interests of market participants. It is essentially a database that contains all the orders sent to the market, with their characteristics such as the sign of the order (buy or sell), the price, the quantity, a timestamp giving the time the order was recorded by the exchange, and often, other market-dependent information.

We call this kind of market in which buyers and sellers meet via a limit order book, as an order-driven market.
Essentially, market participants can submit three types of orders to an order book:

Limit order: We specify a price and a size at which we want to buy or sell, at any point in time in the future.

Market order: We specify a quantity that we want to immediately buy (or sell) at the best available opposite quote.

Cancellation order: Cancel an existing limit order.

Buy and sell orders are matched as they arrive over time, subject to some priority rules. The priority is always based on price first, and then for most of the markets on time.

Limit orders are stored in the order book, until they are either cancelled or executed against an incoming market order. The ask price P^A is the price of the lowest limit sell order. Conversely, the bid price P^B is the price of the best limit buy order.

Definition 2.2 (Spread)**.** The bid-ask spread at time t is $s(t) := P^A - P^B$.

Remark 2.3. The spread is always positive

Definition 2.4 (Mid price)**.** The mid price at time t is the average between the bid and ask $m(t) := (P^A + P^B)/2$.

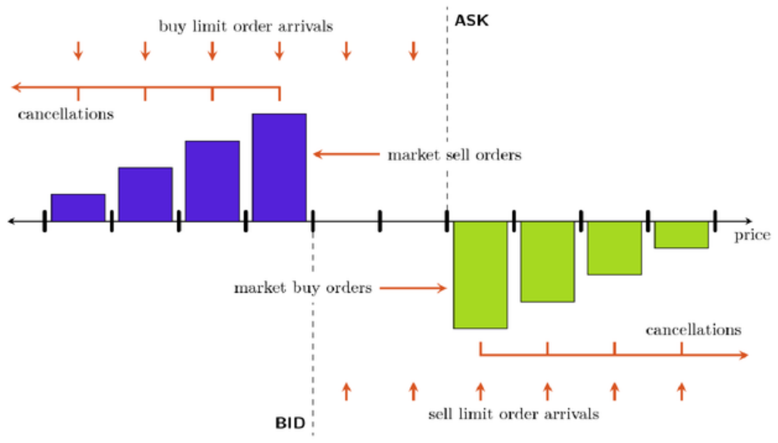


Figure 1: An illustration of a limit order book

We can see from Figure 1 (taken from [3]) an example of the different orders presented just above. On the right we can see the ask side, on the left the bid side. Buy limit orders and sell limit orders are showing in purple and green. We can also see the market buy and sell orders which cross the spread (here of size two ticks) in order to be executed.

Definition 2.5 (Tick size)**.** The tick size is the minimum permitted price variation in a security price.

The tick size for a particular stock depends on the exchange where it is traded, and on the stock price. For example, on the NYSE the minimum price variation for all stocks over $1 is $.01. This means that the NYSE will accept orders to trade at 10.00 or 10.01, but it will not at accept an order priced at $10.001.
For example, Deutsche Boerse has adopted the following ticks for shares in its Xetra system:

Lower Price Limit (€)	Upper Price Limit (€)	Tick Size (€)
0	9.999	0.001
10	49.995	0.005
50	99.99	0.01
100	∞	0.05

Table 1: Xetra tick-size regime

Definition 2.6 (NBBO)**.** The National Best Bid and Offer (NBBO) is the lowest available ask price and the highest available bid price for investors buying and selling securities in U.S.

Remark 2.7. The European Best Bid and Offer (EBBO) is the European equivalent of the NBBO.

Last, we will define a measure that is often used as a benchmark by investor.

Definition 2.8 (Volume-weighted average price)**.** In finance, volume-weighted average price (VWAP) is the ratio of the value traded to total volume traded over a particular time horizon (usually one day). It is a measure of the average price at which a stock is traded over the trading horizon.

$$VWAP = \frac{\sum_j P_j.Q_j}{\sum_j Q_j},$$

where:

P_j is the price of trade j;

Q_j is quantity of trade j;

j is each individual trade that takes place over the defined period of time.

We will now focus on iceberg order in the next section.

2.2 Iceberg Orders

We will now concentrate on a specific kind of order find on many exchanges, know as iceberg orders. A trader seeking to execute a large trade faces the problem that signalling his trading intentions to the market can both help and hurt the effective execution of his trade causing a negative price impact.

A signal may help attract counter-parties to reduce the time it takes to complete the trade. On the other hand, it may also attract opportunistic traders who make the completion less likely or more costly. Traders' solutions may include following dynamic order submission strategies or trading off exchange because such strategies limit the amount of information that is revealed about their trading intentions. Many exchanges provide their own solution by allowing a particular order type known as an iceberg order.

Definition 2.9 (Iceberg Order)**.** An iceberg order is a special type of limit order, where in addition to a price, side, and size, the user is required to specify a "max show value". Max show is the upper limit of the fraction of the total order size that is shown to the market, while the remainder of the order volume remains hidden.

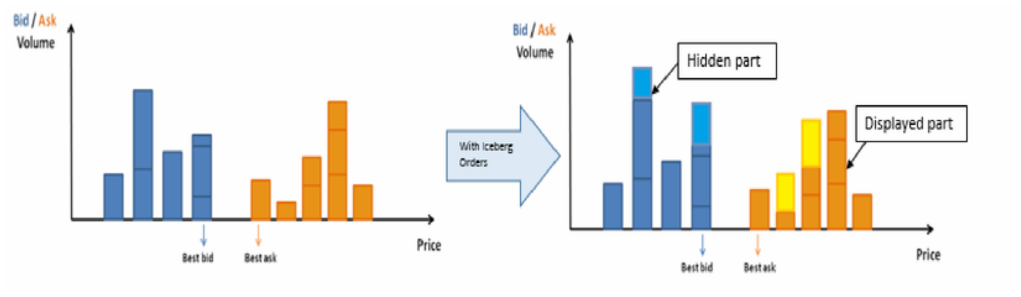


Figure 2: View of Limit Order Book without Iceberg Orders (left) and with Iceberg Orders (right)

In Figure 2 we can see two views of the limit order book, one with iceberg orders and one without. We see that iceberg orders can sit on different levels.

On most exchanges, iceberg orders follow these rules:

- Displayed parts of orders take precedence over non-displayed parts at any price point.
- Non-display portions of icebergs taking precedence over fully Hidden Orders (where no displayed-parts are shown).
- If the incoming order is sufficiently large then each peak at the same price point will be executed against in time priority. However, once peak volume of all iceberg orders at a price level has been fully executed then any remaining incoming volume is allocated to the hidden volume of each iceberg order pro-rated on the remaining size of each iceberg order.
- The display (peak) quantity of an Iceberg Order is refreshed once the display quantity has been fully executed.
  On refresh, the peak will be prioritized after all existing, visible orders at that price point.
- Customers have the option to have the refreshed peak size randomized.

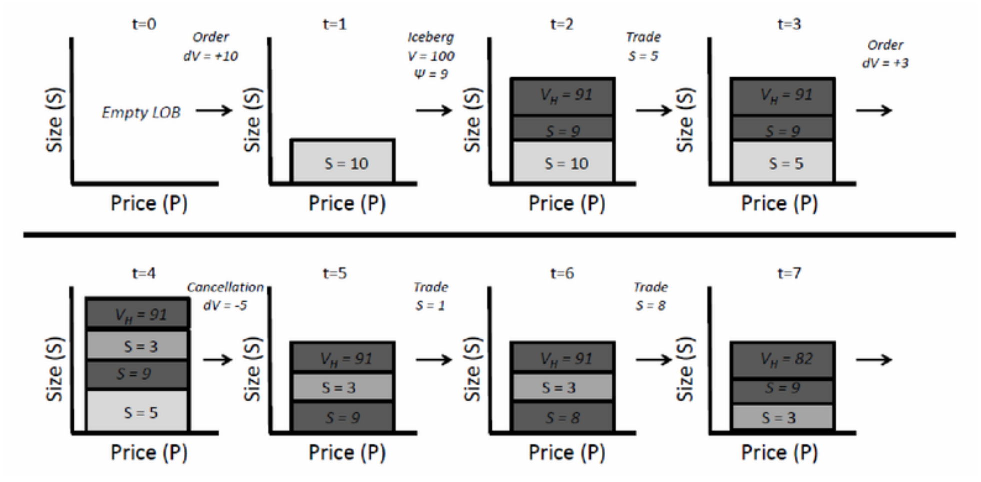


Figure 3: Schematic of Iceberg Order Mechanics

These mechanics are now illustrated with an example In Figure 3. Suppose we are on the ask side at the best price P, at t = 1 a limit order of size 10 is added.

- An iceberg order is specified at t = 2 with a total size V = 100 and a max show $\psi = 9$. The first 11 tranches of the iceberg are of size $\psi = 9$ and the final tranche is size $\psi = 1$, so N = 12 (where N is the number of tranches).
- At time step t = 3, there is a market order to buy S = 5 shares, the volume of the first limit order changes to S = 5.
- At time step t = 4, a limit order of total size V = 3 is added and it is prioritized after the displayed part of the existing iceberg order but before the hidden part.
- At time step t = 5, there is a cancellation of the limit order of size 5, now the displayed part of the iceberg order is the first in time priority.
- At time step t = 6, there is a market order to buy S = 1 shares, the displayed volume of the iceberg order changes to S = 8.
- At time step t = 7, there is a market order to buy S = 8 shares, the displayed volume of the iceberg order has been completely removed. The limit order of size V = 3 now has time priority and a displayed tranche of the iceberg order of size V = 9 has been refreshed.

We have talked about iceberg orders that are on the best bid or the best offer. But there is another kind of iceberg orders, called Hidden Midpoint Order.

Definition 2.10 (Hidden Midpoint Order)**.** Firms may post their orders to the book as hidden midpoint orders to access liquidity at the midpoint. Resting hidden midpoint orders are eligible to execute against market orders, limit orders, non- displayed orders and other midpoint orders. No matter the spread, the hidden midpoint order will always peg to the true midpoint of the NBBO.

Remark 2.11. For example if the NBBO is \$10.02 to \$10.04, a Hidden Midpoint buy order will price at \$10.03..

2.3 Form of the data

We will now present the different forms of data used for the analysis. Principally, I have used quotes and trades data for different stocks on different exchanges. Most of the data comes from Nebula, which is the data provided directly by the exchanges.

2.3.1 Data for the analysis of Iceberg Orders

For my analysis of the iceberg orders, I used the quotes and trades for 7 trading days of June 2019 of Microsoft. The quotes are all the updates of the limit order book for the different days of trading, which for one trading day approximately a million rows of updates on average.
It is constructed with the following columns :
$[date, venue, bid, ask, bidsz, asksz, bidno, askno, seq, xts]$.

- The venue corresponds to the exchange where the stock is traded. For instance XNYS corresponds to the New York Stock Exchange. We have the data for thirteen different exchanges.

- The columns bid, bidsz and bidno correspond respectively to the best bid price, size in number of shares and the number of different limit orders on the best bid (same for the ask).

- The column seq corresponds to the sequence number of the update. For example if the quote update is due to a market order execution, the market order sequence number will match the one of the quote update.

- Finally, the column xts corresponds to the time of the exchanges with a millisecond precision.

- The trades are all the trades that happened for each trading day. The size is about one hundred thousand rows for each day. It is constructed with the following columns : $[date, venue, price, size, agrside,$

- They are similar to the quotes columns, except for price, size which are the price, size of the trade.

- The column agrside can take three values, "Sell" or "Buy" if it is a sell or buy market order, or None if it is an iceberg order (we do not

- The column trdype specify the type of the trade: it can take a value like "Open" or "Close" if it is during the opening or closing auction, "Auto" if it is a regular trade , or "Hidden" if it is a trade on an iceberg order or an hidden midpoint order.

Remark 2.12. We only keep the rows where trdtype takes the value "Auto" or "Hidden".

