POLITECNICO
MILANO 1863

School of Industrial and Information Engineering
MMF - Quantitative Finance

Master's Thesis

A modular model for Limit Order Book forecasting

Supervisor:
Prof. Daniele Marazzina

Master's Thesis of:
Edoardo Verpelli

Academic Year 2020/2021

Abstract

In the financial industry, as in other fields, the usability of models largely depends on the flexibility they offer in a production environment, and the ability to adapt to scenarios that were not necessarily intended for the model to handle. In this work, we present a modular model that offers great flexibility and allows to be scaled according to the needs of a particular situation. The model proposed here is made up of a series of neural network, and its purpose is to forecast price movements in a high-frequency setting. In particular, we apply our model to Limit Order Book data from the NASDAQ and Bitfinex exchanges. Moreover, we illustrate a series of analysis steps that need to be taken in order to determine relevant parameters for models -not necessarily similar to ours- trying to deal with the same data at our disposal.

Sommario

Nel settore finanziario, come in altri campi, l'usabilità dei modelli dipende in gran parte dalla flessibilità che offrono in un ambiente di produzione e dalla capacità di adattarsi a scenari diversi da quelli pensati nella fase di definizione del modello. In questo lavoro, presentiamo un modello modulare che offre grande flessibilità e consente di essere scalato in base alle esigenze. Il modello qui proposto è costituito da una serie di reti neurali e il suo scopo è prevedere i movimenti dei prezzi in un ambiente ad alta frequenza. In particolare, applichiamo il nostro modello a dati provenienti da Limit Order Book estratti dagli exchange NASDAQ e Bitfinex. Inoltre, illustriamo una serie di passaggi di analisi che devono essere compiuti per determinare parametri rilevanti per modelli -non necessariamente simili al nostro- che cerchino di trattare gli stessi dati a nostra disposizione.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General framework

In the last decades, machine learning has gained a lot of attention worldwide. A machine learning model is essentially an universal approximator of non-linear functions. A deep learning model tends to be incredibly robust with respect to overfitting data, and can be trained on enormous datasets to be able to classify and predict new observations. Development in this field have accelerated rapidly in the last two decades, because of the increasing recording and availabilty of large datasets, as well as advancements in hardware, such as GPU, that can be used to train machine learning models.

With the advent of modern electronic markets, most historical data regarding financial markets are readily available to be used to fit deep learning models. However, it is relatively uncommon to be able to use large datasets in this field. For example, if one is to retrieve daily stock market data of the S&P500, one of the most important indices in the field, it would only be able to obtain less than one hundred thousand datapoints. This problem can be avoided if one is to access high frequency data, which is what we use in this work.

Most applications of machine learning in finance are not related to stock forecasting. Rather, most applications are about tasks such as derivatives calibration and pricing, as well as risk management purposes.

Stock forecasting is a difficult task for many reasons: firstly, and most notably, markets are irrational, and large moves in stock prices are exogenous, coming from events such as press releases, earnings report, geo-political turmoil, influencers' tweets, and much more. Therefore, the information that could be consequential to predict new observations is not generally included in the market data itself. Moreover, any consistently successful forecasting model can be used in principle to make low-to-no-risk profits. Therefore, were there any such models, large market participants would surely make use of them, and the markets would 'price'[1] those in, and they wouldn't be successful anymore.

1.2 Brief description of this work

Since the advent of electronic markets, a large amount of literiture has been produced on the topic. This is also due to the fact that, in a high-frequency setting there's availability of large number of data, contrary to most other financial timeseries' analysis. This is only partly balanced by the difficulty of accessing this type of data, that usually require subscription to specific softwares, such as LOBSTER, since the commonly used data providers such as Bloomberg and Reuters don't offer limit order books snapshots.

Generally, we can distinguish between theoretical models and data-driven models. A large number of both types of studies have been produced.

Some examples of theoretical models include Blanchet and Chen (2013), where the authors derive a continuous time model for the joint evolution of the mid price and the bid-ask spread from a multiscale analysis of the whole limit order book dynamics; Avellaneda and Stoikov (2008) and Cont et al. (2010) where the authors propose a diffusion model for the best bid and ask prices; Cont and De Larrard (2013) in which arrivals of LOB events are modelled in terms of Markovian queueing system; Abergel and Jedidi (2013) describe the order book as a multidimensional continuous-time Markov chain, with a particular focus of LOB events, most particularly cancellations; Donier et al. (2015) where limit order book events are analysed through the help of a reaction-diffusion; Carmona and Webster (2012) model the market maker

[1] 'Priced in' is a terms used in finance to indicate that an information, such as earnings reports, is already 'included' in the market price of a given security.

stochastic optimization problem as a stochastic control problem; Avellaneda et al. (2011) try to forecast new prices using a diffusion model.

Some examples of data-driven models include Qureshi (2018) where the authors compare severals machine learning methods for forecasting future mid-prices; Passalis et al. (2018), where the authors use a Bag-of-Features approach to predict future mid-prices; Nevmyvaka et al. (2006) a reinforcement learning approach is used to optimize order executions; Zhang et al. (2019) where a Convolutional Neural Network is used to predict future prices; Kercheval and Zhang (2015) where the authors make use of a Support Vector Machine approach.

This work is based on the work of Sirignano (2019). The authors take advantage of the discrete nature of prices of securities traded in electronics, order-driven exchanges such as the NASDAQ exchange, to frame stock forecasting as a classification problem. They then develop a model, which they call the *spatial neural network*, to model the tails of the distribution of future price movements. Classification problems usually consists in modelling a discrete probability distribution. In the case of forecasting, it consists of, in particular, modelling a *future* probability distribution. The approach of the authors lies in utilising a neural network for modelling a conditional probability, the form of which is justified by a local behaviour of the data.

We approach the classification problem in the same way, with a slightly different target of forecasting, namely the joint distribution of best bid and best ask price. Moreover, we make use of a modular model in which 3 neural networks per side[2] are used.This architecture was inspired by a analogous architecture proposed in the first version of the work from Sirignano, uploaded to ArXiv and available at `https://arxiv.org/pdf/1601.01987.pdf`.
In their architecture, 2 *spatial neural networks*, are used, along with a third neural network. This third neural network is a classificator, which determines the general direction of prices (either up, down, or constant).

However, we find that for 2 out of those 3 neural networks, the features used as input are somewhat naive. Indeed, in the final published work, the authors give most of the attention to the network that we do not modify, which is

[2] 3 networks for the bid side, and 3 for the ask side, for a total of 6 neural networks.

used to forecast movements of the bid and ask prices moving towards their respective side of the book[3]. We reproduce their study intended to justify its input, with some slight differences dictated by practical limitations. As far as the other two networks are concerned, we propose two different analysis to justify the use of certain features as inputs. In the case of the second *spatial neural network*, we analyse the bid-ask spread behaviour and other local properties, before using the same features used by the authors. In the case of the third neural network, we radically change the input following an extensive work of feature selection and literature review.

There are many examples for the third neural network used in this work, and we took inspiration from the one used in Zhang et al. (2019), where the authors used a Recurrent Neural Network (RNN), paired with a convolutional layer to predict next prices' movements. The convolutional layer is applied to the entirety of the limit order book information, and so it is meant to automatically detect important features of the data.
Dissimilarly to this, we firstly identify relevant features, and then hard code features into our input pipeline.

Other than modifying the inputs of the models proposed by Sirignano after some due diligence, we also perform some empirical analysis to determine other relevants decisions for our models, such as sampling time and the range in which to predict new observations. This analysis is valid for any work trying to approach Limit Order Book forecasting as a classification problem, and it's not restricted to this particular model.

1.3 Chapters summary

This thesis is organised as follows:

In Chapter 2, we will give a brief introduction of modern order-driven markets, where our data is from. We will explore some features of order-driven markets, such as hidden liquidity, and we will delve a little deeper on the particular exchanges that we used.

In Chapter 3, we will set the stage for the rest of this work, formulating

[3]For the best ask, that means increasing in price, and decreasing for the best bid

our problem and exploring which behaviour of our data justify the nature of the proposed architecture.

In Chapter 4, we will delve deeper on the data we have available, and what empirical properties and features they have, and what this means for our model. The last part of the Chapter is dedicated to explain the feature selection process we utilized.

In Chapter 5 we completely specify how the model is composed, and how it can be used to predict new data. We also present the benchmark model that we will compare our model to.

In Chapter 6, results are presented and discussed.

In Chapter 7, we draw some conclusion and present ideas for future work.

In Appendix A, we specify the equity data, i.e. we list the stocks used in this analysis, as well as the time windows from which our data are from.

Chapter 2

Order-driven markets

An increasing proportion of financial transactions -in stocks, futures and other contracts- take place in electronic, order-driven markets where all buyers and sellers display the prices at which they wish to buy or sell a particular security, as well as the amounts of the security desired to be bought or sold. This kind of trading environment is the opposite of a quote-driven market, which only displays bids and asks of designated market makers and specialists for the specific security that is being traded.

In a order-driven market, traders may submit limit orders (for buying or selling), market orders and order cancellations which are then centralized in a limit order book and executed according to precise time and price priority rules. Priority is always based on price, and then, in most markets, on time, according to a FIFO (First In, First Out) rule.
Essentially, three types of orders can be submitted:

- Limit order: An order to specify a price at which one is willing to buy or sell a certain number of shares, with their corresponding price and quantity, at any point in time.
- Market order: An order to immediately buy or sell a certain quantity, at the best available opposite quote.
- Cancellation order: An order to cancel an existing limit order.

2.1 Limit Order Books

The Limit Order Book represents, at each point in time, the outstanding orders which are awaiting execution: it consists in queues at different price levels where these orders are arranged according to time of arrival. A limit new buy (resp. sell) order increases the size of the bid (resp. ask) queue. Market orders are executed against limit orders at the best available price: a market order decreases of size x the corresponding queue size by x. Limit orders placed at the best available price are executed against market orders. In contrast to markets where a market maker or specialist centralizes buy and sell orders and provides liquidity by setting bid and ask quotes, these electronic platforms aggregate all outstanding limit orders in the limit order book that is available to market participants and market orders are executed against the best available prices, in a mechanical manner.

Established exchanges such as the NYSE, Nasdaq, the Tokyo Stock Exchange, Toronto Stock Exchange, Vancouver Stock Exchange, Euronext (Paris, Amsterdam, Brussels), and the London Stock Exchange have fully or partially adopted electronic order-driven platforms. At the same time, the frequency of submission of orders has increased and the time to execution of market orders on these electronic markets has dropped from more than 25 milliseconds in 2000 to less than a millisecond in 2010. As a result, the evolution of supply, demand and price behaviour in equity markets is being increasingly recorded: this data is available to market participants in real time and for researchers in the form of high frequency databases.

2.2 Hidden Liquidity in Limit Order Books

Along with the previously discussed types of orders, there's another, special type of order, called 'hidden' or 'iceberg', that allows traders, through a higher commission cost, to limit their exposure by hiding a portion of the quantity they are willing to trade. In some opaque limit order books, traders may even either fully hide the quantity of their limit order and disclose the price only (as in the Australian Stock Exchange), or hide the price and the quantity of their order (as in the dark pool[1] Turquoise).

[1]Dark pools are private exchanges for trading securities that are not in general accessible by the investing public.

Hidden orders amount to a striking proportion of trading volume: for example, they correspond to more than 44% of Euronext volume , about 28% of the Australian Stock Exchange volume. In the case of the exchange from which our data are from, Tuttle (2003) finds that the hidden liquidity represents 20% of the inside depth in Nasdaq 100 stocks.