2.3.2 Data for the analysis of the price impact

For my analysis of the price impact, I have the quotes of ten different stocks from the FTSE 100 stock index for the month of June 2019. There are about two hundred thousand rows for each day of trading. The data corresponds to the price, size and number of limit orders of the five best bid and ask levels of the quotes, associated with the time (precision of milliseconds) and the date.

2.3.3 Data for the analysis of the shape of the order book

For my analysis of the shape of the order book, I have a sample of the quotes and trades of twenty three different stocks from the FTSE 100 for the month of June for each minute of the day. I have two different sets of quotes: one where we take the last quotes for each minute bin, the other where we have the means of the trades and quotes for each minute bin. We have for each set of quotes and trades the medians of the prices and numbers of limit orders, and the means of the sizes for each level and the means of the trade prices.

3 Iceberg Orders

In this section we will explain how the data is used for prediction of iceberg order volume, how my algorithm works to detect the iceberg orders and finally some results. It is on this part that I have encounter the most issues, most of them with the data, and so through the different parts I will explain what are the main problems that I have encountered and why they have prevented me from obtaining good results.

Two papers have been of great used to understand iceberg orders, the first one is [4], where I take inspiration from their detection algorithm. The other one is [5], which gives a clear understanding of iceberg orders mechanisms and in which they introduce a Bivariate Gaussian Kernel to predict the volume.

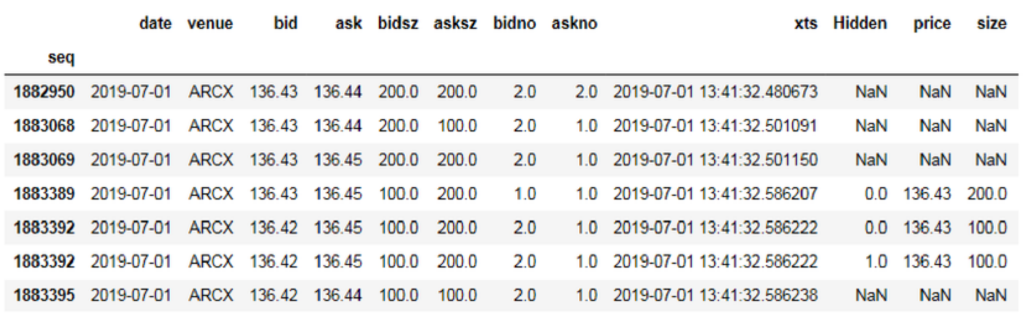
3.1 Data Cleaning

The data set consists of Microsoft quotes and trades of seven trading days. We have transformed the trade column "trdtype" as Hidden with value 1 if the trade is on a hidden part and value 0 if it is on a displayed part. We will call a trade of type 1 a hidden trade and a trade of type 0 a displayed trade.

To simplify the analysis, we have added together the trades that have the same sequence number and same time of execution. Normally when a market order is executed against n different limit orders, in the quotes data we would see n different updates, now we see only one update.
The problem is that we are losing some information: for hidden trades we do not know on how many iceberg orders they are executed against. So we will consider, iceberg orders on the same level (best bid or best ask) as one. It distorts the analysis but it was necessary to make the analysis feasible in the given time.

Remark 3.1. Notice also that the lot size of the quotes are rounded to the nearest hundred, in consequence a bid size of 270 shares will appear as 300.

The quotes and the trades have been merged using the sequence number column, the venue column and the date column. There are more updates of the quotes than trades, so most of the time there is not any trade sequence number associated with a quote. One of the first problem is that, on the contrary, not all the sequences numbers of the trades match with a sequence number of the quotes, e.g. the sequence of a trade at a hidden midpoint order will not match any quotes sequence. To encounter this problem, the quotes and trades have been joined using a left outer join, dropping all the trades for which the sequences numbers do not match any quotes.



seq	date	venue	bid	ask	bidsz	asksz	bidno	askno	xts	Hidden	price	size
1882950	2019-07-01	ARCX	136.43	136.44	200.0	200.0	2.0	2.0	2019-07-01 13:41:32.480673	NaN	NaN	NaN
1883068	2019-07-01	ARCX	136.43	136.44	200.0	100.0	2.0	1.0	2019-07-01 13:41:32.501091	NaN	NaN	NaN
1883069	2019-07-01	ARCX	136.43	136.45	200.0	200.0	2.0	1.0	2019-07-01 13:41:32.501150	NaN	NaN	NaN
1883389	2019-07-01	ARCX	136.43	136.45	100.0	200.0	1.0	1.0	2019-07-01 13:41:32.586207	0.0	136.43	200.0
1883392	2019-07-01	ARCX	136.42	136.45	100.0	200.0	2.0	1.0	2019-07-01 13:41:32.586222	0.0	136.43	100.0
1883392	2019-07-01	ARCX	136.42	136.45	100.0	200.0	2.0	1.0	2019-07-01 13:41:32.586222	1.0	136.43	100.0
1883395	2019-07-01	ARCX	136.42	136.44	100.0	100.0	2.0	1.0	2019-07-01 13:41:32.586238	NaN	NaN	NaN

Figure 4: Input data

As we can In Figure 4, there are hidden trades and displayed trades on ARCX (Archipelago Exchange, a subsidiary of the NYSE) for which the sequences numbers match quotes sequences numbers. There is a shift between the quotes and the trades : For a trade on the right of the table, on the left we see the quotes update of the limit order book after the trade occurred.

Remark 3.2. The data has then been sorted by sequences then time.

Another problem is that for hidden trades, the side (buy or sell) is not specified. To find the side, we look for each hidden trade, up to three quotes updates before to find if the price of the trade is matching either the bid (so that it is a sell hidden trade) or the ask (a buy hidden trade). If there is no match we drop the hidden trade.

Now that we have our data set cleaned and organized, we will try to compute some characteristics for each iceberg order.

3.2 Detection Algorithm

In this section we will describe the algorithm used to compute some characteristics for each iceberg order, using the data described above. We are not trying to detect the iceberg orders (it is already done by the data with the flag Hidden), but assembling the different hidden trades to estimate the volume of the iceberg orders. We will call a detection of an iceberg order the first time there is a hidden trade on the best bid or best ask, because before we did not know there was an active iceberg order at this time on this side.

What we are trying to find is first, the distribution of the displayed ratio of total volume for each iceberg order:

$$Ratio = \frac{Displayed}{Total}$$

Then the goal is, to predict from the known size of an iceberg order, its total size. In order to do this we need to compute for each iceberg order its displayed part. It is not that easy to compute because, when an iceberg order is put on an exchange the displayed size can be randomized each time there is a refresh of the displayed volume. Another thing that makes it difficult to evaluate the displayed part is that we consider different iceberg orders as one, so if between two detections of an iceberg order that contains multiple iceberg orders, one of them is cancelled or fully filled, the two displayed parts will not be the same.

The solution found was to do at each detection, the mean of the previous estimated displayed volumes. Each detection of an iceberg order (so each hidden trade) is preceded with a regular trade on the displayed part (with the flag hidden =0). We are using this trade size to compute the estimated displayed volume of the iceberg order EstDisp. We are then using the following formula to compute the displayed mean.

$$\begin{aligned} DispVolMean[0] &= 0 \\ DispVolMean[n] &= \frac{(n-1)DispVolMean[n-1] + EstDisp}{n} \quad \text{for } n > 0, \end{aligned}$$

where:

EstDisp is the estimated displayed volume using the n^{th} detection of an iceberg order;

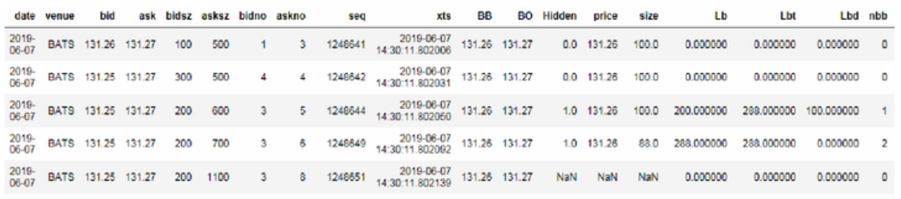
DispVolMean[n] is the mean of the displayed volume after the n^{th} detection of the same iceberg order, with n = 1 the first detection.

We are careful to not average all the estimated displayed volumes to avoid using future information in the past. This allows the algorithm to be run in real time.

We will also compute at each quote updates, the known cumulative volume of active iceberg orders at the bid and at the ask and the total volume of active iceberg orders at the bid and at the ask (It will be our target), it takes 0 if there is not any iceberg order active at this level at this time. Last we will also compute the number of first detections for an iceberg order, with value 0 if no iceberg order is active.

```
Algorithm 1: Detection Algorithm
  Input: Quotes and Trades merged
  Result: cumulativeVolume, totalVolume, volumeDisplayed, numberDetection
  initialization;
  for exc in venue do
      for day in date do
          for side in [bid,ask] do
              Select the subset of the quotes;
              while Looping on the subset do
                  if Hidden trade then
                      Look backward if there exists an iceberg order same price on the last 5
                        minutes still active;
                      Compute displayed part;
                      Look forward until move of price of the quotes;
                  end
              end
          end
      end
  end
```

In algorithm 1 above the input corresponds to the data In Figure 4. Once we detect an hidden trade when we are on the subset of interest (exchange, date and side), the algorithm is decomposed in three parts after. The first is to look backward in the quotes to see if there was an iceberg order at the same price in the last five minutes, then to compute the estimated displayed part using the equation **??**. Last we look forward and compute all the different volumes of interest until the iceberg order is not active or there is a move of price.



date	venue	bid	ask	bidsz	asksz	bidno	askno	seq	xts	BB	BO	Hidden	price	size	Lb	Lbt	Lbd	nbb
2019-06-07	BATS	131.26	131.27	100	500	1	3	1248641	2019-06-07 14:30:11.802006	131.26	131.27	0.0	131.26	100.0	0.000000	0.000000	0.000000	0
2019-06-07	BATS	131.25	131.27	300	500	4	4	1248642	2019-06-07 14:30:11.802031	131.26	131.27	0.0	131.26	100.0	0.000000	0.000000	0.000000	0
2019-06-07	BATS	131.25	131.27	200	600	3	5	1248644	2019-06-07 14:30:11.802060	131.26	131.27	1.0	131.26	100.0	200.000000	288.000000	100.000000	1
2019-06-07	BATS	131.25	131.27	200	700	3	6	1248649	2019-06-07 14:30:11.802092	131.26	131.27	1.0	131.26	88.0	288.000000	288.000000	0.000000	2
2019-06-07	BATS	131.25	131.27	200	1100	3	8	1248651	2019-06-07 14:30:11.802139	131.26	131.27	NaN	NaN	NaN	0.000000	0.000000	0.000000	0

Figure 5: Output data of Detection Algorithm

In Figure 5, we can see the output of the algorithm for the bid side. BB and BO correspond to the best bid and the best offer, computing by keeping track of all the best bids and asks on the different exchanges. Lb corresponds to the cumulative hidden size found when an iceberg order is active on the best bid, Lbt to the total hidden volume of the active iceberg order (so using information from the future), Lbd to the estimated displayed size (positive only for the quotes where the iceberg order is detected) and finally nbb the number of hidden trades on the active iceberg order on the bid.

On the third line we see that there is a hidden trade of size 100 with an estimate displayed size of 100 (using the last quotes where we find the bid at the price of the trades, here the first row), so the cumulative hidden size is 200. Then on the row just after, there is another hidden trade of size 88. There is a move of price on the best bid (from 131.26 to 131.25) so it is the end of the iceberg order: the total size of the iceberg order is 288.

3.3 Results

Most of the iceberg orders disappear (there is a change of quote price) just after they are detected. Many reasons can explain this, but the main theory is that exchanges are on purpose reporting hidden trades later, to avoid people use the information in real time of the iceberg order, and thus to cancel the goal of iceberg orders. Indeed, when looking at the data we see that lots of hidden trades have time stamps that do not correspond to their sequences: their reporting times are greater by some seconds to times of displayed trades that have sequences numbers just before or just after them.

After having dropped the hidden trades for which sequences numbers do not match quote sequences, as well as hidden trades where we cannot find the side, we are left with 45 % of the hidden trades. In consequence the total volume of the iceberg is not detected, and the total volume found does not correspond to the real size of the iceberg order.
Nevertheless, with the data left we have found an estimate of the distribution of the ratio displayed on total for the bid and the ask separately.

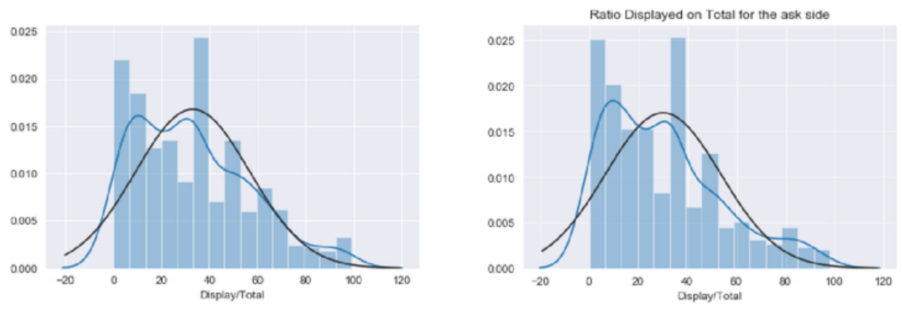


Figure 6: Ratio Displayed on Total volume for the bid (left) and ask (right)

In Figure 6 we can see the density estimate of the ratio displayed on total size. The x coordinate corresponds to the percentages of the ratio, the black line corresponds a fitted Gaussian and the blue line to a kernel density estimation. We notice that the distribution is left skewed: those who are using iceberg orders have an interest using them, only if the displayed size is much smaller than the total size. We notice also that there are some spikes on the ratio of 10% ,33% and 50%, which corresponds to the total size being 10 times, 3 times and 2 times bigger than the displayed size. So clearly there is a human impact on the choice of the ratio.

4 Background

In this section we provide an overview of the models used for my different predictions. The general framework corresponds to a model that takes n variables as input, all the variables are real numbers and our target variable is also a real number.

4.1 General Framework

For a given (x, y), we define first the idealized map F, which is the ground truth:

$$F : \mathbb{R}^n \to \mathbb{R}$$
$$F(x) \mapsto \hat{y},$$

We then define our model, which is the model used to fit the reality, it receives x and outputs $\hat{y}$, the prediction of y. We can rewrite it as:

$$M^P : \mathbb{R}^n \to \mathbb{R}$$
$$M^P(x) \mapsto \hat{y},$$

with P the parameters of the model.

Definition 4.1 (Loss Function)**.** Loss functions are used to determine the error (the loss) between the output of our algorithms and the given target value. In other terms, the loss function expresses how far off the mark our computed output is.

More explanations about loss functions can be found in [6].

Definition 4.2 (Objective Function)**.** The function we want to minimize or maximize is called the objective function, or criterion. When we are minimizing it, we may also call it the cost function.

The loss function is for a single training example, while the cost function is over the entire training set. Therefore, a loss function is a part of a cost function which is a type of an objective function. For example, a probability of generating training set in maximum likelihood approach is a well-defined objective function, but it is not a loss function (however you could define an equivalent cost function).
Below, we can find an example of cost function that we will use for our analysis.

Example 4.3 (Mean Squared Error). *The MSE measures the average amount that the models predictions vary from the correct values.*
$MSE(P) = \frac{1}{N}\sum_{i=1}^{N}(y_i - M^p(x_i))^2$ *with* N *the number of training example.*
We can think of it as a measure of the models performance on the training set. It penalizes more the models predictions that are far from the correct values.

Definition 4.4 (Training/Test set). The training dataset is the actual dataset used to train the model.
The test dataset is a dataset used to provide an unbiased evaluation of a final model fit on the training dataset.

Suppose we have a set of observations (x_i, y_i) for $1 \leq i \leq N$, the set of which is denoted $(X, Y) \in \mathbb{R}^{N(n+1)}$, we want to find the set of parameters P which minimize the expected prediction error (our objective function):

$$E[L(Y, M^P(X))|X = x],$$

with L a loss function.
For our analysis we will take the loss function associated with the MSE, $L(y, x) = \frac{1}{2}(y - M^p(x))^2$.

Generally we split our model in two part, one is the training or learning set and the other is the test set. We minimize the expected prediction error on the training set in order to find the best parameters for our model. We do not use the test set to find the best parameters of the model in order to evaluate our model, and avoid over-fitting.

Definition 4.5 (Over-fitting). The production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations.

Remark 4.6. We want our model to be general and to give good results on new data. Different techniques exist in order to prevent over-fitting, in our case we will use Walk-Forward optimization. More details about it is given in Section 4.8.

In the different prediction problems encountered, we will set 75% of our data for the training set and 25% for the test set.

Remark 4.7. [13] determines a general formula for the validation-set and training-set size ratio. The result of the paper for the ratio is the square root of two complexity parameters. In our different cases we will simply use a general ratio used in Machine Learning (even if the most common one is 80/20 ratio known as the Pareto principle).

4.2 Assessing Goodness-of-Fit

Different metrics can be used to evaluate our predictions depending of the problem. For regression problems we will use R-Squared (R^2) metrics:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{N}(Y_i - \bar{Y}_i)^2},$$

R^2-score is the percentage of the dependent variable variation that a regression model explains. Its values can be between $-\infty$ and 1, with 1 meaning that we can perfectly predict our target. So our goal is to maximize the R^2-score.

R^2-score does not indicate if a regression model provides an adequate fit to your data. A good model can have a low R^2 value. On the other hand, a biased model can have a high R^2 value. In [7], more explanations can be found on why in some fields of study your R^2 values are bound to be lower (studies related to human behaviour for example).

Despite that R^2-score has limitations, we will use it for our different studies to assess the goodness of our fits.

Remark 4.8. We have not done that in our studies, but we could also have computed the Root Mean Square Error (RMSE) in order to have an idea of the sample standard deviation of the differences between predicted and observed values.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2},$$

Contrary to the R^2-score, we want to maximize the RMSE.

4.3 Bias-Variance Trade-off

We would like to reason about the expected loss (Prediction Risk) over (x_i, y_i) for $1 \leq i \leq N$ our training set D, and a test point (x_*, y_*). We will consider the MSE for this explanation

Theorem 4.9 (Bias-Variance decomposition)**.** *We will decompose the expected loss into:*

$$E_{D,(x_*,y_*)}[(y_* - M(x_*|D))^2] = Noise + Bias^2 + Variance$$

Proof. Define the true model h (unobserved): $y_* = h(x_*) + \epsilon_*$, we will note $h(x_*) = h_*$.

We are using

$$\begin{aligned}E_{D,(x_*,y_*)}[(y_* - M(x_*|D))^2] &= E_{D,(x_*,y_*)}[(y_* - h_* + h_* - M(x_*|D))^2] \\ &= E_{\epsilon_*}[(y_* - h_*)^2] + E_D[(h_* - M(x_*|D))^2] \\ &+ 2E_{D,(x_*,y_*)}[y_*h_* - y_*M_* - h_*h_* + h_*M_*]\end{aligned}$$

where we used a trick by adding and subtracting the true model h_*, and then developing the square using the linearity of the expectation.

Using that $y_* = h(x_*) + \epsilon_*$ and $E[\epsilon_*] = 0$, we have:

$$\begin{aligned}E_{D,(x_*,y_*)}[y_*h_* - y_*M_* - h_*h_* + h_*M_*] &= h_*h_* + E[\epsilon_*]h_* - E[M_*]h_* - M_*E[\epsilon_*] - h_*h_* + h_*E[M_*] \\ &= 0\end{aligned}$$

So we end with,

$$E_{D,(x_*,y_*)}[(y_* - M(x_*|D))^2] = E_{\epsilon_*}[(y_* - h_*)^2] + E_D[(h_* - M(x_*|D))^2] \tag{4.1}$$

$E_{\epsilon_*}[(y_* - h_*)^2]$ is the noise term, we cannot control it.
$E_D[(h_* - M(x_*|D))^2]$ is the model estimation error, it is what we want to minimize.

We will denote $E[M(x_*|D)] = \bar{M}_*$, the expectation of our model.
We have,

$$\begin{aligned}E_D[(h_* - M(x_*|D))^2] &= E[(h_* - \bar{M}_* + \bar{M}_* - M(x_*|D))^2] \\ &= E[(h_* - \bar{M}_*)^2] + E[(M(x_*|D) - \bar{M}_*)^2] \\ &+ 2E[h_*\bar{M}_* - h_*\bar{M}_* - \bar{M}_*M_* + \bar{M}_*^2]\end{aligned}$$

where we used the same trick as before with this time our $\bar{M}_*$, and then developing the square using the linearity of the expectation.

As before we have $E[h_*\bar{M}_* - h_*\bar{M}_* - \bar{M}_*M_* + \bar{M}_*^2] = 0$.

Moreover we have $\begin{cases} E[(h_* - \bar{M}_*)^2] = (h_* - E[\bar{M}_*])^2 = (Bias)^2 \\ E[(M(x_*|D) - \bar{M}_*)^2] = Variance \end{cases}$

Getting finally,

$$E_{D,(x_*,y_*)}[(y_* - M(x_*|D))^2] = Noise + Bias^2 + Variance$$

□

When we choose a model and fit its parameters, we want to minimize the expected loss, which will be a trade-off between the bias and variance. For simple models we have high bias and low variance and for most complex models we have low bias and high variance. Over-fitting corresponds to the case where the variance is too high, whereas under-fitting is when the bias is too high.

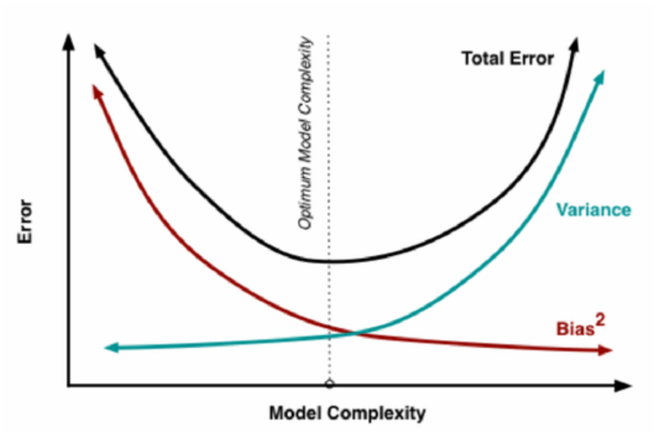


Figure 7: Bias-Variance Trade-off (from Bias Versus Variance 2019 [11])

Figure 7 shows the trade-off between the bias and the variance, with Model Complexity in x-axis which corresponds to the number of parameters for a model. When the Variance is low we have an high $Bias^2$ and inversely when Variance is high we have a low $Bias^2$. The optimum model complexity is when the total Error (Variance + $Bias^2$) is minimized.

We will next give an overview of different basic linear methods, and then explain techniques that we will use for feature selection. (X, Y) will denote our dataset with X our input variables and Y the corresponding target variables. All the code and models are in Python and the different libraries used will be mentioned.

4.4 Ordinary Least Squares

Ordinary least squares (OLS) is a basic model of regression. It fits a linear model with coefficients $w = (w_1, ..., w_p)$ to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. Mathematically it solves a problem of the form:

$$\min_w ||y - Xw||_2^2,$$

where $||.||_2^2$ is the squared L^2-norm.