Of course, this affects the quality of the data. Since, and as we will see in Section 4, our LOB data is reconstructed by submitted orders, hidden orders are not included. Therefore, the sizes -but not the prices- contained in our data do not necessarily reflect the real sizes that drives prices dynamics. This is an unsurpassable obstacle, and we can't to better than just use the data at our disposal .

One important concern, however, is the informative value of the non-observable orders. Since in effect market participants hide their quantity in order to conceal information, one may argue that informed traders will, on average, resort to hidden orders more frequently than uninformed traders, and that price movements will reflect that information, making hidden sizes' knowledge more consequential to forecast future movements.

However, Aitken et al. (2001) show that in the Australian stock market there is no difference in the stock price reaction between disclosed and undisclosed limit orders, and conclude that there is no evidence that undisclosed limit orders are more frequently used by informed traders. Moreover, Bessembinder et al. (2009) show that in the Euronext Exchange hidden orders are actually mainly used by uninformed traders.

2.3 Importance of modelling Limit Order Books

The analysis of such high frequency data constitutes a challenge, not the least because of their sheer volume and complexity. These data provide us with a detailed view of the complex dynamic process through which the market 'digests' the inflow of supply and demand to generate the price. The large volume of data available, the presence of statistical regularities in the data and the mechanical nature of execution of orders makes order-driven markets interesting candidates for statistical analysis and stochastic modelling.

At a fundamental level, statistical analysis and modeling of high frequency data can provide insight into the interplay between order flow, liquidity and price dynamics and might help bridge the gap between market microstructure theory –which has provided useful insights by focusing on models of price formation mechanism stylized equilibrium settings– and 'black box' stochastic models used in financial risk management, which represent the price as an exogenous random process.

At the level of applications, models of high frequency data provide a quantitative framework for market making and optimal execution of trades. Another obvious application is the development of statistical models in view of predicting short term behavior of market variables such as price, trading volume and order flow. The study of high frequency market dynamics is also important for risk management and regulation. Even though the horizons traditionally considered by risk managers and regulators have been longer ones (typically, daily or longer), trading strategies, at different frequencies may interact in a complex manner, leading to ripples across time scales which propagate from high frequency to low frequency and even leading to possible market disruptions, as shown by the Flash Crash of May 2010.

2.4 NASDAQ exchange

NASDAQ is the acronym for National Association of Securities Dealers Automated Quotation. It is an American stock exchange based in New York City. It is ranked second on the list of stock exchanges by market capitalization of shares traded, behind the New York Stock Exchange. The exchange platform is owned by Nasdaq, Inc., which also owns the Nasdaq Nordic stock market network and several U.S. stock and options exchanges.

This market was established on Wall Street on February 8, 1971 and was the world's first purely electronic stock exchange. Originally, computers were used only to disseminate price information continuously and not to connect operators: the passage of orders, up to 1987, took place via telephone. However, the dissemination of online quotes ensured a significant increase in transparency and market efficiency: for a long time, the Nasdaq was the stock market where spreads were lower.

The entirely electronic transmission of orders was also established at the end of 1987. During the collapse of the market that took place in October of that year, the unsustainability of order collection by telephone was evident: given the large number of incoming orders, traders and dealers could not physically answer phones and the lines often dropped.

Most of the data available to us comes from the NASDAQ Stock Exchange. In Section 4, we will more thoroughly explain exactly what kind of data is publicly available and the steps to take in order to get to actual Limit Order Book data.

2.5 Bitfinex exchange

Bitfinex is a Hong Kong-based digital asset trading platform, owned and operated by iFinex Inc., which is headquartered in Hong Kong and registered in the British Virgin Islands. It allows its users to participate in several markets. The Exchange Trading entails trading in a regular cryptocurrency exchange, by making use of limit order books for the spot trade of digital tokens.

Moreover, it also offers the possibility to take part in an Over The Counter trading system, where market participants can conduct large deals directly with a counterparty without using public order books, which allows them to access significant amounts of liquidity without affecting the exchange market price.

2.5.1 Tether

By far the most traded asset on Bitfinex is a cryptocoin known as Tether (often called by its symbol USDT) which is a cryptocurrency with tokens issued by Tether Limited, which in turn is controlled by the owners of Bitfinex. There are about 63.2 billion USDT tokens in existence. This number of USDT amounts to roughly the same value in USD.

This is because the Tether coin is a stablecoin, i.e. it's meant to closely

follow the value of the US Dollar. It was originally designed to always be worth $1.00, maintaining $1.00 in reserves for each tether issued. While, according to its 2021 settlement with the New York Attorney General Letitia James, "Tether represents to users that any holder of tethers can redeem them from Tether the company at the rate of one tether for one U.S. dollar." Tether Limited as of 2017 stated that owners of tethers have no contractual right, other legal claims, or guarantee that tethers will or can be redeemed or exchanged for dollars. On 30 April 2019 Tether Limited's lawyer claimed that each tether was backed by $0.74 in cash and cash equivalents. In May 2021, Tether published a report showing that only 2.9% of Tether was backed by cash, with over 65% backed by commercial paper. As we will see later this has profound impacts for our purposes.

We have at our disposal some Tether data, along with the data from the NASDAQ exchange. Over this work, we will analyze what differences arise between the two for the scope of our model. We will also see what are the implications, for our work, of the stablecoin nature of Tether.

Chapter 3

Problem Formulation

This chapter is the central chapter of this work and, after some basic definitions, we will formulate the problem we are trying to solve, as well as which features of the data justify the particular formulation used in this work.

3.1 Limit order books

3.1.1 Basic Definitions

In general, the best bid price of a security is the highest amount that market participants are willing spend in order to buy said security. Conversely, the best ask price is the lowest amount they are willing to sell it for. The mid price is defined as the mean of these two prices. In an order-driven market, a trader can place an order below the current best bid price, and above the best ask price. Therefore, a Limit Order Book is characterized by more than 2 prices -and relative order sizes-, but by two sets of quantities. The first set refers to all the prices at which an order is placed, and the second set refers to the sizes of such limit orders.

Definition 3.1.1 (**Limit Prices**)**.** Limit prices are defined as all prices at which a limit order is currently placed.

$$\textbf{Limit prices} = [..., P_k^B, ..., P_1^B, P_0^B, P_0^A, P_1^A, ..., P_k^A, ...].$$

They are divided into bid prices, i.e. prices at which market participants are willing to buy the security in question, and ask prices, i.e. prices at which

market participants are willing to sell it.
In this notation P_0^B and P_0^A refers to the best bid and best ask prices, respectively. $P_k^{A/B}$ refers to the k-th limit order, respectively below and above the best bid and ask prices.

Along with the limit order prices, the order book is characterized by their respective order sizes, i.e. the overall number of securities that can be either bought or sold at a given limit price.

Definition 3.1.2 (**Order sizes**)**.** Order sizes are defined as all the amounts of securities available to be traded at the various limit prices

$$\textbf{Order sizes} = (.., V_k^B, .., V_1^B, V_0^B, V_0^A, V_1^A, .., V_k^A, ..),$$

where $V_k^{A/B}$ is the order size correspondent to the price $P_k^{A/B}$.

A crucial notion in orden-driven markets is that of tick size, or resolution limit: a limit order cannot be placed at any conceivable price, i.e. any positive real number. Possible limit orders can only be placed at multiples of the tick size. There is in effect a resolution limit for LOB prices:

Definition 3.1.3 (**Tick size**)**.** Tick size is defined as the smallest possible difference between two prices:

$$tick \in R^+ : \exists h \in N : |P_k^{B/A} - P_j^{B/A}| \; = \; h \cdot tick \quad \forall k, j.$$

From now on, we will be using the words "ticks" and "levels" interchangeably, and we will be referring to possible limit prices -multiples of the tick size- as price levels.

As we will see later, ticks, rather than amounts of currency[1] will be the main 'unit of measure' of securities' movements.

[1] In the case of the NASDAQ exchange, one tick corresponds to 1 cent of the currency, i.e. US Dollars.

3.1.2 Limit Orders Book representation

The discrete nature of a limit order book makes possible to allow a simpler representation of its state, at any given time. This can be done by assuming that every possible price level below the best bid and and above the best price are occupied up to a certain depth N, i.e assuming that the following equation holds:

$$\textbf{Limit Prices} = (P_0^B - N \cdot tick, .., P_0^B - tick, P_0^B,\ P_0^A, P_0^A + tick, .., P_0^A + N \cdot tick),$$

where **Limit Prices** is defined according to 3.1.1.

Therefore, the state of the order book can be uniquely characterized by $P_0^B(t)$ and $P_0^A(t)$, and the order sizes correspondent to all included price levels, so that:

Definition 3.1.4 (**LOB**)**.** A simplified representation of the state of the limit order book:

$$\textbf{LOB} = \{(P_0^B, P_0^A)\ ,\ (V_N^B(t), ..., V_1^B(t), V_0^B(t), V_0^A(t), V_1^A(t), ..., V_N^A(t))\}\ ,$$

where $V_k^B(t)$ is the total size associated to the price $P_0^A(t) - k \cdot tick$ and $V_k^A(t)$ is the total size associated to the price $P_0^A(t) + k \cdot tick$.

Note that this representation lacks no generality, in the sense that if an order is not present at a given price level, say the $\bar{k}$-th, we allow $V_{\bar{k}} = 0$.
For liquid stocks, this is relatively uncommon, i.e. there tend to be outstanding limit orders on all possible price levels. However, this allows our representation to be spatial in nature, where limit orders can occupy predetermined states -i.e. price levels-. As we will see later in this section, this is highly connected with the model architecture.

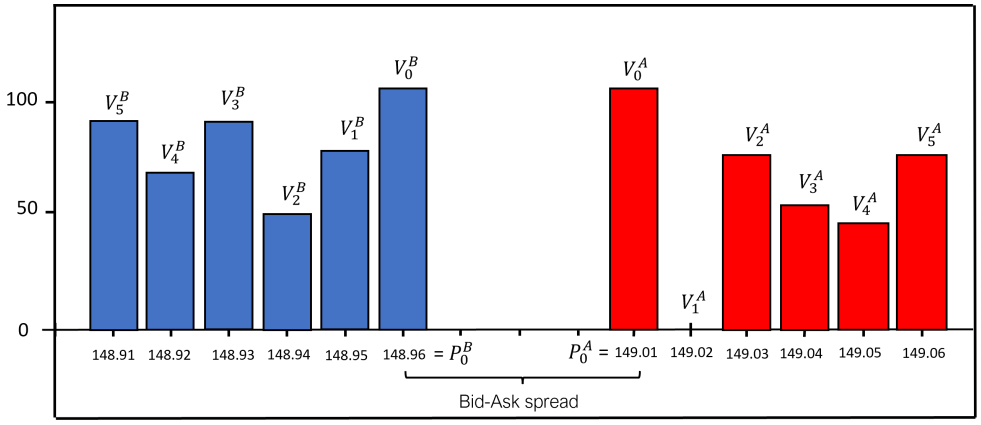


Figure 3.1: Limit Order Book spatial representation with a 1 cent tick size.

3.1.3 Orders impact on the Limit order book

Order books data vary between exchanges and Book Construction methods, as we will see. But on a general level, the book is updated whenever a new order, out of the different types of orders, is submitted. Therefore, any subsequent change in the order book doesn't necessarily reflect a change in either best bid or best ask price.