Theorem 4.10. *The solution of the OLS is given by:*

$$\hat{w} = (X^T X)^{-1} X^T Y.$$

Proof. The normal equations can be derived directly from a matrix representation of the problem as follows. The objective is to minimize

$$\mathrm{S}(\boldsymbol{w}) = \left\|\mathbf{y} - \mathbf{X}\boldsymbol{w}\right\|^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{w})^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{w}) = \mathbf{y}^{\mathrm{T}}\mathbf{y} - \boldsymbol{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} - \mathbf{y}^{\mathrm{T}}\mathbf{X}\boldsymbol{w} + \boldsymbol{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{w}.$$

Here $(\boldsymbol{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y})^{\mathrm{T}} = \mathbf{y}^{\mathrm{T}}\mathbf{X}\boldsymbol{w}$ has the dimension 1x1 (the number of columns of $\mathbf{y}$) , so it is a scalar and equal to its own transpose, hence $\boldsymbol{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} = \mathbf{y}^{\mathrm{T}}\mathbf{X}\boldsymbol{w}$ and the quantity to minimize becomes

$$\mathrm{S}(\boldsymbol{w}) = \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2\boldsymbol{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{y} + \boldsymbol{w}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{w}.$$

Differentiating this with respect to $\boldsymbol{w}$ and equating to zero to satisfy the first-order conditions gives

$$-\mathbf{X}^{\mathrm{T}}\mathbf{y} + (\mathbf{X}^{\mathrm{T}}\mathbf{X})\boldsymbol{w} = 0, ,$$

which is equivalent to the above-given normal equations. A sufficient condition for satisfaction of the second-order conditions for a minimum is that $\mathbf{X}$ have full column rank, in which case $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is positive definite.

□

There are six classical OLS assumptions for linear regression to produce the best estimates. Below are these assumptions:

- The regression model is linear in the coefficients and the error term.
- The error term has a population mean of zero.
- All independent variables are uncorrelated with the error term.
- Observations of the error term are uncorrelated with each other.
- The error term has a constant variance.
- No independent variable is a perfect linear function of other explanatory variables.

When your linear regression model satisfies those assumptions, the procedure generates unbiased coefficient estimates that tend to be relatively close to the true population values (minimum variance). In fact, the Gauss-Markov theorem states that OLS produces estimates that are better than estimates from all other linear model estimation methods when the assumptions hold true. In [8], we can find a proof of the Gauss-Markov theorem and some further interpretations about it.

Remark 4.11. Statistical tests can be perform to evaluate the significance of the coefficients.

The two main advantages of this model are its low computation time and that it does not have parameters to tune, making it very practical for parameters selection or features selection when constructing a model.

When features are correlated and the columns of the design matrix X have an approximate linear dependence, the design matrix becomes close to singular and as a result, the least-squares estimate becomes highly sensitive to random errors in the observed target, producing a large variance. This situation of multicollinearity can arise, for example, when data are collected without an experimental design.
To solve this problem we will consider other models where we add a regularization term to the minimization problem.

We are using the Python's OLS implementation **s**klearn.linear_models.LinearRegression.

4.5 Ridge

Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients. The ridge coefficients minimize a penalized residual sum of squares:

$$\min_{w} ||Xw - y||_2^2 + \alpha ||w||_2^2,$$

The complexity parameter $\alpha \geq 0$ controls the amount of shrinkage: the larger the value of α , the greater the amount of shrinkage and thus the coefficients become more robust to collinearity.

Theorem 4.12. *The solution of Ridge is given by:*

$$\hat{w} = (X^T X + \alpha I_N)^{-1} X^T Y,$$

with N the number of variables used for the model and I_N the N-dimensional identity matrix.

Proof. The result can be obtained as before using matrix calculus. □

Remark 4.13. The assumptions are the same as those used in OLS: linearity, constant variance (no outliers), and independence.

Increasing α will increase the bias of our prediction by reducing the impact of the coefficients but decrease the variance. So in order to find a good trade-off between our bias and variance, we have to select a good α , which can be done by cross-validation.

We have that $\lim_{\alpha \to \infty} w(\alpha) = 0$, but the coefficients will rarely reach exactly zero, so ridge does not perform any feature selection. Other models can be used to perform selection as we will see in the next section.

We are using the Python's Ridge implementation sklearn.linear_model.Ridge.

4.6 Lasso

The Lasso is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer non-zero coefficients, effectively reducing the number of features upon which the given solution is dependent. For this reason Lasso and its variants are fundamental to the field of compressed sensing.

Mathematically, it consists of a linear model with an added regularization term. The objective function to minimize is:

$$\min_{w} ||Xw - y||_2^2 + \alpha ||w||_1,$$

The lasso estimate thus solves the minimization of the least-squares penalty with $\alpha ||w||_1$ added, where α is a constant and is a constant and $||w||_1$ is the l_1-norm of the coefficient vector. This latter constraint makes the solutions nonlinear in the y_i, and there is no closed form expression as in OLS or Ridge regression. In [16] more details can be found about it, and on how to get the Lasso estimate.

We will see in the next part how Lasso can be used to do features selection.

We are using the Python's Lasso implementation sklearn.linear_model.Lasso.

4.7 Feature Selection

One of the most important parts when carrying out machine learning is the feature selection. The features that we use in our machine learning models have a huge impact on the performance. Using irrelevant features can negatively impact the performance so it has to be done carefully. Different methods exist and in general to get the best results we have to use a combination of those methods. We will describe below some methods that we use.

Correlation Matrix: Correlation allows us to see how features are related to each other or the target variable. Correlation can be positive (an increase in one variable return to increase in the other) or negative (increase in one variable decreases the other).

When comparing the correlation of one feature to another, the idea is to choose a threshold (depending of the problem we are working on) and drop one of the features if the absolute value of the correlation is above the threshold (if the features are too correlated).

When comparing the correlation of the features to the target, the idea is to choose a second threshold and keep the features for which the absolute values of their correlations with the target is above the threshold.

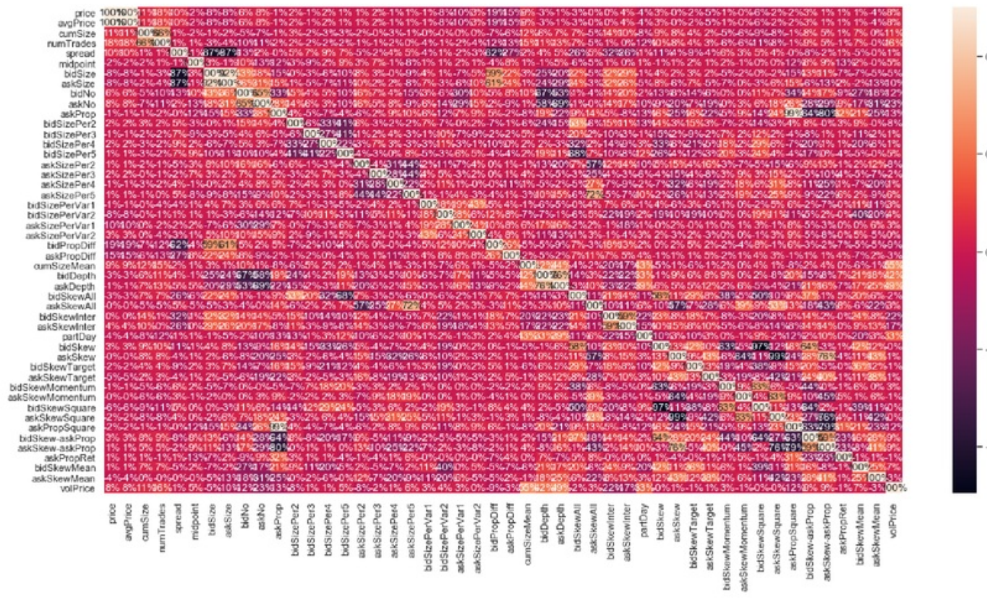


Figure 8: Correlation Matrix with Heatmap

In Figure 8 we see the Heatmap of the features used for the shape of the order book. The number of features here is important, so instead of doing the selection described above manually, we have implemented an algorithm that does the selection automatically.

We are using the Python's Heatmap implementation seaborn.heatmap.

Lasso: To have a fast overview of the features that are relevant, we can apply Lasso on our data without doing any features selection and return the list of coefficients. he features for which the coefficients are non-zero are then relevant. The problem with this method is that Lasso is in general to hard and penalize the features to much, so some features that are relevant can have for coefficients zero.

Lasso alone is not enough to give the best results and so it needs to be used in conjunction with other methods.

See [10] for an alternative solution to feature selection using Lasso. Satoshi Hara and Takanori Maehara propose an algorithm for enumerating solutions to the Lasso regression problem, not only the one global optimum. The algorithm outputs many possible feature sets for human inspection, possibly including relevant features not selected by the Lasso.

maximizing the score: We create a subset of features and add the features that give the best scores for the OLS. We add features as long as the score (R^2-score) is improving.

Algorithm 2: Features selection algorithm by maximization

```
Input: Features, InitialSet, X, Y
Result: BestFeatures
initialization;
R1, Features, SelectFeatures = BestOLS(InitialSet,Features,X,Y);
R2 = $-\infty$;
while $R1 - R2 > 0$ do
    R2 = R1
    R1, Features, SelectFeatures = BestOLS([SelectFeatures],Features,X,Y);
end
```

In the algorithm above, Features is the list of all the features, InitialSet is an initial set of features already selected for our model, X are the values of the features and Y of the target.
The score used here is not specified but in 6.3 we will use the R^2-score.
BestOLS takes as input a subset L of features already selected, a list of features F we want to test and X and Y.
It returns a subset Features where we add to L the feature of F which maximize (or minimize) the score when using an OLS, with the associated score R1 and the updated list of features F.

The scheme that we use when doing features selection is to use the correlation matrix to do a first selection of the features, then use Lasso to create the initial subset of features in the features selection algorithm and finally used the algorithm to complete the initial subset of features.

All the above methods are used on the training set, or a subset of the training set in order to avoid over-fitting. We will see in the next section some methods very useful when doing features selection and that help to avoid over-fitting.

4.8 Cross-Validation

Cross-validation (CV) is a popular technique for tuning hyperparameters and producing robust measurements of model performance. Hyperparameters for our models are the alpha from Ridge and Lasso. First, we split the dataset into a subset called the training set, and another subset called the test set. If any parameters need to be tuned, we split the training set into a training subset and a validation set. The model is trained on the training subset and the parameters that minimize error on the validation set are chosen. Finally, the model is trained on the full training set using the chosen parameters, and the error on the test set is recorded.

Two of the most common types of cross-validation are k-fold cross-validation and Nested cross-validation. K-fold cross-validation has a single parameter called k that refers to the number of groups that a given data sample is to be split into.

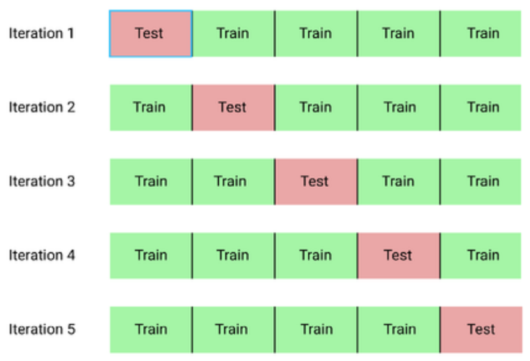


Figure 9: K-fold cross-validation (from Cross Validation Explained: Evaluating estimator performance 2018 [12])

In Figure 9 we see the data is split when using K-fold cross-validation. All the steps are as follow :

1. Shuffle the dataset randomly (Optional).
2. Split the dataset into k groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

The problem with time series data is that we need to take care of how we are splitting the data because, we have to avoid using the future to predict the past, to simulate the real world forecasting environment. So we have to exclude k-fold cross-validation. Instead, we use a technique similar to Nested cross-validation, walk-forward optimization.

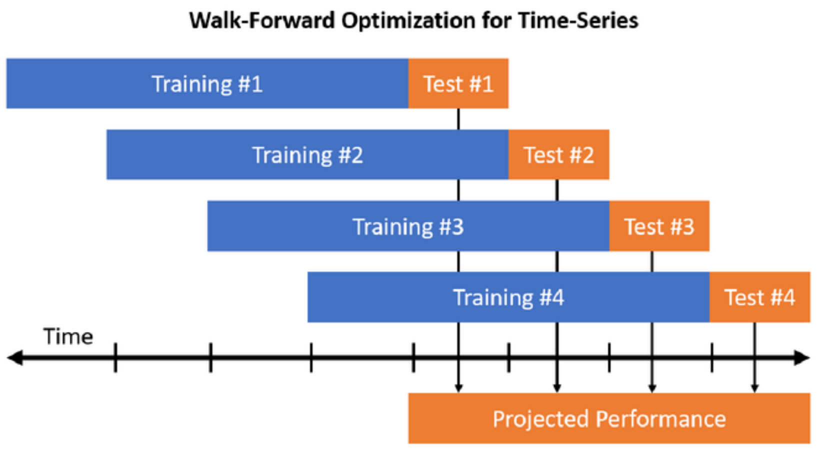


Figure 10: Walk-Forward Optimization

Walk-forward optimization contains an outer loop for error estimation and an inner loop for parameter tuning. The concept for walk-forward testing is similar to using "in-sample" and "out-of-sample" testing periods. Suppose we have twenty years of data, instead of optimizing on sixteen years of data and using the last four years of data for testing, the optimization is done across eight years and the system is tested on the ninth. Once this test is completed, the whole time window is moved forward one year and run the test again. The optimum set of parameters for each of the 8-year windows and is used as the set of parameters for the next year. The time window is moved forward one year and the test is run on the next year until all the years in the data series have been tested. When the system performance is evaluated, all the one-year windows are consolidated to compose the out-of-sample periods for each of the optimal windows. The out-of-sample performance is used to judge how good the system is.

5 Price impact of order book events

High-frequency records of trades and quotes stimulated and allowed the research of correlation between the order flow and the price movements in order-driven-markets. Impact of orders on prices is of particular importance, which is why we conduct a study of the instantaneous impact of order book events (market orders, limit orders and cancellations) on different stock prices.

See also the pioneering work on the optimal execution problem from Bertsimas and Lo [18], they provide an intuitive optimal strategy to minimize the purchasing cost. See also Almgren and Chriss [19], where the performance criterion of the execution consists not only of the expected revenue but also a penalty for the uncertainty of the revenue. In these papers, the market impact is defined by two components: the temporary impact and the permanent impact.

5.1 Context

Rama Cont, Arseniy Kukanov and Sasha Stoikov [2] have shown that instantaneous effects on prices may be modelled through the order flow imbalance (OFI). OFI represents the net order flow at the best bid and ask and tracks changes in the size of the bid and ask queues by:

- increasing every time the bid size increases, the ask size decreases or the bid/ask prices increase,
- decreases every time the bid size decreases, the ask size increases or the bid/ask prices

The model used is a linear regression between the price change and the OFI, with a single coefficient that we will call the impact coefficient. For our analysis we are using the quotes of ten different stocks from the FTSE 100 stock index for the month of June 2019.

We will see that the average R^2-score between the different stocks are very similar but with a lot of variation intraday due to some events, such as the mid-day auction. Overall when dropping those kinds of events, the R^2-score are robust. Order flows deeper in the order book do not contribute as much to prices changes as we may have thought.

We will also see that the variability of the impact coefficient can be explained by the market depth, and more precisely that there exists an inverse relation between the two variables.

5.2 Price impact model

Consider a time interval $[t_{k-1}, t_k]$ and denote by L_k^b, C_k^b the total size of limit orders that arrive to, and are cancelled from, the current best bid during that time interval. Also denote by M_k^b the total size of market buy orders that arrive to the current best ask, and by P_k^b the bid price at time

t_k.

L_k^s, C_k^s, M_k^s and L_k^s are defined analogously for sell orders.

We consider mid-price changes normalized by tick size: $\frac{P_k^b+P_k^s}{2\delta}$, where δ is the tick size.
The OFI_k is then defined by:

$$OFI_k = L_k^b - C_k^b - M_k^s - L_k^s + C_k^s + M_k^b \tag{5.1}$$

In Equation (5.1), OFI_k can be seen as the imbalance between the supply and demand. If there are more buy limit and marketable orders than sell ones (without taking cancellations into account), then the OFI_k is positive and the price of the stock is more susceptible to increase. We see that there is a positive correlation between the OFI_k and the share price change.

We assume that there is a noisy relation between price changes and OFI which holds locally for short intervals of time $[t_{k-1}, t_k] \subset [T_{i-1}, T_i]$ where $[T_{i-1}, T_i]$ are longer intervals.

$$\Delta P_{k,i} = \beta_i OFI_{k,i} + \epsilon_{k,i}$$

where:

β_i is a price impact coefficient for an i-th time interval;

$\epsilon_{k,i}$ is a noise term summarizing influences of other factors (e.g. deeper levels of the order book).

Remark 5.1. We fit a different coefficient β_i for each index i because of intraday seasonality effects.

5.3 Data

Our data set consists of one calendar month of trades and quotes for 10 stocks of the FTSE 100. Every observation consists of the prices and sizes of the five first levels. We will denote for the first level by P^b and q^b the best bid price and size respectively, P^s and q^s for the best ask price and size.

We use two uniform time grids $\{T_0, ..., T_I\}$ and $\{t_{0,0}, ..., t_{I,K}\}$ with time steps $T_i - T_{i-1} = M$ in minutes and $t_{i,k} - t_{i,k-1} = S$ in seconds. We will try different values for M and S as part of an optimization procedure, but the ones that seem to give the best results are M = 30 minutes and S = 10 seconds. We enumerate the n-th observations of the quotes for each day by n. Using only the quotes, we can compute the OFI define by (5.1) in the following way :

First, compute the difference between each consecutive observation:

$$e_n = q_n^b \mathbb{1}_{\{P_n^b \geq P_{n-1}^b\}} - q_{n-1}^b \mathbb{1}_{\{P_n^b \leq P_{n-1}^b\}} - q_n^s \mathbb{1}_{\{P_n^s \leq P_{n-1}^s\}} + q_{n-1}^s \mathbb{1}_{\{P_n^s \geq P_{n-1}^s\}} \tag{5.2}$$

In Equation (5.2), the variables e_n are contributions of order book events to supply and demand. This relation allows the computation of OFI_k using only the quotes and not the marketable orders and cancellation orders. When a buy limit order arrives, q^b increases but P^b remains the same, leading to $e_n = q_n^b - q_{n-1}^b$ which corresponds to the size of that order. IF q^b decreases, we have $e_n = q_n^b - q_{n-1}^b$ representing the size of a marketable sell order or buy order cancellation. Now if P^b changes, then $e_n = q_n^b$ or $e_n = -q_{n-1}^b$ representing respectively the size of a price-improving order or the last order in the queue that was removed.

Within each $[t_{i,k-1} - t_{i,k}]$, we compute the price changes and order flow imbalances index by k and i.

$$\Delta P_{k,i} = \frac{P^b_{N_{(t_{i,k})}} + P^s_{N_{(t_{i,k})}}}{2\delta} - \frac{P^b_{N_{(t_{i,k-1})}} + P^s_{N_{(t_{i,k-1})}}}{2\delta}$$

$$OFI_{k,i} = \sum_{n=N_{(t_{i,k-1})}+1}^{N_{(t_{i,k})}} e_n$$

with:

$N_{(t_{i,k-1})} + 1$ and $N_{(t_{i,k})}$ are the index of the first and the last order book event in the interval $[t_{i,k-1} - t_{i,k}]$;

$P^b_{N(t_{i,k-1})}$ and $P^b_{N(t_{i,k})}$ (similarly $P^s_{N(t_{i,k-1})}$ and $P^s_{N(t_{i,k})}$) are the bid (ask) price at the beginning and at the end of the interval $[t_{i,k-1} - t_{i,k}]$.

Finally, in order to compare our impact coefficients with the depth, we will estimate for each interval $[T_{i-1} - T_i]$ the depth by averaging the bid/ask queue sizes right before or right after a price change.

$$D_i = \frac{1}{2}\left[\frac{\sum_{n=N_{(T_{i-1})}+1}^{N_{(T_i)}} (q_n^b \mathbb{1}_{\{P_n^b < P_{n-1}^b\}} + q_{n-1}^b \mathbb{1}_{\{P_n^b > P_{n-1}^b\}})}{\sum_{n=N_{(T_{i-1})}+1}^{N_{(T_i)}} \mathbb{1}_{\{P_n^b \neq P_{n-1}^b\}}} + \frac{\sum_{n=N_{(T_{i-1})}+1}^{N_{(T_i)}} (q_n^s \mathbb{1}_{\{P_n^s > P_{n-1}^s\}} + q_{n-1}^s \mathbb{1}_{\{P_n^s < P_{n-1}^s\}})}{\sum_{n=N_{(T_{i-1})}+1}^{N_{(T_i)}} \mathbb{1}_{\{P_n^s \neq P_{n-1}^s\}}}\right] \tag{5.3}$$

In equation 5.3, we compute, for each time interval i, the average of the bid/ask queue sizes after a price change. If there is no price-move on the bid and on the ask during all the time interval we have $D_i = 0$. Otherwise if there is a cancellation order or a marketable sell order (so that the best bid price decreases), we will take the mean over the new best bid size q_n^b. Conversely if there is a best bid price increase due to a buy limit order, we will take the mean over the old best bid size q^b_{n-1}.

To avoid perturbations due to the different auctions of the day, we drop the quotes three minutes after and before the opening and closing auction respectively, and five minutes around the mid-day auction (from 12:00 to 12:05).

5.4 Results

In this section, we are only using Ridge when doing linear regression.

5.4.1 Estimation Results

First, we have to determine which time intervals for the first and second grid give the best results. In order to find this we compute the mean of the R^2-scores on five different stocks across all the days, for five different time intervals.

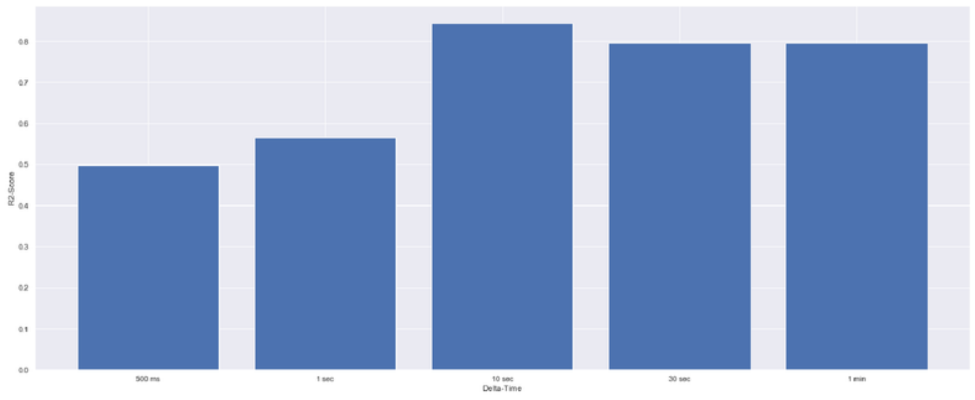


Figure 11: Average R^2-score across time for different Δt on five stocks
Here we are computing the average R^2-score across stocks for five different values of bins interval.

In Figure 11, we see the R^2-score for five different values of S. For each S, we have taken a different value of M in order to make the computations feasible. For example for S = 0.5 seconds we have taken M = 2 minutes, whereas for S = 10 seconds we have taken M = 30 minutes.
We see that S = 10 seconds and M = 30 minutes give the best results, so for the rest of the analysis we will keep these two values.

First, we perform our estimations using only the first level of the limit order book. We are using 30 minutes time intervals for the first grid and 10 seconds for the second one.