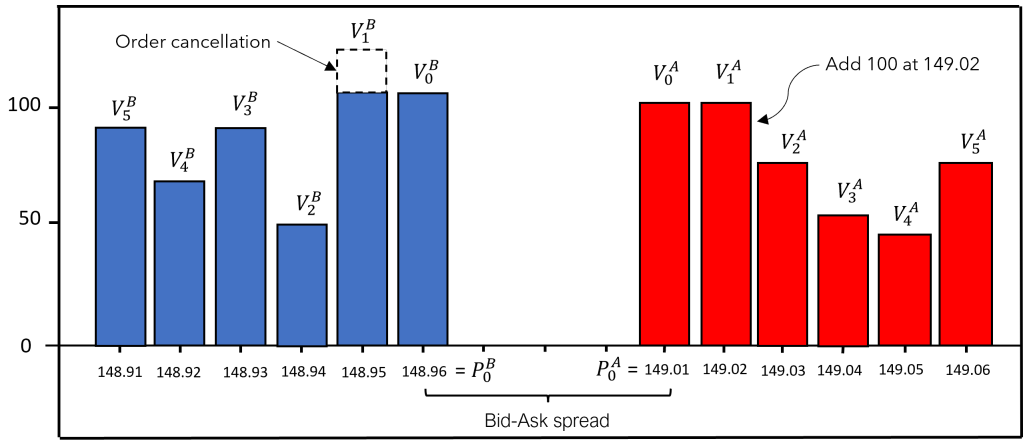


Figure 3.2: Example of two limit orders on the order book.

In Figure 3.2 we can see the impact of two different types of orders execution on the initial state in Figure 3.1. A sell limit order for 100 shares, at price

149.02 -1 price level away from the best ask price-, is submitted. Moreover, a previously submitted buy limit order of 20 shares at price 148.95 -1 price level away from the best bid price- is cancelled, reducing the overall bid volume at level 2, V_1^B. Note that these were two events, corresponding to two updates of the book.

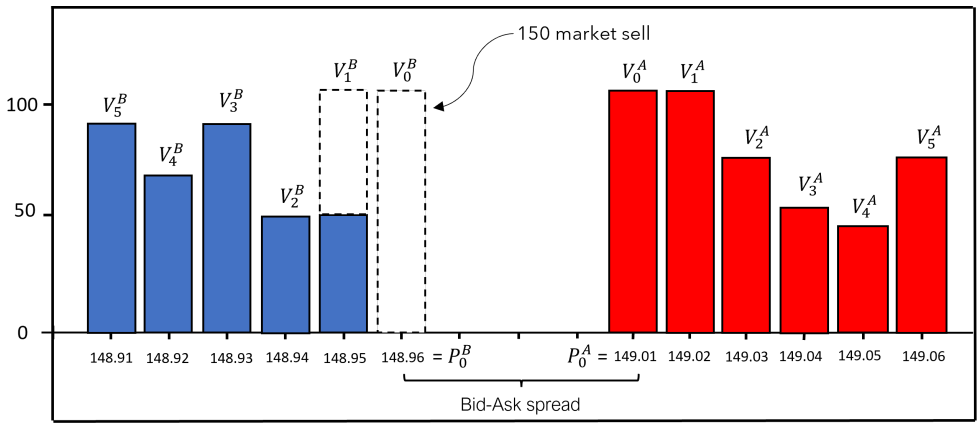


Figure 3.3: Example of a market sell order on the order book.

Subsequently to the situation in Figure 3.2, in Figure 3.3 it is shown the impact of a market sell order of 150 shares. A market sell is executed at the best available bid price. In this case, since the best bid price corresponds to 100 shares, the sell order consumes all the available size, as well as part of the second best bid price. Note that this changes the Bid-Ask spread, differently from the two previous limit orders. Naturally, limit orders can impact best ask and best bid as well, by simply posting either a buy limit order on a price greater than the best bid, or by posting a sell limit order on a price smaller than the best ask. On the other side, market buy/sell orders may not change best bid or best ask prices, if the size of the order is not enough to consume the total size posted at the best bid/ask price. In Figure 3.3 this would happen if the size of the market sell order would have been lower than 100.

3.2 Stock forecasting as a classification problem

The discrete nature of the limit order book also implies that time variations of a given price level, say the best bid, also can only amount to multiples of the tick size, i.e.:

$$\exists h \in (.., -2, -1, 0, 1, 2, ..) : |P_0^B(t) - P_0^B(s)| = h \cdot tick \quad \forall t, s \in R^+ \ .$$

This means that price forecasting is in effect not a regression problem, where $h \in R$, but rather a classification problem , aimed at determining which h characterises the price movement. Indeed, the output of our model will be a probability distribution on the h-space, i.e. the set of integers $Z = (.., -2, -1, 0, 1, 2..)$.

A couple of natural questions arise, however. The first question is that any price forecasting application on an exchange with a resolution limit, and not only a high-frequency application, should be a classification problem. Secondly, in a classification framework classes are uncorrelated, whereas in our problem this is clearly not the case. This is because our application is spatial in nature, since h represents the magnitude of a price movement, and a notion of distance between classes is trivially defined. For example, the events $h = 4$ and $h = 5$ are much more correlated than the events $h = -1$ and $h = 17$.

Having access to limit order book data can be a possible solution for both of this problems.

As for the first, it is true that any forecasting application in presence of a tick size should be a classification problem, and a regression is only an approximation. However, if the time horizon is anything higher than what is generally considered high frequency, e.g. daily, the natural range of possible tick sizes of variation becomes unfeasibly large, as a daily variation ranges orders of magnitude more than a 1s variation. This causes the number of classes to be considered to increase dramatically, causing stability problems. Therefore, the use of LOB data allows us to select an appropriate sampling time to somewhat control classes' balancing. We will delve on this more deeply in the next chapter.

As for the second problem, the use of limit quotes beyond best bid and best ask allow our architecture to not only take this spatial nature of our classification problem into consideration, but also to make use of any spatial relationship in our data. We explain how this can be done in practice in the next section.

3.3 Local behaviour of the Limit Order Book

In order to make use of spatial relationships deep in the order book, Sirignano (2019), builds an appropriate architecture to do so. The first building block of this architecture is based on the idea that there's a local relationship in the way the best bid and best ask price move.
Let us define the prices variations, that will be the target of our model:

Definition 3.3.1 (**Prices Variation**).

$$Y^{A/B} = P_0^{A/B}(t+\tau) - P_0^{A/B}(t),$$

where τ can either be a sampling time or a random variable indicating the next price move.

Y^A refers to the best ask price variation, whereas Y^B to the best bid variation. When the difference between them is not relevant, we will stick to the $Y^{A/B}$ notation, as done in the definition. Throughout this work, we will measure $Y^{A/B}$ in ticks, i.e. if $P_0^{A/B}(t+\tau) = P_0^{A/B}(t) + 2 \cdot tick$, then $Y^{A/B} = 2$.

Let us consider best ask variation, without lack of generality. In principle, it could depend on the whole state of the order book. However, Sirignano (2019) finds that the probability that Y^A moves more than a certain amount of levels y, conditional on moving by at least y levels, mostly depends on the size at level y, V_y^A:

$$P(Y^A > y | Y^A \geq y) = f(V_y^A)\,. \tag{3.1}$$

Looking at Figure 3.4, it is quite clear why there should be this local relationship: conditioned on the fact that the next best ask price $P_0^A(t+\tau)$ will assume any price level greater or equal to $P_0^A(t+\tau) + 1 \cdot tick$, the event that it will actually go over that level can only occur if the entire liquidity

present at V_1^A is consumed by subsequent orders. In other terms, V_1^A acts as a barrier for Y^A to move higher than $y = 1$ levels.

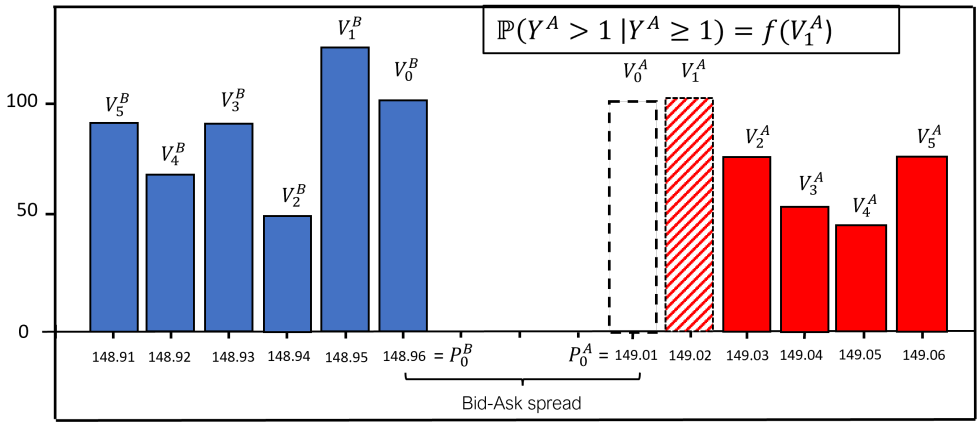


Figure 3.4: Visualisation of the event $Y^A > 1|Y^A \geq 1$. The first ask size is left blank as we are conditioning on $Y^A \geq 1$, so taking for granted that that event occurred. The event in which we are interested refers to whether or not the best ask price will move up by 2 levels, by consuming the size highlighted by the diagonal lines.

Again looking at Figure 3.4, it should be clear why we use the LOB representation introduced in Definition 3.1.4, where we allow V_k^A to be zero if there's no outstanding limit order at price level $P_0^A + k \cdot tick$, instead of making V_k^A refer to the k-th non-zero limit order. We need V_k^A be the liquidity to be consumed at exactly $P_0^A + k \cdot tick$, in order not to have a spatial mismatch between order sizes and price levels. If V_k^A happens to be zero, then $P(Y^A > k|Y^A \geq k)$ will simply increase. Note that $V_k^A = 0$ doesn't imply $P(Y^A > k|Y^A \geq k) = 1$, since between t and $t+\tau$ a limit order could fill the k-th price level.

A good measure for capturing this local behaviour comes from fitting a logistic regression to estimate the conditional probability in (3.1), using the ask sizes in a neighbourhood of y:

$$P(Y^A > y|Y^A \geq y) = f(V_{y-k}^A, .., V_{y-1}^A, V_y^A, V_{y+1}^A, .., V_{y+k}^A) \,,$$

where f is the logit function, i.e.

$$f(\vec{x}) = (1 + \exp\{\beta + \sigma \cdot \vec{x}\})^{-1} .$$

In Figure 3.5, coefficients from the logistic regression using $y = 12$ and $k = 5$ are shown. The central coefficient, correspondent to V_y^A is greater than all others, having more influence on the event $Y^A > y | Y^A \geq y$. Moreover, the p-value associated is the sole p-value below the traditional significance threshold, confirming that V_y^A is the only significant regressor.