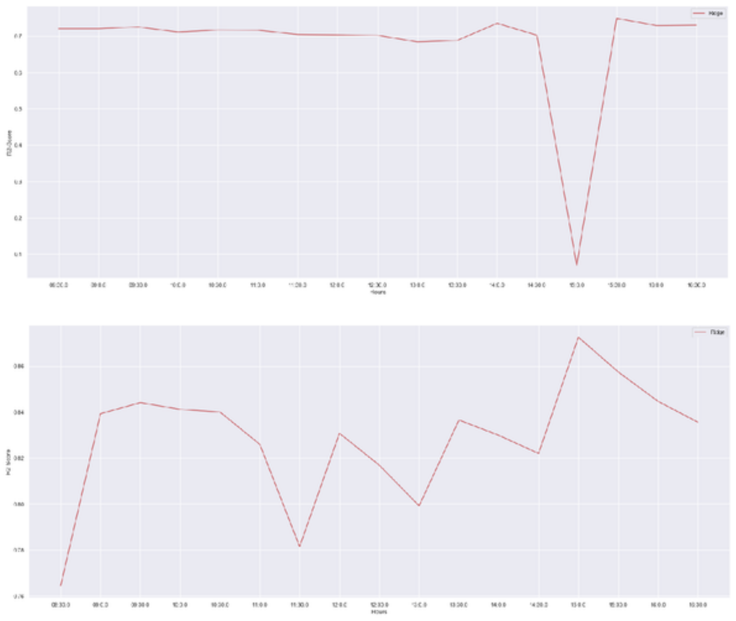


Figure 12: R^2-score for AAL.L (top) and AZN.L (bottom) every half hour using only the first level of the LOB

In Figure 12, we have the R^2-score of our price-change estimations for each thirty-minute interval, for AAL.L (Anglo American public limited company) and AZL.L (AstraZeneca public limited company) using only the first level of the order book. We see that the R^2-score is more or less constant over the different time intervals, except for AAL.L at 15:00. After some research, it was found that the closing auction of the Johannesburg Stock Exchange, as Anglo American plc is based in Johannesburg, has an impact on the estimation of the price changes.

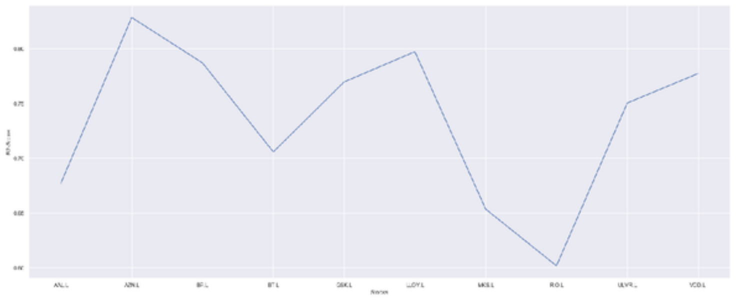


Figure 13: R^2-score by stocks using only the first level

In Figure 13, we can see for each stock, the mean R^2-score for the different time intervals. There are some variations through the different stocks, which can be due to the lack of data, but we see that the model is consistent across stocks.

Ticker (stock symbol)	$\widehat{\beta_i}$	$\widehat{\frac{1}{2D_i}}$	R^2
AAL.L	$3.51e^{-04}$	$5.53e^{-04}$	0.71
ANL.L	$3.21e^{-03}$	$7.10e^{-04}$	0.83
BP.L	$2.08e^{-05}$	$4.78e^{-05}$	0.79
BT.L	$4.22e^{-05}$	$7.42e^{-05}$	0.69
GSK.L	$8.54e^{-05}$	$2.37e^{-04}$	0.76
LLOY.L	$9.05e^{-06}$	$1.01e^{-05}$	0.80
MKS.L	$8.96e^{-05}$	$5.19e^{-05}$	0.70
RIO.L	$9.01e^{-04}$	$1.03e^{-03}$	0.61
ULVR.L	$4.91e^{-04}$	$5.58e^{-04}$	0.75
VOD.L	$1.10e^{-05}$	$3.15e^{-05}$	0.76

Table 2: Relation between price changes and order flow imbalance

$\widehat{\beta_i}$ is the mean of the β across time for each stock and, $\widehat{\frac{1}{2D_i}}$ is the mean of $\frac{1}{2D_i}$ across time for each stock.

Table 2 presents a cross-section of results (averaged across time) for regressions:

$$\Delta P_{k,i} = \beta_i OFI_{k,i} + \epsilon_{k,i}$$

using only the first level of the order book.

Now, we repeat our estimations using the first two levels of the limit order book. We are still using 30 minutes time intervals for the first grid and 10 seconds for the second one. It is 10 % better than before when using only one level, but as we will see it also has the advantage of being more stable.

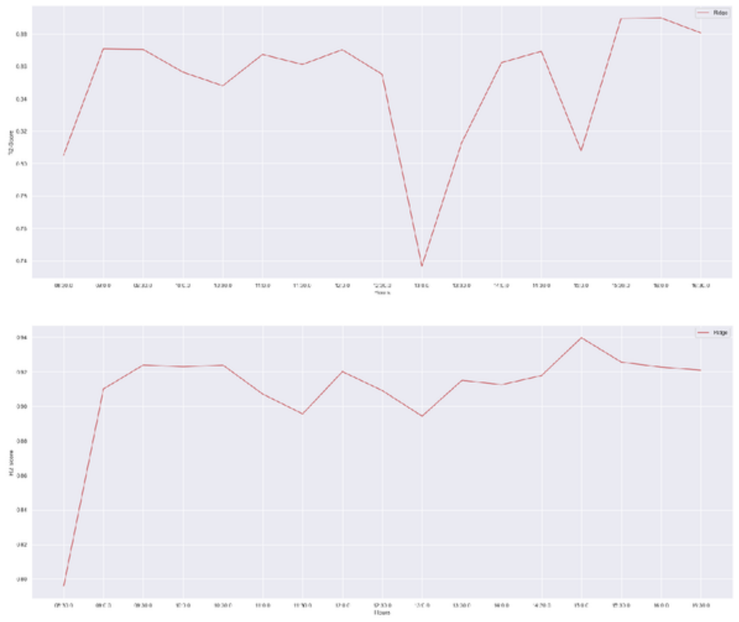


Figure 14: R^2-score for AAL.L (first one) and AZN.L (second one) every half hour using the first two levels of the LOB

In Figure 14, we have the R^2-score of our price change estimations for each thirty minutes interval, for AAL.L (Anglo American plc) and AZL.L (AstraZeneca plc) using the first two levels of the order book. We see that the R^2-score is a little bit higher, and far more stable than when we use only one level of the order book. The drop of R^2-score at 15:00 for AAL.L is not present any more, showing that a two level model captures in a better way the price changes during disturbing events.

5.4.2 Link between Impact coefficients and depths

Recall that we denote by β_i the price impact coefficient of the i-th time interval and by D_i the corresponding depth. Using discussions and results from [2], we will interpret $\frac{1}{2D_i}$ as an implied impact coefficient.

It can be viewed instinctively that the impact coefficient is inversely proportional to the depth of the order book: the higher the depth is, the lower is the impact of the order flow imbalance on the price change. In [2], this relation is shown more concretely using a stylized order book.

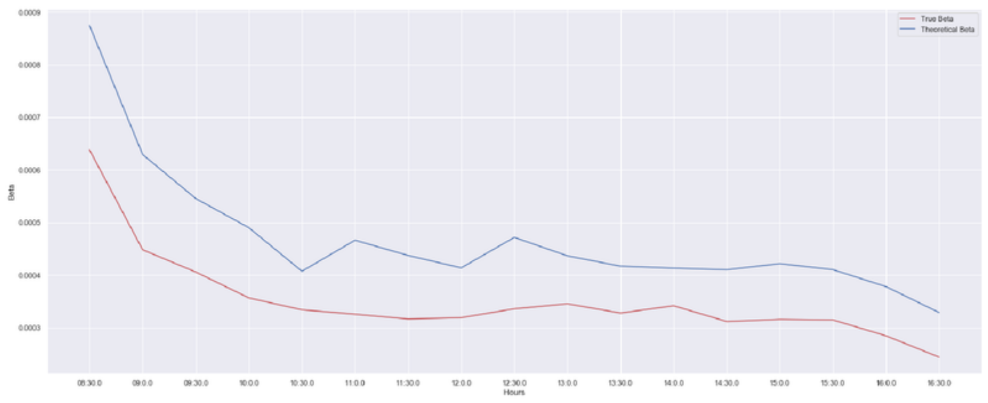


Figure 15: Comparison of impact coefficients and depth for AAL.L

In Figure 15, the red curve corresponds to the mean of all the impact coefficients on the different days, whereas the blue curve corresponds to the mean of the theoretical coefficients, that is to say, the mean of the $\frac{1}{2D_i}$ (this is all for AAL.L). The two curves are not exactly equal but are close and have the same shape. We can also see that between 08:30 and 09:00 the impact coefficient is two times higher than on average, indicating that the market is relatively shallow. In a shallow market, incoming orders can easily affect the mid-price. Moreover, price impact coefficients between 08:30 and 09:00 are five times higher than between 16:00 and 16:30.

5.5 Applications

For the moment, we do not predict the price changes on each interval, indeed to estimate the price change on an interval $[t_{i,k-1} - t_{i,k}]$, we are using all the available information until $t_{i,k}$. So the interest of the previous analysis is not to predict the price changes, but to see that the short term price changes are almost entirely governed by the order flow imbalances.

One of the first applications of this relationship could be to predict the adverse selection engendered by a market order, for example to try to predict the change between the price just before the market order and some milliseconds after it. We can be tempted to use this relationship to predict the future price changes. However, a simple approach, where we used the actual order flow imbalance to predict the price changes of the next time interval gave bad results (negative R^2-score). We can see in Figure 16 below that the R^2-score is around zero across the day. Another approach could be to reduce the time horizon of the prediction, and use a longer time interval for the computation of the order flow imbalance.

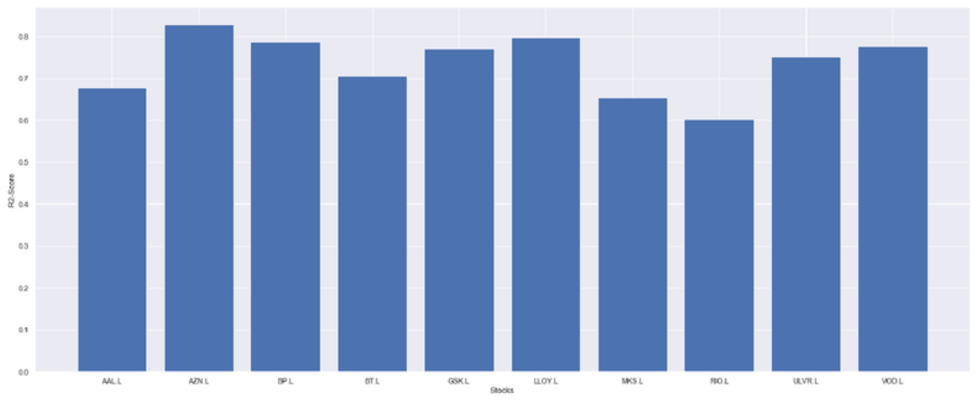


Figure 16: R^2-score by stocks of price changes prediction

In Figure 16, we can see the R^2-score for our simple approach to predict future price changes. We see that the R^2-score is negative and near zero.