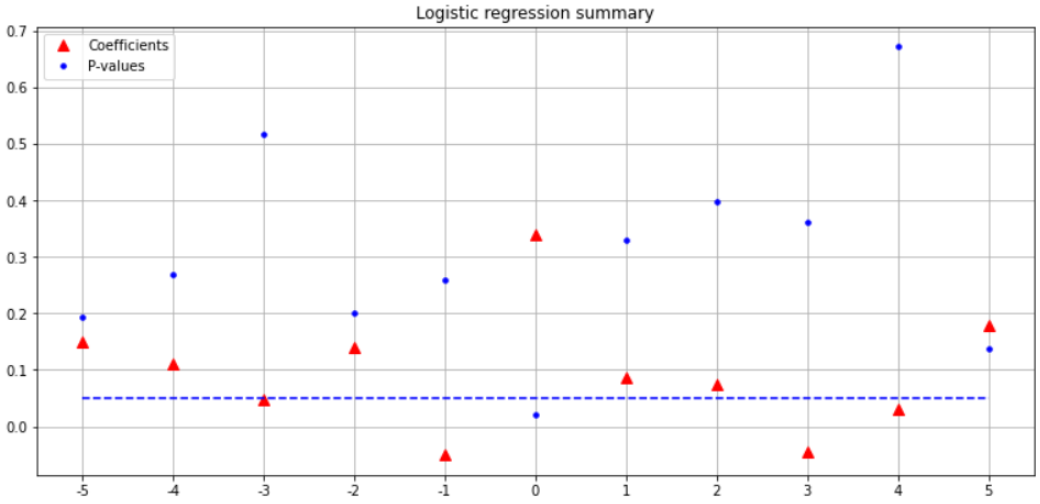


Figure 3.5: Coefficients from the logistic regression, as well as correspondent p-values and the threshold $\alpha = 0.05$. On the x-axis, the distance from $y = 12$, with V_y^A corresponding to $x = 0$. The logistic regression was fitted on the stock Amazon.

As we will explain better in the next chapter, statistical analysis **across** stocks is restricted whenever the use of a full-depth of a limit order book is required. This is because of the large memory required to store full-depth limit order book is much greater than the one required to store first-level limit order book data. Therefore it is difficult confirm this behaviour across stocks, by keeping y and k fixed. To avoid this, we perform the logistic regression of above varying y and k, as well as iterating across the stocks in Set B[2]. We then compute two sets of quantities:

[2]As explained in the next chapter, Set B refers to the list of stocks used for analyses

- $CoefRatio_k$: The frequency with which the coefficient correspondent to the y-th level is within the top k greatest coefficients.
- $PvalueRatio_k$: The frequency with which the p-value correspondent to the y-th level is within the top k smallest p-values.

Results are summarized in Table 3.1 below:

	$CoefRatio_k$	$PvalueRatio_k$
$k = 1$	47.1%	52.3%
$k = 2$	72.5%	78.1%
$k = 3$	95%	98.9%

Table 3.1: Summary of the local behaviour of the limit order book.

After estimating $f(\cdot)$, it is possible to reconstruct the entire distribution of Y^A via a simple application of conditional probability properties:

Lemma 3.3.1 (**Spatial representation**).

$$P(Y^A = y|Y^A > 0) = (1 - f(V_y^A)) \prod_{k=1}^{y-1} f(V_k^A) \, .$$

Proof. Let us prove instead, but equivalently, that:

$$P(Y^A > y|Y^A > 0) = \prod_{k=1}^{y} f(V_k^A),$$

by induction:

- $y = 1$: by definition of $f(\cdot)$ if follows that,

  $$P(Y^A > 1|Y^A > 0) = P(Y^A > 1|Y^A \geq 1) = f(V_1^A).$$

- $y \implies y + 1$: by definition of conditional probabililty, and by the fact that $P(\cdot|Y^A > 0)$ is a probability, it follows that:

  $$P(Y^A > y+1|Y^A > 0) = P(Y^A > y+1|Y^A \geq y+1)P(Y^A \geq y+1|Y^A > 0),$$

where a full-depth order book is required.

so that, given that $P(Y^A \geq y+1|Y^A > 0) = P(Y^A > y|Y^A > 0) = \prod_{k=1}^{y} f(V_k^A)$ because of the induction hypothesis,

$$P(Y^A > y+1|Y^A > 0) = P(Y^A > y+1|Y^A \geq y+1) \prod_{k=1}^{y} f(V_k^A) = \prod_{k=1}^{y+1} f(V_k^A)$$

□

Lemma 3.3.1 will be the only theoretical tool used in this work. It's worth noticing how $P(Y^A = y|Y^A > 0)$ is reconstructed, and not the full probability $P(Y^A = y)$.

This is because the reasoning for which the order sizes below the best bid/ask should act as a kind of 'barrier' against its movement it's valid whenever the best ask and bid move into their respective side of the limit order book, i.e. the best ask increasing in price, and the best bid decreasing.
So whenever trying to model $P(Y^A = y|Y^A < 0)$, one cannot make use of the this local behaviour.

However, the bid-ask spread is often modelled as a constant, meaning that the best ask and best bid tend to move in lockstep, as represented in Figure 3.6. This would allow us to keep leveraging the local behaviour of the order book, with a simple modification: the probability of the best ask decreasing should depend on the bid size of the book:

$$P(Y^A = y|Y^A \leq y) = f(V_y^B). \tag{3.2}$$

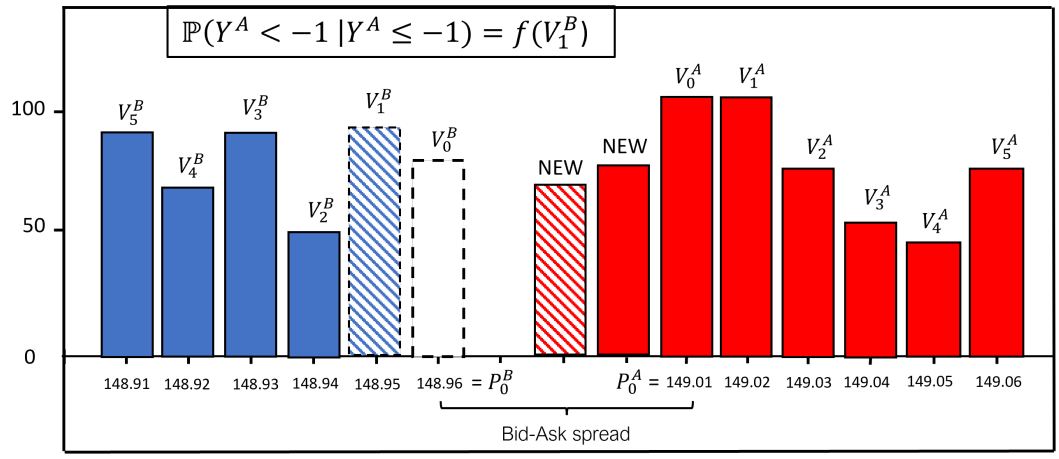


Figure 3.6: Best bid and best ask moving in lockstep 2 levels down. In this case, it is sensible to allow $P(Y^A < -1|Y^A \leq -1)$ to depend on the size at level 1 in the bid size, i.e. V_1^B.

To show this, we fit a similar logistic regression for the ask movement as above, but this time using the bid order sizes i.e.:

$$P(Y^A < y|Y^A \leq y) = f(V_{y-k}^B, .., V_{y-1}^B, V_y^B, V_{y+1}^B, .., V_{y+k}^B).$$

In Figure 3.7, coefficients from the logistic regression using $y = -12$ and $k = 5$ are shown. The central coefficient, correspondent to V_y^B is greater than all others, having more influence on the event $Y^A < y|Y^A \leq y$. Moreover, the p-value associated is the sole p-value below the traditional significance threshold, confirming that V_y^B is the only significant regressor also in this case. Results similar to those of Table 3.1 can also be achieved.

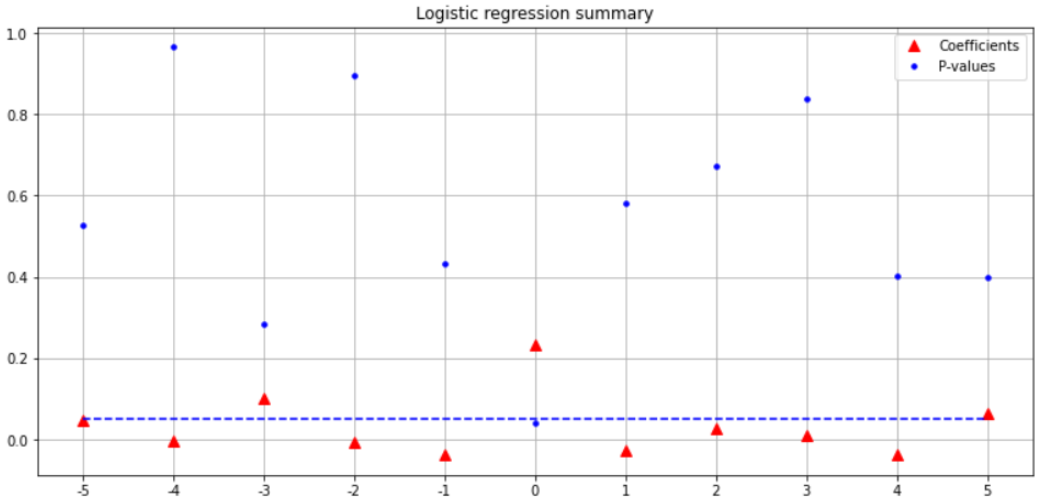


Figure 3.7: Coefficients from the logistic regression, as well as correspondent p-values and the threshold $\alpha = 0.05$. On the x-axis, the distance from $y = 12$, with V_y^B corresponding to $x = 0$. The logistic regression was fitted on the stock Amazon.

Everything mentioned in this section about behaviour of best ask applies symmetrically to the best bid price, i.e. it depending on its side of the order book when decreasing and on the ask side when increasing, so that Equations 3.1 and 3.2 become:

$$P(Y^B = y | Y^B \leq y) = f(V_y^B) \quad \textit{and} \quad P(Y^B = y | Y^B \geq y) = f(V_y^A).$$

However, as it's shown in the next section, there's evidence to conclude that the bid ask spread shouldn't be modelled as a constant too confidently.

3.4 Joint bid ask distributions

In this section we will explore how best bid and best ask tend to move, whether jointly or not. To do this, for every stock j in the dataset we compute a set of quantities:

- Z_j: the frequency of best bid and ask moving in lockstep.

- M_j: the frequency of one of the two best prices moving in the same general direction, i.e. both moving up, both moving down, or both remaining constant.
- Q_α: the empirical quantiles of the bid ask spread, measured in number of levels.

Once these quantities are computed for every stock, we compute their quantiles **across** the stocks in Set A. These quantiles are indicated by a non capital letter q_α, in order not to confused them with the quantiles referring to the different stocks' distributions.

Results are summarized in Table 3.2 below:

	q_{05}	q_{10}	q_{20}	q_{50}	q_{80}	q_{90}	q_{95}
Z	0.075	0.198	0.382	0.785	0.945	0.971	0.983
M	0.61	0.638	0.662	0.717	0.946	0.977	0.991
Q_{10}	1	1	1	1	3.8	7.9	9
Q_{50}	1	1	1	2	8.4	145	14.95
Q_{90}	1	1	1	3	14.4	23.9	50.75

Table 3.2: Spread Summary

Looking at the quantiles of Z, for about 50% of the stocks, best bid and best ask move in lockstep about 80% of the time, whereas for 20% of the stocks, only about 40% of the time. Still, the co-movement remains a powerful feature that we think justifies the use of Equation 3.2.

Either way, it is clear that best bid and best ask share a powerful correlation. This means that it is important to model them jointly, rather then predicting them separately.

One way to do this is to choose a side to predict first, say the bid ask, and then use the best bid movement as a feature to predict the best ask movement, i.e.:

$$P(Y^A > y | Y^A \geq y) = f(V_y^A, Y^B).$$

Note that this doesn't impede predicting in the test set. It is possible to just substitute the best bid movement with the predicted best bid movement, i.e.

$$P(Y^A > y | Y^A \geq y) = f(V_y^A, Y_{pred}^B).$$

Looking at the quantiles of M however, is quite clear that the general direction of the best prices tends to be the same the majority of times for all stocks. As mentioned before, we will make use of a neural network to predict the general direction of prices. Therefore, it seems reasonable that for this particular model is even more important to use the best bid movement as a feature for predicting the best ask movement, i.e.:

$$P(Y^A \in A | X = x) = f(x, Y^B) \quad A \in (\{Y = 0\}, \{Y > 0\}, \{Y < 0\}),$$

where X are other features, that will be explored further in the next chapter.

3.5 Modelling the future probability of stock movement

The goal of our model is to forecast the probability distribution of the best bid price and best ask price at a future time $t + \tau$, i.e. we want to model:

$$P(Y^{A/B} = y).$$

In the next section we will be, among other things, discussing what is the most appropriate value for τ.

The local behaviour of the order book allows us to model $P(Y^{A/B} > y | Y^{A/B} \geq y)$ and $P(Y^{A/B} < y | Y^{A/B} \leq y)$. Starting from this, it is possible to reconstruct, respectively, $P(Y^{A/B} = y | Y^{A/B} > 0)$ and $P(Y^{A/B} = y | Y^{A/B} < 0)$ with two different models[3].
A third model is also used, in order to determine whether $Y^{A/B}$ is greater than zero, lower than zero, or constant, thereby indicating that $P_0^A(t+\tau) = P_0^A(t)$. This last model cannot make use of the local behaviour feature of the order book, and therefore must rely, as we will see in the next section, on a carefully chosen set of features.

There are several advantages in modelling the future distribution rather than making a punctual prediction: first off, modelling the distribution still allows to compute to make punctual prediction, simply by taking $\overline{y} = argmax_y P(Y = y)$ as the most likely movement.

Therefore, modelling the future distribution is more general and allows to handle a wide variety of scenarios. For instance, a market partecipant such as a market maker would be more interested in estimating the probability that the best prices will move within a predefined range, with a punctual prediction having little use when directionality is not a particular concern.
Lastly, modelling the whole distribution offers many solutions as far as risk management is concerned, being able to properly quantify all possible moves' scenarios.

[3] Two for the bid side, and two for the ask size.

Chapter 4

Data and Features

4.1 Dataset

The dataset at our disposal consists of two very different sets of data, one referring to stocks, more precisely stocks traded in the NASDAQ exchange, and one referring to cryptocurrencies, in particular data from the cryptocurrency known as Tether, traded on the Bitfinex exhcange. In this section we will explain the precise format of our data, along with some specifications of our dataset.

4.1.1 NASDAQ dataset

The stock data of our disposal consist of NASDAQ's Historical TotalView-ITCH data. Until 2019, some days worth of data, usually the last day of the month, were published online for free. It's still possible to access some of these data via FTP[1] clients, suchs as Cyberduck, accessing the server ftp://@emi.nasdaq.com/ITCH/.

A .ITCH50 file covers the entirety of negotiations for all stocks traded in the exchange, and it covers the length of the trading day.
Figure 4.1 illustrates the structure of the TotalView-ITCH messages from which the limit order book can be reconstructed. The original data is stored

[1]The File Transfer Protocol (FTP) is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network. FTP is built on a client–server model architecture using separate control and data connections between the client and the server.

in binary files, depicted here is a 'readable' version.

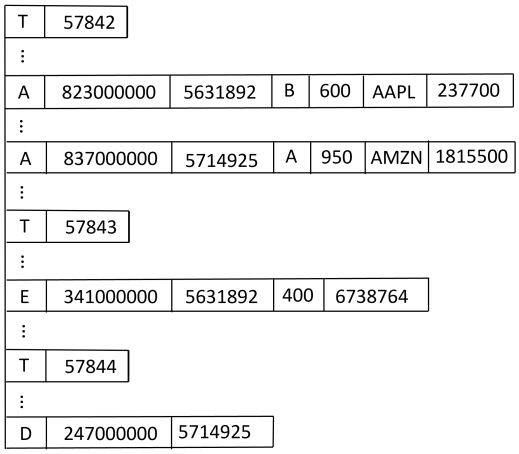


Figure 4.1: Fictitious of a series of TotalView-ITCH messages.

There are two types of messages. The messages indicated by 'T' are time messages giving the number of seconds after midnight. The first entry in the figure below indicates that the subsequent events occur 57842 seconds after the last midnight, i.e. at 16:04:02.
The second types of messages concerns received orders. The second message is a limit order indicated by the type 'A' (for 'add'). The second element of this message gives the number of nanoseconds since the last message of type 'T', i.e. since the last full second after the previous midnight. The exact time in which this order was submitted is therefore 16:04:02.823000000. The nanosecond time stamp is followed by a unique order ID, 5631892, which is used to reference the order in case of an execution or cancellation. The rest of the message gives further details about the order: the market side indicator, 'B', indicating a buy order, the order volume, '600', the stock ticker, 'AAPL' for Apple, and the limit price of '1482300', i.e. $148.23.

The third message is analogous to the previous one, but is an ask order for the stock Amazon. The message of type 'E' together with the preceding message of type 'T' indicate a partial execution of the order with ID '5631892' at 16:04:03.341000000. The transaction of 400 shares is identifiable by the transaction ID, 6738764. Note that in the execution message,

only the unique order ID, '4172917', is referenced. Further detail about this order, such as the price, the ticker and market side have to be inferred from the previous submission message.
The last message is the cancellation of the previous Amazon ask order of ID 5714925.

Starting from .ITCH50 data, it is possible to reconstruct the history book of a given stock of in an iterative manner, by updating its state after every message regarding the stock in question. Note that this is computationally intensive as previous orders' IDs have to be kept in memory for executions and cancellations referring that particular order. In order to do this, we use the BookConstructor.sh function as available at Bernasconi-De-Luca et al. (2021). It takes as input a .ITCH50 file, the ticker of the stock to reconstruct, and the maximum depth at which the Book Constructor should update limit orders.

Below it is shown the output for the Google stock, with 50 levels depth. The first column is the nanoseconds from the previous midnight. The 'k_bid_price' column refers to the k-th best bid price, whereas 'k_bid_vol' refers to its total size. Analogously for the ask side.

	time	1_bid_price	1_bid_vol	1_ask_price	1_ask_vol	2_bid_price	...	49_ask_price	49_ask_vol	50_bid_price	50_bid_vol	50_ask_price	50_ask_vol
0	3.521831e+13	1084.54	105.0	1085.23	30.0	1084.52	...	1091.4	1.0	1080.3	5.0	1091.5	100.0
1	3.521831e+13	1084.54	105.0	1085.23	30.0	1084.52	...	1091.5	100.0	1080.3	5.0	1092.0	1037.0
2	3.521831e+13	1084.54	105.0	1085.23	30.0	1084.52	...	1091.5	100.0	1080.3	5.0	1092.0	1037.0
3	3.521831e+13	1084.54	105.0	1085.23	30.0	1084.52	...	1091.5	100.0	1080.3	5.0	1092.0	1037.0
4	3.521835e+13	1084.54	105.0	1085.23	30.0	1084.52	...	1091.5	100.0	1080.3	5.0	1092.0	1037.0

Figure 4.2: Output of the BookConstructor for the Google stock. Data from 30/01/2019.

The days at our disposal are listed in Appendix A. We use a total of 8 days, amounting to a total size of about 15 GB.

This is not a large amount of days, however it is not uncommon for the literiture to work with similar amounts of data, as in Qureshi (2018) and Zhang et al. (2019). Indeed it is important to keep in mind that whereas the ITCH files are relatively small, from each ITCH file it's possible to reconstruct the daily book of every single stock traded on the NASDAQ exchange.

An average 50-level book for a mid-to-large cap stock occupies around 0.5 GB. So unless only a single stock is of interest, the memory usage to store reconstructed limit order book data increases dramatically from the raw ITCH data.

The fact that a 50-level book, which is the depth we usually use for our models, takes up much more memory than a 1-level book[2], has deep implication for our results. Statistical analysis **across** stocks are only possible for us to do whenever only the first level is required, e.g. for an analysis of the bid-ask spread. This is because in order to perform such analysis a relatively large amount of stocks are required, and it would be impractical for us to keep a large amount of 50-level books in memory.

In Appendix A, two sets of stocks are listed: Set A, which contains the stocks we used for a 1-level analysis, and Set B, which contains the list of stocks we used for all other analyses that required more than 1-level, e.g. fitting our models.

4.1.2 Bitfinex data

Dissimilarly from the stock data, our crypto data consists in already reconstructed and sampled data. In particular, we have at our disposal the limit order book snapshots in CSV format. These snapshots are already sampled with a frequency of 1 second,spanning the entire month of September 2020. Below it is shown how the Tether limit order book file looks, with 50 levels depth. The first two columns represents the time in *h:m:s:ns* format. The first refers to the time of the exchange, and the second the time of the api snapshotting data from the exchange. The 'ask[k].price' column refers to the k-th best bid price, whereas 'ask[k].price' refers to its total size. Analogously for the bid side.

[2] i.e. a book that contains only the best bid and the best ask prices

	time_exchange	time_coinapi	asks[0].price	asks[0].size	bids[0].price	bids[0].size	...	bids[48].price	bids[48].size	asks[49].price	asks[49].size	bids[49].price	bids[49].size
0	00:00:00.7416557	00:00:00.7416557	1.0013	71571.693148	1.0011	84018.570076	...	0.99946	3000.0	1.0070	15.000000	0.99945	2408.372411
1	00:00:03.8882137	00:00:03.8882137	1.0013	71571.693148	1.0011	84018.570076	...	0.99946	3000.0	1.0070	15.000000	0.99945	2408.372411
2	00:00:05.2003892	00:00:05.2003892	1.0013	71571.693148	1.0011	84018.570076	...	0.99946	3000.0	1.0069	213287.308367	0.99945	2408.372411
3	00:00:06.8981948	00:00:06.8981948	1.0013	71571.693148	1.0011	84018.570076	...	0.99946	3000.0	1.0067	184907.854463	0.99945	2408.372411
4	00:00:07.9217394	00:00:07.9217394	1.0013	71571.693148	1.0011	84018.570076	...	0.99946	3000.0	1.0069	213287.308367	0.99945	2408.372411

Figure 4.3: First five seconds of sampled Tether limit order book data since the midnight of the 1-st of September 2020.

The reconstructed stock data and the sampled crypto data share an equivalent format. Our input pipeline is designed around the BookConstructor output format, so Tether data is firstly translated into the stock format, and then used in the exact same way in our code.

4.2 Empirical Properties of Limit Order Book data

In this section we will be presenting some statistical properties from our data, and how some of these properties impact some characteristics of the input pipeline and model parameters, most notably sampling time.

There's extensive literiture on empirical properties of Limit Order Book data. Cont (2011), Cont and De Larrard (2012), Gopikrishnan et al. (2000) all find that order sizes are highly heterogeneus and with heavy-tails. Cont (2011), in particular, finds that order sizes tend to follow a power law. Moreover, in Bouchaud et al. (2002), heavy tailed distribution are shown to provide a good fit for price movements as well.