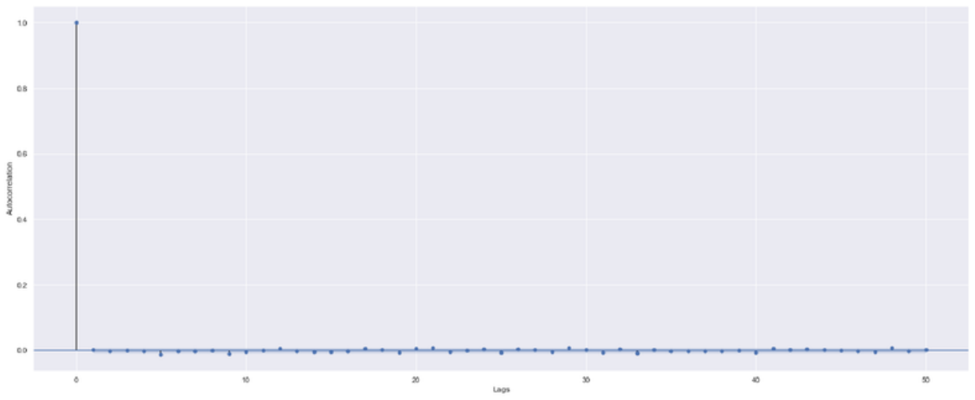


Figure 17: ACF of the mid-price changes $\Delta P_{k,i}$ for AAL.L

As shown in Figure 17, the autocorrelation of the mid-price changes $\Delta P_{k,i}$ at a timescale of 10 seconds is small and quickly vanishes. We get the same results for the autocorrelation of the OFI (can deduce this due to the linear relation between $\Delta P_{k,i}$ and OFI): it explains the bad results of our simple approach above.

6 Shape of the order book

In this section we study the shape of the order book, and we try to predict some characteristics related to the order book, such as the ratio of the first level size, that are relevant to know when buying or selling. Indeed, before trading an asset, it can be useful to predict the future shape of the order book. For instance to know whether the limit orders will be more on the bid side or the ask side, or are more on the first levels or lower levels, can be very relevant to determine when and how to trade.

In general, if not specified when using linear regression, Ridge is used to predict our targets, Lasso is used for the feature selection and OLS is used to find the optimal parameters.

6.1 Data

For my analysis of the shape of the order book, I have a sample of the quotes and trades of twenty-three different stocks from the FTSE 100 for the month of June for each minute of the day. The sizes and number for quotes columns correspond to the median of the sizes during the last minute. For instance, the column corresponding to the size of the first level of the bid, is the median of the first level of the bid during the last minute. We have decided to use the median instead of the mean to keep prices rounded to the nearest tick.

The prices columns for both the quotes and the trades for each minute correspond to the VWAP of each column during the last minute.

The sizes column of the trades correspond to the total number of stocks traded during the minute.

We could have kept the last quotes update for each minute, but when we tried to do predictions on this dataset, it gave very bad results due to the fact that the last quotes update is too random. For example if the quotes update is just after a buy trade, the size of the first ask level will not accurately reflect the shape of the order book during the minute.

Remark 6.1. In Chapter 2 of [14], more details can be found about information-driven bars. Different ways to sample data are presented with justifications of their uses. We have chosen to simply sample by minute bins, but more sophisticated methods exist, as for instance sampling using the volume traded.

The data has been split up as 75% for the training set and 25% for the test set.

6.2 Choice of targets

Before trying to predict anything, it is important to choose a good target. A good one will involve a trade-off between its practicability and the feasibility of the prediction.
Our first goal is to determine how the order book shape affects the future order book shape. First it is important to see what it looks like.

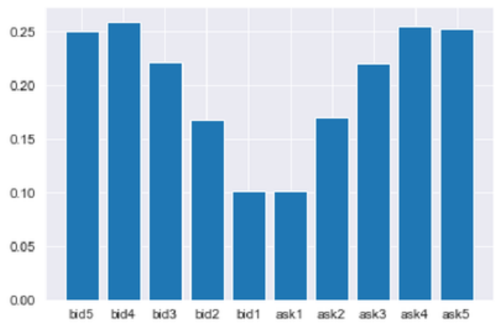


Figure 18: Limit order book shape distribution for the first five levels

In Figure 18, we can see the distribution of the first five levels of the limit order book for the month of June for AAL.L, with Bid_1 and Ask_1 corresponding to the first level of the bid and the ask. We have computed for each level i of the bid, the ratio of the volume of level i to the total volume of the bid, and we have done the same for the ask. For other stocks, the shape of the order book is very similar.

To have a good view of the order book, two important features are the skew and the depth (both for the bid and the ask) of the order book.
We first focus on the skew. As we can see on the figure 18, the ask side is naturally left skewed whereas the bid side is right skewed. Therefore, it is important to define a target that takes this into account. A way to do it is is to compute the deviation of the distribution from the mean.

We want to have something positive when it is right skewed and negative otherwise. We define:

$$SkewBid = \frac{(\mathbb{1}_{\{i<=2\}} - \mathbb{1}_{\{i>=4\}}) \sum_{i=1}^{5} (Bid_i - \widehat{Bid_i})^2}{\widehat{Bid_i}^2},$$

where:

Bid_i is the ratio of the level i of the bid to the total volume of the bid;
$\widehat{Bid_i}$ is the mean of the ratio over the last N minutes (N to be determined).

The term $\mathbb{1}_{\{i<=2\}} - \mathbb{1}_{\{i>=4\}}$ is here in order to have a positive term when this is right skew; for the ask it will be $\mathbb{1}_{\{i>=4\}} - \mathbb{1}_{\{i<=2\}}$.

This definition of the skew is very natural and easily understandable as it is only a one parameter skew, but the results of the prediction are very bad (the R^2-score is near zero).
Instead we choose to predict each Bid_i and Ask_i separately, so we end up with a five parameters skew. In practice to know the ratio of the first level is enough as it is the one that has the most impact on the stock price return.
For the next section, we will consider our skew targets as the ratio of the first level of the bid and the ask. But the work down below can also be done for the next level of the order book.

Now we have to define the depth, which has to be complementary to the skew. Our definition of the skew allows only to know if the liquidity is more on the first levels or on the last, so we need something that allows to know if there is more liquidity available than usual on each side.
We want to define different categories of depth in order to use classification algorithms, this is why we use the following definition of the depth.

$$DepthBid = 1 + \sum_{i=1}^{5} (\mathbb{1}_{\{\frac{\widehat{CumSize}}{2} > \sum_{j=1}^{i} bidSize_i\}}),$$

where:

CumSize is the total size in notional traded during the last bin;
$\widehat{CumSize}$ is the rolling mean of CumSize on N bins;
$bidSize_i$ is the total size in notional on level i of the bid.

With this definition we end with 6 categories of depths, 1 if there is a lot of liquidity on the first level, through 6 if we need to go deeper in the order book to find liquidity.

6.3 Feature Selection

Here we present the results of our selection using the methods described in section 4.7 (combining correlation matrix, Lasso and our algorithm maximizing the score). The selection of the features below has been done on the training set, keeping the test set to evaluate for the back-testing.

First we define the main features derived from the original features that we will use. In order to compare the stocks between them, we convert the size in number of shares into size in notional ($price \times size$).
Below we detail the features for the ask side, but we have done similar computation for the bid side.

Remark 6.2. The parameters N and Nb below, which give the number of rows where we compute the mean and variance, are determined using grid-search.

$askSize_i$: Size for level i of the ask in notional.

$askSize$: Total size of the ask in notional.

$askProp$: Ratio of the total sizes of the ask to the total size of the bid and ask, $askProp = \frac{askSize}{bidSize+askSize}$
Note that to compute $bidProp$ = 1 - $askProp$.

$askNo_i$: Number of limit orders for level i of the ask.

$askSizePer_i$: ratio of $askSize_i$ to the total ask size, $askSizePer_i = \frac{askSize_i}{\sum_{j=1}^{5} askSize_j}$.

$askSizePerMean_i$: Mean of the last N $askSizePer_i$.

$askSizePerVar_i$: Variance of the last N $askSizePer_i$.

$midPoint$: The midpoint.

$midPointRet$: log-return return of the mid-point $midPointRet = \log(midPoint) - \log(midPoint.shift)$.

$volPrice$: The volatility of the mid-point return, we compute it using the standard deviation of the mid-point return on the Nb last rows.

$askSkew$: $\log(askSizePer_1)$.

$askSkewMean$: Mean of the last N $askSkew$.

$askSkewInter$: Mean of the ask skew for all the stocks over the last five minutes.

And finally the target is :

$askSkewTarget$: $askSkew$ shifted by one row.

We take the logarithm of $bidSizePer_1$ and $askSizePer_1$ in order to make our targets stationary. $bidSizePer_1$ and $askSizePer_1$ will be dropped as they are strongly correlated to their logarithms.

For each stock we have done a selection of features, so each stock has its own features.

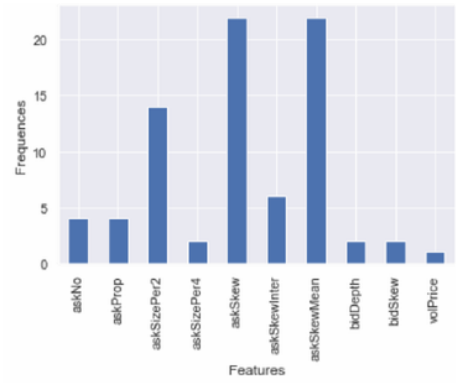


Figure 19: Frequencies of the features selected for the prediction of the ask skew

In Figure 19 we can see the frequencies of the features selected for the ask skew prediction, for 23 stocks. We see that $askSkew$ is selected for all the stocks so the last skew size is of particular importance, $askSkewMean$ appears also to be very important. We see that the ratio of the second level is also selected around half the time, so it plays an important role in the next minute first level ratio. In the end only three features seems to play an important role in the prediction.

Remark 6.3. $askSkewInter$ seems also to play an important role for eight stocks, so for those stocks the moves of the skews on the other stocks have an impact on their skews.

The problem with doing a selection of features for each stock, is that features that are relevant for all the stocks may not be selected for some stocks due to over-fitting of the data on the training set. An alternative method of selection is to "freeze" some features that seems relevant for all the stocks in our model, that is to say to force our model to take some features. Here the features that seems to be relevant from the Figure 19 are $askSkewMean$, $askSkew$ and $askSizePer2$.

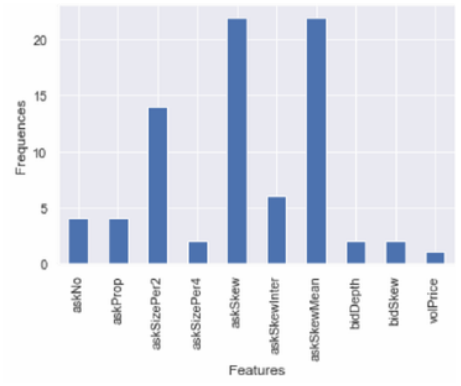


Figure 20: Frequencies of the features selected for the prediction of the ask skew, freezing some features

In Figure 20 we can see the frequencies of the features selected for the ask skew prediction for 23 stocks, this time freezing three features $askSkewMean$, $askSkew$ and $askSizePer2$. We see that now more features are considered than before (14 against 10 when no freezing).

For the first method when no features were frozen, we get a test-set R^2-score on average through all the stocks of 0.181. When doing the second method, freezing $askSkewMean$, $askSkew$ and $askSizePer2$, we get an R^2-score of 0.179. The difference in R^2-scores is very low between the two methods, and (not as expected) the first method gives slightly better results.

As the difference is very low it is hard to draw any conclusion, but we can say that for this set of features and stocks, the method described in 4.7 already avoids over-fitting, and that there is a difference of relevant features for each stock: $askSizePer2$ is important for some stocks and not for others. Maybe if the set of features and of stocks was bigger the difference would be more significant.

In the next section, we will describe our results using the features created and selected in this section.

6.4 Results

Here we will see the the results of our R^2-score for the bid skew predictions. For a benchmark to compare our model against, we just use the current $askSkewMean$ as the prediction of the next skew. The number of rows on which we compute the mean is optimized using a grid search on the training set data.
The scores below have been computed on the test set and we are using Ridge to predict our targets.

We will first fit only one model for all the hours of the days for each stock (one set of parameters for each stock), then we will fit multiple models for the different intervals of hours of the days.