Since many of the statistical properties of interest are heavily impacted by a single stock's features such as liquidity, stock price, and volatility, we'll be presenting, in this section, properties from a single stock, Microsoft, with data from all days at our disposal.

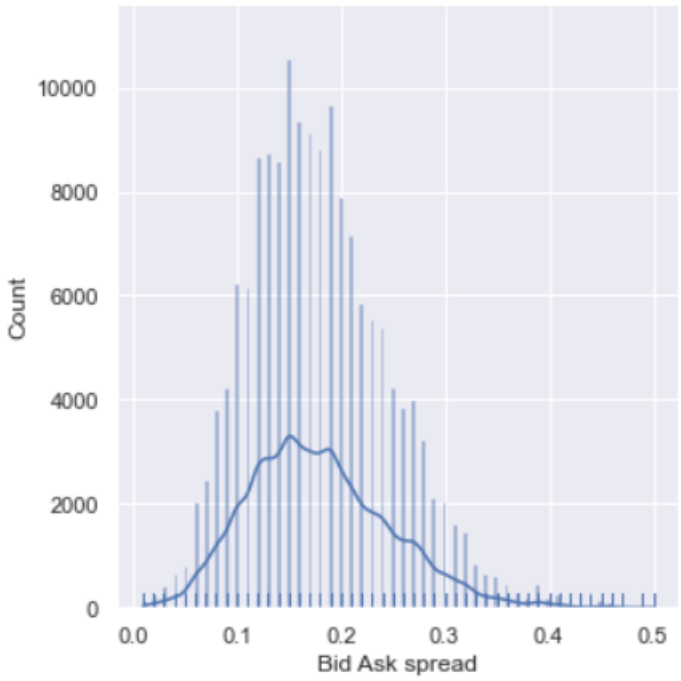


Figure 4.4: Bid-Ask spread distribution plot. Data from Microsoft stock.

The bid ask spread throughout our dataset doesn't present any particular feature, apart from the left skewness that is to be expected with positive prices. Of course, as it is possible to see in the distribution plot, the bid-ask spread can only assume a discrete amount of values, i.e. the possible price levels / tick sizes.

As previously mentioned, there's plenty of evidence for heavy tails in price quotes' movements. We confirm this by analysing the absolute non-zero movements of best bid and best ask in our data. This is because, as we will see in the next section, the overwhelmingly majority of order book updates doesn't change neither the best bid nor the best ask. Moreover, movements are reported by number of levels.

For both bid and ask, respectively in Figure 4.5 and in Figure 4.6, the distribution plot and the boxen plot are shown. A boxen plot, introduced by Hofmann et al. (2011), is a generalization of a boxplot to properly visualize a large, heavy tailed distribution, in which what would be defined as "outliers"

for a boxplot make up a large percentage of data, thereby hampering visualization. A boxen plot works by determining which quantiles, apart from the 50% quantile, which is always used, to use in order to fill a pre-determined number of boxes. The height of the boxes is proportional to the level of the quantile, i.e. the box corresponding to the median is the one with greatest height.

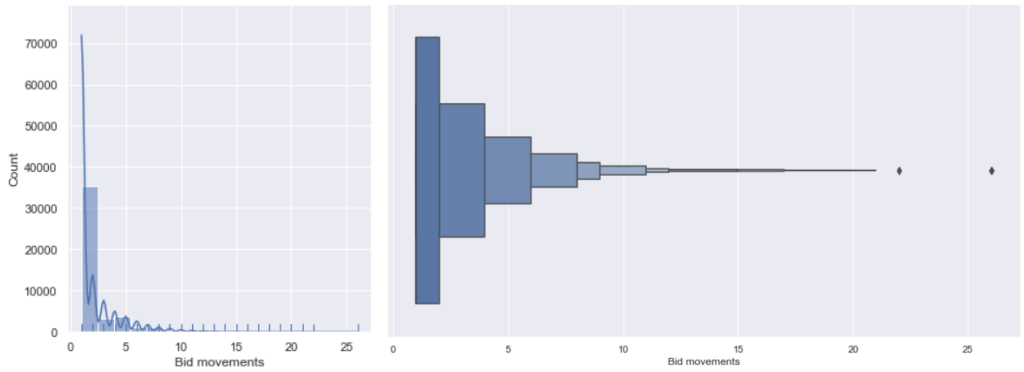


Figure 4.5: Distribution plot and boxen plot for absolute non-zero movements of best bid. Data from Microsoft stock.

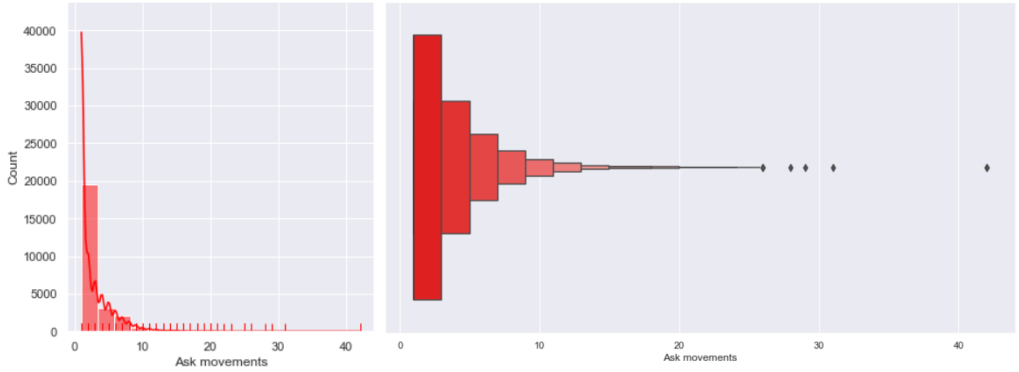


Figure 4.6: Distribution plot and boxen plot for absolute non-zero movements of best bid. Data from Microsoft stock.

As seen in the distribution plots, the overwhelmingly majority of movements correspond to a 1-tick movement, and all other movements have a decreasing occurrence. This is to be expected by high-frequency data, as prices naturally tend to move in a continuous fashion, which in this discrete framework

means that they tend to move by the smallest possible tick sizes.

In the boxen plots, heavy tails are clearly recognizable by the shape of the last boxes, with a very low height, indicating that it represents a very low quantile, and by it's elongated shape, suggesting that it contains a large range of observations.

Heavy tails in the best bid prices and best ask prices suggest that a large range of possible movements must be included in the predictions, however rare they might be in the training set. As we will see, the benchmark model we will be comparing our model with falls quite short in this regard, as the possible range of price movements must be specified before training.

4.2.1 Data sampling

In this section we will analyze some reasons behind one of the crucial decision in designing a high-frequency forecasting model: the sampling time.

This would be a significant decision in a regression model, but, for reasons that will be explained in this section, has particular implications for a classification model, most notably in terms of balancing classes.

A 0-levels move of best bid or best ask, i.e. them remaining constant, will be one of the classes of interest. Since, as explained in the previous paragraph, our data contains all available changes in order book data -and not just to best bid and best ask-, sampling is necessary since most updates leave best bid and best bid ask unchanged.
We would like to sample with a frequency such that a 0-level move of best bid or best ask wouldn't make up the overwhelmingly majority of classes, as it would be using all available lines of the book.

At the same time, by sampling, one reduces the number of data available for training, so it is of interest to select a sampling time as short as possible.

To get a measure of how fast a stock changes in price, for every t, we can define a random variable τ, describing the random time associated with a change in the mid price:

Definition 4.2.1 (Mid change time)**.** The mid change time is defined as the time between two movements of the mid price:

$$\tau_t = \inf\{s > 0 \mid mid(t+s) \neq mid(t)\}.$$

An appropriate metric for price changing can then be the median value of τ_t.

We compute the median value of τ_t for all the stocks at our disposal. We then average the results, which are shown in Table 4.1 below. Time is measured in seconds.

Mean	0.010211
Std Dev	0.017052
25%	0.001349
50%	0.004547
75%	0.010612

Table 4.1: Descriptive statistics of median jump times.

Therefore, 0.01 seconds looks like an appropriate rounded value. However, this number should be considered more of as a lower bound for sampling times, rather than a rule of thumb for sampling.

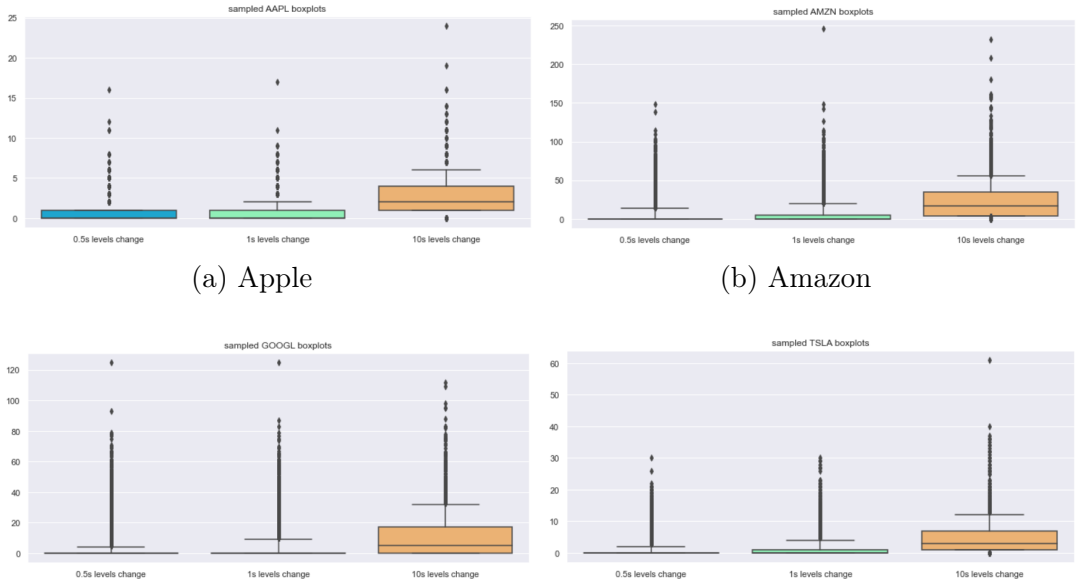
There are two reasons for this:

Firstly the median jump time refers to a change in mid price, which encompasses changes both in the best ask and best bid prices, whereas the our models only train on one of the two at a time.

Secondly, and more importantly, having a median jump time of 0.01 seconds, **doesn't** mean that by using a sample frequency of 0.01 seconds we will obtain roughly 50% realizations of nonzero-levels moves. An uniform sampling negates clustering effects, so if between t and $t + 0.01$ more than one price change occur, that will drive down the median jump time, but it will only count as one move after discretizing at 0.01 seconds.

4.2.2 Stocks sampling

To determine an appropriate sampling time we then discretize our data with sampling times greater than the median jump time, and then compute variation quantiles on the sampled data.
In Figure 4.7 25% − 90% boxplots of levels variations corresponding to various sampling times are shown for selected stocks.



(a) Apple (b) Amazon

(c) Google (d) Tesla

Figure 4.7: Boxplots of four selected stocks.

In Figures 4.8 and 4.9, 25% and 90% quantiles are reported in a uniform scale.
Once sampled at 0.1 seconds, the median move is almost never a non-zero move, and most stocks have only have a 1-level move only as a 90% percentile, with some stocks barely ever moving at that timescale. As in Sirignano (2019), we find that 1 seconds seem to be an appropriate sampling time, allowing enough movement such that classes are not overwhelmingly unbalanced towards the 0-ticks class.