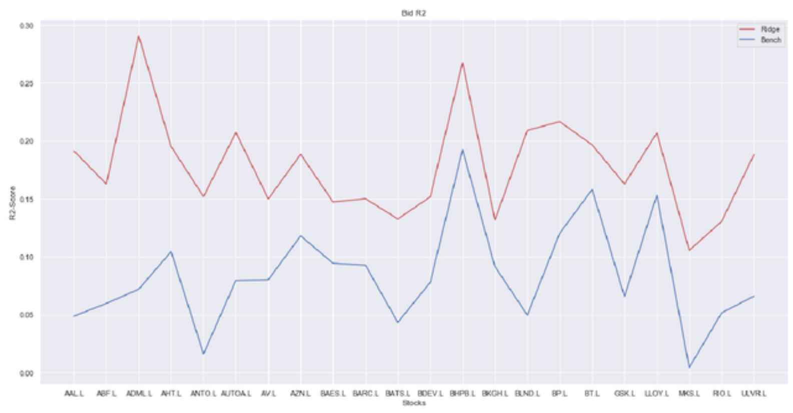


Figure 21: R^2-score for the prediction of the bid skew for 23 different stocks for the benchmark (blue) and our model (red).

In Figure 21 we can see the different R^2-scores, when only one regression is performed for each of the 23 stocks. The R^2-scores for our models (in red) is quite stable and its R^2-score is around twice as much as the benchmark R^2-score.

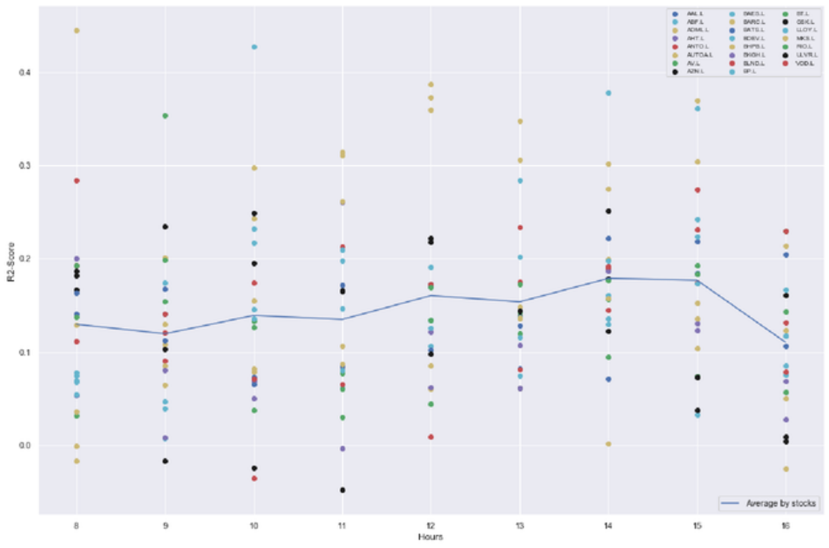


Figure 22: R^2-score for the prediction of the bid skew with one model by stock

In Figure 22 we can see the different R^2-scores for each hour interval, when only one regression is performed for each of the 23 stocks. The numbers of the x-axis correspond to the next hour range interval, so 8 corresponds to the time interval 08:00 to 09:00 and 16 to the time interval 16:00 to 16:30. We see that there is an intraday difference for the R^2-scores: it is easier to predict the skew after 12:00 than before.

The mean of R^2-scores on all the stocks and hours is 0.181.

Remark 6.4. We are not computing the average of the R^2-scores by time intervals, since there is not the same number of events for each interval (due to the drop of some rows for example). Instead we are putting a weight in front of each R^2-score :

$$Mean = \sum_{i=1}^{N_{interval}} \frac{NumberEventsInterval_i}{TotalNumberEvents}$$

The differences of R^2-scores during the day can make us believe that our predictions are working differently in function of the hours of the day, this is why we will try to fit a model for each hours interval.

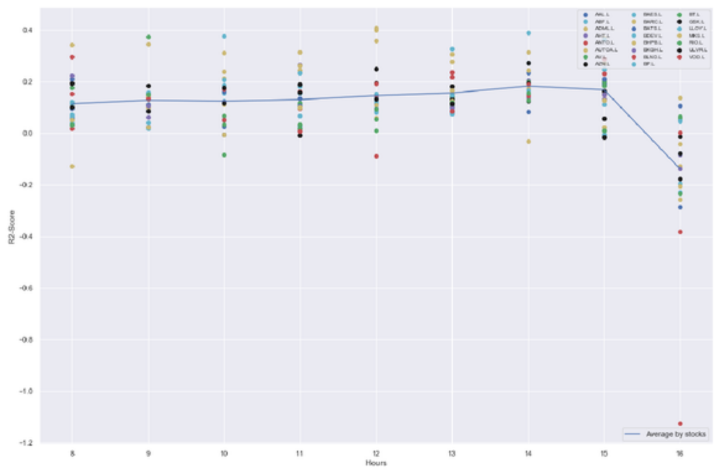


Figure 23: R^2-score for the prediction of the bid skew with multiple regressions are performed for each stock

In Figure 23 we can see the different R^2-scores for each interval of hours, when regressions are performed for each of the 23 stocks (9 in total). We see that there R^2-scores are more constant during the day than in figure 22, except for 16:00 to 16:30 where there is a big drop. This drop can be explained by the lack of data for this particular interval as it is only half an hour.
The mean of R^2-scores on all the stocks and hours is 0.140, when giving the same weights for each hours interval. So it is less than when we are doing one regression for each stock, this is maybe also due to a lack of data as we are now splitting our dataset in nine.

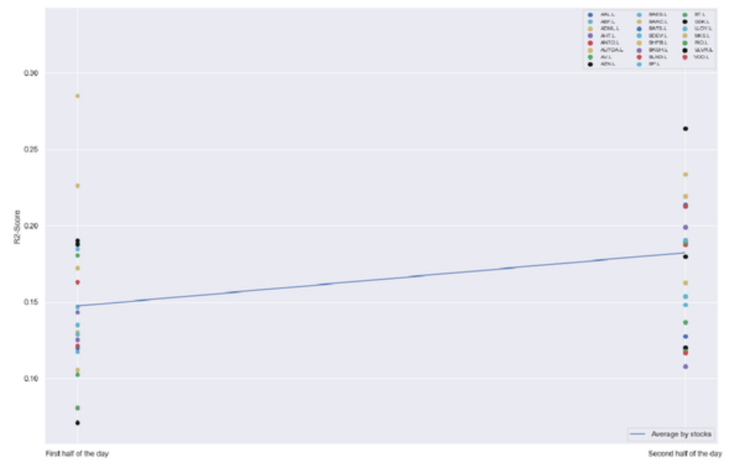


Figure 24: R^2-score for the prediction of the bid skew

In Figure 24 we can see the different R^2-scores for each interval of hours, when we are splitting the days in two: before and after 12:00, so now we are fitting two models. We see that the R^2-score of the second half of the day is bigger than the first one.
The mean of R^2-scores on all the stocks and hours is 0.174.

6.5 Further Research

Further things can be done in order to improve the results. First when computing $askSkewInter$ and $bidSkewInter$, instead of considering the average of the last skews through all the stocks, we could do it by groups of stocks where the skews are correlated. This can be relevant if the changes of some stocks skews drive the changes of the skews on other stocks (if they are correlated), but twenty-three stocks is not enough to find correlations so we will need more data.

Getting more data for the same stock could be very helpful, in order to complete our analysis on doing different regressions across the day as we are not able to draw any conclusion at the moment. We can also get data with different bins size, e.g. 5, 10 or 30 seconds, as one minute bin size has been chosen somewhat arbitrarily. We can expect the targets to be less predictable, as the means over shorter intervals will be more noisy.

Also other targets could be found to complete the analysis of the shape of the order book, as for example we could try to predict the ratio of the bid and the ask (the features $askProp$ shifted by one). It will be a good complement to the prediction of the skew, but we will have to create some new features more relevant for this target.

The uses of these predictions are multiple. For the skew, knowing the ratio of the first level for the next minute can be helpful when deciding where and when putting an order and to choose what kind of order is the best: if I put a limit order, will my order be executed before there is a price change, or is it better to cross the spread ?

Conclusion

Throughout Section 3, we saw the difficulties we can encounter when working with high frequency data, in particular iceberg orders data which is already a hard problem. In spite of that, we were able to obtain the distribution of displayed on total volume ratio, where we saw that there is still an important human impact when choosing it.

In Section 5, we have introduced order flow imbalance, a variable that cumulates the sizes of order book events, and studied its impact on the mid-price price changes. Moreover, we have provided empirical evidence for a linear relation between price changes and order flow imbalance for individual stocks. We saw that using only the quotes to compute the OFI explains the price changes with an average R^2-score of 85% for bins intervals of 10 seconds. The impact coefficients which are the coefficients of the linear regression are inversely proportional to the depth of the order book, with an impact five time bigger in the morning than at the end of the day.

In the last section, we have seen a general application of different Machine Learning methods where we predict the skew of the order book, which is defined as the proportion of the first level of the bid and of the ask. We have seen in particular how important can be the choice of targets and the feature selection on our results. We also saw the importance of the choice of the number of regressions a day for each stock, seeing that one regression a day gave the best results, but multiple regressions a day could give better ones with more data available.

In general, we have seen how difficult it can be to use financial data and that we should not expect all the time to get results. Moreover, we have seen that there is no need to use complex Machine Learning methods to get results and that in particular linear regression (including OLS, Ridge and Lasso) can be very helpful for finding the relationship between complex financial features.

References

[1] Martin D.Gould, Mason A. Porter, Stacy Williams, Mark Mcdonald, Daniel J.Fenn, and Sam D.Howison
Limit Order Books, 2012.

[2] Rama Cont, Arseniy Kukanov and Sasha Stoikov.
The price impact of order book events, 2012.

[3] Ciamac C. Moallemi and Kai Yuan.
A model for Queue Position Valuation in a Limit Order Book, 2017.

[4] Frey Stefan; Sandas Patrik.
The impact of iceberg orders in limit order books, 2009.

[5] Hugh L. Christensen, Robert Woodmansey.
Prediction of Hidden Liquidity in the Limit Order Book of GLOBEX Futures, 2013.

[6] Algorithmia.
Introduction to Loss Functions, 2018.
https://blog.algorithmia.com/introduction-to-loss-functions/

[7] Jim Frost.
How To Interpret R-squared in Regression Analysis, 2018.
https://statisticsbyjim.com/regression/interpret-r-squared-regression/

[8] Leigh J. Halliwell, FCAS, MAAA.
The Gauss-Markov Theorem: Beyond the BLUE, 2015.

[9] Joseph E.Gonzalez
Linear Regression and the Bias Variance Tradeoff.

[10] Satoshi Hara, Takanori Maehara
Enumerate Lasso Solutions for Feature Selection.

[11] Sydney Firmin.
Bias Versus Variance, 2019.
https://community.alteryx.com/t5/Data-Science-Blog/Bias-Versus-Variance/ba-p/351862

[12] Raheel Shaikh.
Cross Validation Explained: Evaluating estimator performance, 2018.
https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85

[13] Isabelle Guyon

A scaling law for the validation-set training-set size ratio.

[14] Marcos Lopez de Prado.

Advances in Financial Machine Learning, 2018.

[15] Hastie, Tibshirani, Friedman.

Elements of Statistical Learning. Springer, 2008.

[16] Robert Tibshirani.

Regression Shrinkage and selection via the lasso.

[17] Hastie, Tibshirani, Friedman.

Elements of Statistical Learning. Springer, 2008.

[18] Dimitris Bertsimas and Andrew W. Lo.

Optimal control of execution costs, Journal of Financial Markets 1, 1998, no. 1, 150.

[19] Robert Almgren and Neil Chriss.

Optimal execution of portfolio transactions, Journal of Risk, 2001, no. Kyle 1985, 539..

Iceberg Orders and Shape of the Limit Order Book

GRADEMARK REPORT

FINAL GRADE

/0

GENERAL COMMENTS

Instructor

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

PAGE 30

PAGE 31

PAGE 32

PAGE 33

PAGE 34

PAGE 35

PAGE 36

PAGE 37

PAGE 38

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53

PAGE 54

PAGE 55

PAGE 56