It's worth noticing how there's significant variation between different stocks, that would allow for some space to 'calibrate' appropriate sampling times

on individual stocks. However, in order to produce consistent comparisons, whenever we will be presenting model results across different stocks, we will stick to a 1s sampling time.

Also note that in a real-life production environment, having a fixed sampling time for all stocks would be much more convenient, especially in order to allow multiple-stocks trading strategies.

The trade-off in sampling data is that, by balancing classes, one also reduces the dataset size used for training. In this case 1-second sampling time reduced the available dataset size by around 2 orders of magnitudes.

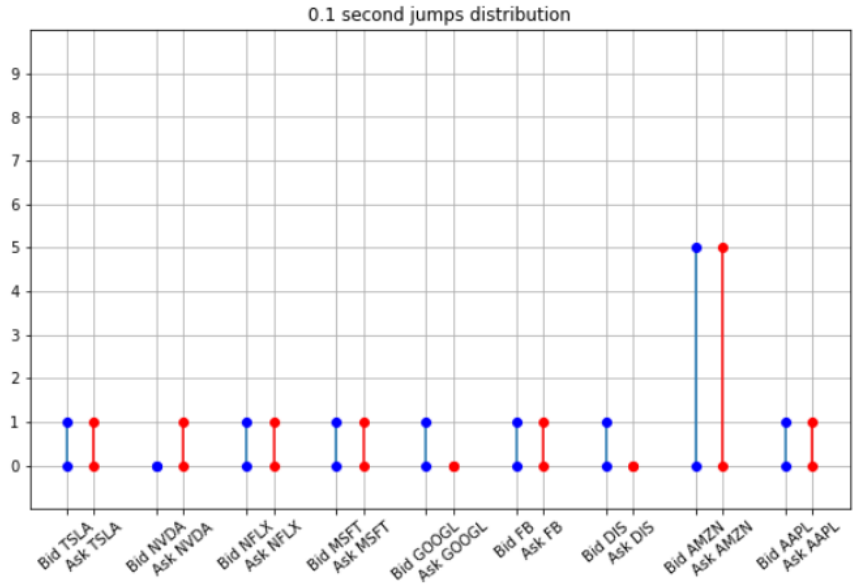


Figure 4.8: (25,90) percentiles for 0.1s-sampled stocks. Blue-colored lines refer to the bid side, whereas red-colored refer to ask side.

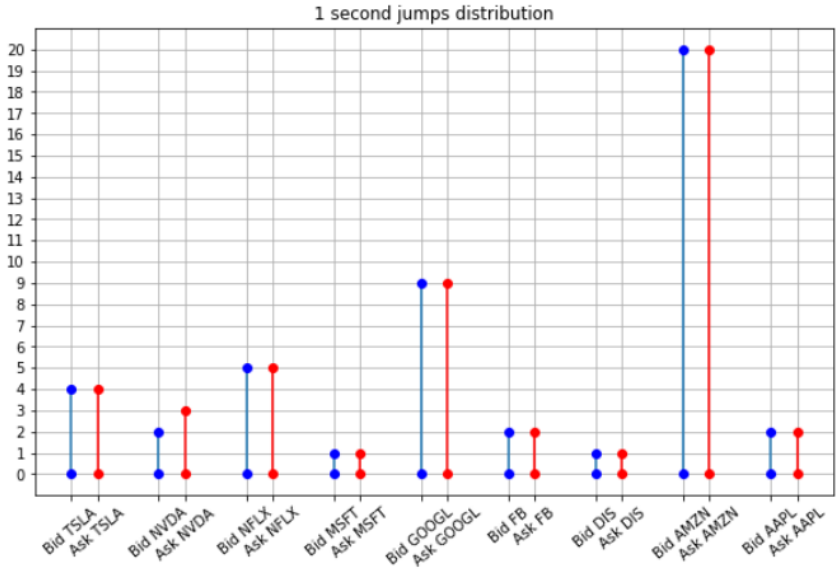


Figure 4.9: [25,90] percentiles for 1s-sampled stocks. Blue-colored lines refer to the bid side, whereas red-colored refer to ask side.

4.2.3 Tether Sampling

Unfortunately, in the case of USDT/USD, in order to obtain the same level of movement such that the classes are not overly unbalanced, it is necessary to sample with a much lower frequency. This is most likely due to the fact that Tether is supposed to be worth almost exactly 1 USD, with BitFinex claiming to have 1 USD in reserve for each USDT issued. Still, since they have no legal -or otherwise- obligations for Tether to be exactly equal to 1 USD, its price can fluctuate to a certain degree due to supply and demand dynamics within the Bitfinex Exchange.

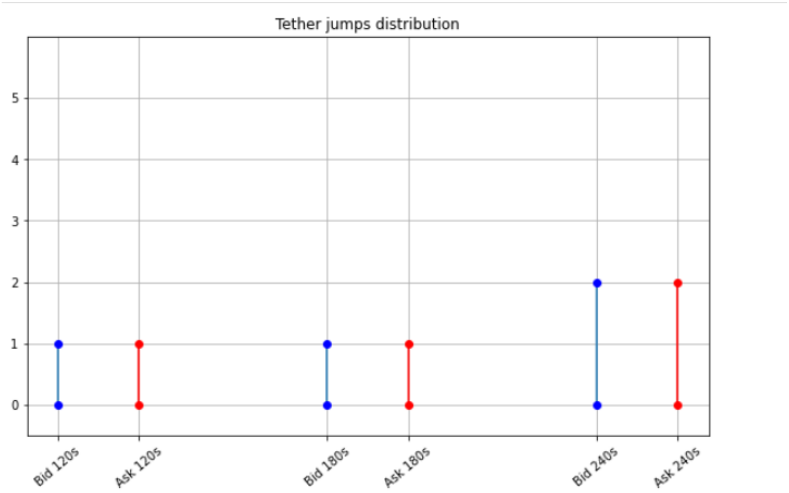


Figure 4.10: (25,90) percentiles of tick changes over three different sampling frequencies. Blue-colored lines refer to thebid side, whereas red-colored refer to ask side.

Given this behaviour, we select 240 seconds as the minimum acceptable sampling time. Note that this decreases significantly the dataset size. In particular, our crypto dataset contains 1739600 datapoints, but after sampling only 9240 remains.

4.3 The role of Liquidity

As in every stock market forecasting study analysis, liquidity plays a crucial role in obtaining robust results. This is because liquidity guarantees market depth and tight bid-ask spreads, as well as reducing significant jumps in the stock price.
Market depth is particularly important in this application as it is desirable that limit orders beyond best bid and best ask play a significant role in determining future best bid and best ask prices. Any kind of information contained deep in the order book can be used by our model. As described in the next section, we will hand-pick features obtained order book in order to try and forecast future best bid and best ask prices.
More importantly, and as we will see, the model used tries to predict the full distribution of next best bid and best ask prices. In order to do so, it uses every limit order size at depth y in order to estimate the probability that the next bid or ask price will move by at least y levels. It is therefore crucial that a stock is liquid to guarantee enough market depth in order to do this task. As one would expect, price variation its more frequent if the stock is liquid. We use market capitalization a proxy for liquidity, as in Bariviera (2011).

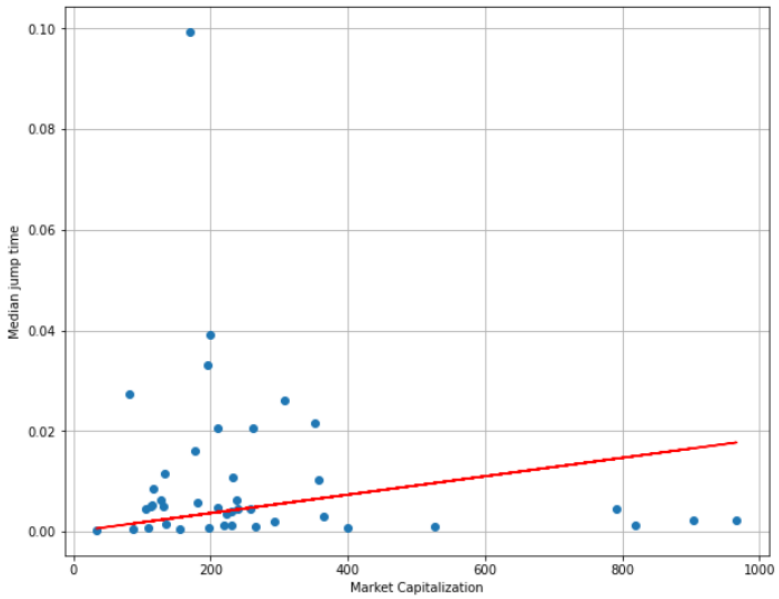


Figure 4.11: Linear Regression of median jump times over market capitalization. data refer to 200B+ capitalized companies listed on NASDAQ.

The linear relationship confirms the general idea that a liquid stock is a better candidate to apply our model to. This is particularly true in this particular application, where the sampling time could -in principle- be selected to be lower for liquid stocks, as they are associated to more frequent movements. Note that the same apply to volatility, so historically more volatile stocks will tend to move more in a given span of time than less volatile stocks.

4.4 Feature Selection

4.4.1 Feature engineering

Order book data contains more information than the simple timeseries of the prices we want to predict, as it contains volumes and prices for all levels beyond the first level. However it is not straightforward how this information is relevant and how it should be fed to a neural network.

In Cao et al. (2009), using data from the Australian Stock Exchange, the authors assess the information content of an open limit-order book with a particular focus on the incremental information contained in the limit orders behind the best bid and offer. The authors find that the order book is moderately informative—its contribution to price discovery is approximately 22%. The remaining 78% is from the best bid and offer prices on the book and the last transaction price. Furthermore, the authors find that order imbalances between the demand and supply schedules along the book are significantly related to future short-term returns. As we will see later in this section, order imbalances will be crucial features for our model.

Our model makes use of the entire book, in two different ways.

Firstly, as we introduced in Chapter 3 and will be explained better in the next Chapter, we use all volumes and prices in the books when predicting by how much the price will change, making use of spatial relationships in the data. This is done via two neural networks, predicting respectively $P(Y^{A/B} = y|Y^{A/B} > 0)$ and $P(Y^{A/B} = y|Y^{A/B} < 0)$. These two networks make use of all the order sizes in the book.

However, there's a third neural network, that tries to predict whether the $Y^{A/B}$ will be either positive, negative, or zero,i.e. :

$$P(Y^A \in A|X = x) \quad A \in (\{Y = 0\}, \{Y > 0\}, \{Y < 0\}).$$

In this section we will be exploring what set of features X seems reasonable to include.

In Zhang et al. (2019) the authors pose themselves the same target, i.e. forecasting the price general direction. To do so, they use a convolutional

layer that takes as input the whole book, and automatically selects relevant features, passing them over to an LSTM layer which then classifies the move.

However, we consider training a CNN layer to be an unnecessary use of computational power for our circumstances. This is because there's extensive literature of non-linear transformations of order book data that provides some kind of predictive power, such as order book imbalances. We therefore-hard coded many transformations of LOB data, and subsequently used feature-selection methods to rule out insignificant features. We now list out all features that ended up being selected as an input to our model.

4.4.2 Comulated volumes

The first selected feature, as used by Sirignano and Cont (2019) is simply the cumulated levels of bid and ask sizes. We propose a slight difference, namely where the information about the trading day is included.

Definition 4.4.1 (**Cumulated volumes**)**.** For every price level, they are defined as the cumulated order size for that price level, since the start of the trading day:

$$CV_k^{A/B}(t_j) = \sum_{i=start\_day}^{j} V_k^{A/B}(t_i),$$

where $start\_day$ is the index corrisponding to the start of the trading day relative to the time t_j.

In the case of cumulated volumes is important to keep track of which day a given time refers to, given that during preprocessing data are shuffled in time. As simple as it is, we will see that it consistently holds more predictive power than its non-cumulative counterpart.

4.4.3 Volume Imbalance

In recent years, some authors (see, e.g., Cartea et al. (2018) and Yang and Zhu (2016)) have proposed that the queue imbalance, which describes the difference between the volumes offered for purchase or sale at the best bid and ask quotes in a limit order book (LOB), could constitute a simple yet powerful quantity that is suitable for this purpose. In particular, Gould and

Bonart (2016), find that volume imbalances increases accuracy as much as 60% w.r.t. to a null model for liquid stocks.

Definition 4.4.2 (**Volume Imbalance**)**.** Volume Imbalance is defined as the percentage difference between the best ask order size and the best bid order size:

$$VI = \frac{V_0^A - V_0^B}{V_0^A + V_0^B}.$$

Volume imbalance can be defined for several levels, with analogous meaning to the first level imbalance, i.e. measuring the imbalance between two symmetric levels, on the bid side and on the ask size:

$$VI_k = \frac{V_k^A - V_k^B}{V_k^A + V_k^B}.$$

In particular, Qureshi (2018) finds that deeper levels imbalances has informative value for predicting mid price movements.

4.4.4 Weighted Price

In order to combine price and volumes imbalance, Cao et al. (2009), propose a weighted price defined as:

Definition 4.4.3 (**Weighted price**)**.** The Weighted Price is analogous to the bid-ask spread, but where the best bid and best ask are weighted by their relative order size:

$$WP_k = \frac{V_k^A P_k^A - V_k^B P_k^B}{V_k^A + V_k^B}.$$

4.4.5 Order Flow Imbalance

Similarly to Cont et al. (2014), the last feature included in the input pipeline represents a less static feature, encompassing and summarizing a number of events in a temporal frame, rather than providing a quantity referring to a snapshot of the book, such as the other features included.
Order flow imbalance represents the net order flow at the 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 decrease.

Interestingly, this variable treats a market sell and a cancel buy of the same size as equivalent, since they have the same effect on the size of the bid queue. In particular, bid price and size represent the demand for a stock, while the ask price and size represent the supply. Indicating by $P_t^{B/A}$ the best bid/ask price at time t and by $V_t^{B/A}$ the best bid/ask price order size at time t , between two data points a number of events could happen:

- $P_t^B > P_{t-1}^B$ or $V_t^B > V_{t-1}^B$ signifying an increase in demand
- $P_t^B < P_{t-1}^B$ or $V_t^B < V_{t-1}^B$ signifying a decrease in demand
- $P_t^A > P_{t-1}^A$ or $V_t^A > V_{t-1}^A$ signifying an increase in supply
- $P_t^A < P_{t-1}^A$ or $V_t^A < V^A t - 1$ signifying a decrease in supply

Finally, we define our own version of Order Flow Imbalance as:

Definition 4.4.4 (**OFI**)**.** The order flow imbalance is an engineered metric of supply and demand movements in time:

$$OFI(t) = I_{P_t^B \geq P_{t-1}^B} V_t^B - I_{P_t^B \leq P_{t-1}^B} V_{t-1}^B - I_{P_t^A \geq P_{t-1}^A} V_t^A + I_{P_t^A \geq P_{t-1}^A} V_{t-1}^A.$$

Note that if V^B increases but P^B remains the same, the contribution from the bid size is $V_t^B - V_{t-1}^B$, representing the size that was added at the bid. If V^B decreases, this contribution is also V_t^B, but representing the size that was removed from the bid, whether due to a market sell or cancel buy order. If P^B increases, we let this contribution becomes V_t^B representing the size of a price-improving limit order. If P^B decreases, we let then the contribution be V_{t-1}^B, representing the size that was removed, whether due to a market order or a cancellation. The contributions for events on the ask side are analogous, with signs reversed.

4.5 Feature Importance

In order to determine which features from the LOB should be part of the input of our model, we make use of a very popular method to select features in Machine Learning, namely Random Forests.

4.5.1 Decision trees

Random forests are based on decision trees. A decision tree determines a number of cutoffs for each of the features, in order to build a binary tree to classify the outcome. A decision tree determines not only the cutoffs to use, but the order in which to use features while it trains on the data.

In Figure 4.12 a tree trained on Amazon stock, as well as the head of the training data. The training data contains a simplified version input to the one we will use later in our architecture. It contains two features, namely best ask order size and volume imbalance, and its forecasting target is the direction of the best ask price move.

V_0^A	VI_0	P_0^A move
200	10.1%	DOWN
700	52.1%	DOWN
1200	-40.5%	CONST
500	-19.7%	UP
300	-5.3%	UP

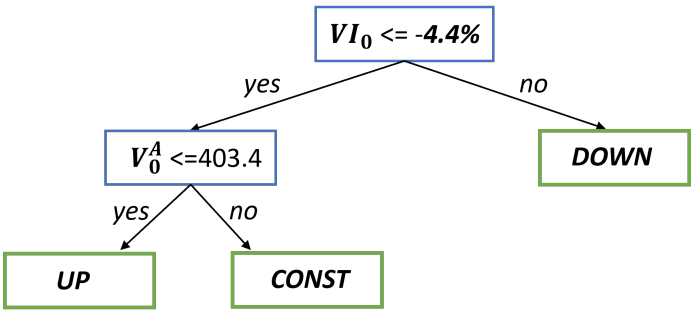


Figure 4.12: Decision Tree classifier trained on a simplified input. Data from 30/01/2019 Microsoft stock.

4.5.2 Random Forests for feature selection

Random forest consists of a large number of decision trees. Each tree is trained on a bootstraped sample of the dataset. This sample contains a random subsample of rows, with rows repeated to match the original length of the dataset. The rows that were not included constitute the 'out-of-bag' sample. Out-of-bag samples can then be used to measure the accuracy of the forest.
In Figure 4.13 below, the dataset in Figure 4.12 is bootstrapped in two bags.

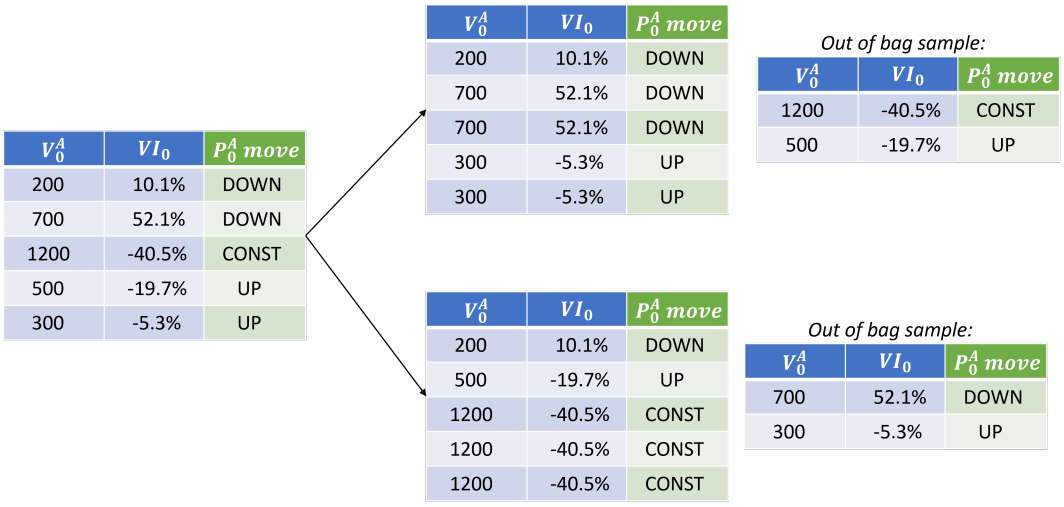


Figure 4.13: Visualization of data bootstrapping.

To measure the importance of the j-th feature after training, the values of the j-th feature are permuted among the training data and the out-of-bag error is again computed on this perturbed data set. The importance score for the j-th feature is computed by averaging the difference in out-of-bag error before and after the permutation over all trees. The score is normalized by the standard deviation of these differences.

Features which produce large values for this score are ranked as more important than features which produce small values. The statistical definition of the variable importance measure was given and analyzed by Zhu et al. (2015).

On each of the stocks, we fit a random forest classifier and then rank features importances. Below, in Figure 4.14 is the output for the forest trained on Microsoft data.

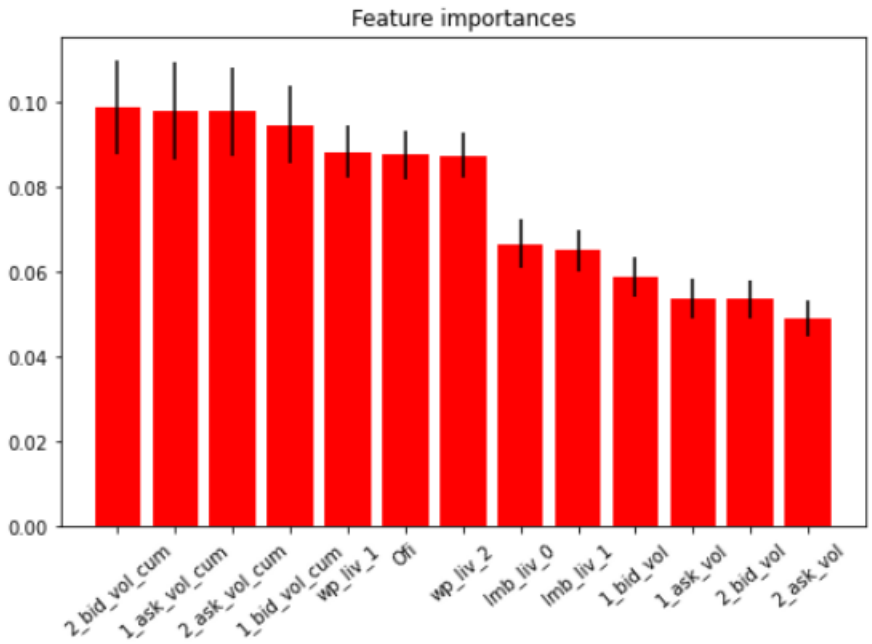


Figure 4.14: Features importances computed by fitting a random forest classifier.

Features importances somewhat varies between stocks, but 'vanilla' volumes are consistently overperformed by other features. For 100% of the stocks in set B, i.e. the one containing the stocks used to train the models, the 'vanilla' volumes are always the least important features, altho the skew of the 'importance gradient' in Figure 4.14 varies.

Interestingly enough, cumulative volumes play an important role. This is probably due to the fact that it introduces a long term dependence in the input, by keeping an account on cumulative buying and selling pressure over the course of a given trading day.

Some features of the ones just mentioned can be indexed to include order book information deeper than the best bid and best ask prices. As stated, Cao et al. (2009) find that this information is consequential. We therefore use this information, and throughout this work we will be indexing all indexable features up to $k = 5$ levels. This allows us to make use of some information, without -possibly unnecessarily- increasing to much the dimension of the in-

put. Note that in Figure 4.14 features are indexed up to $k = 2$ levels for better readability.

Chapter 5

The model

In this section we will be explaining how the model is structured. As explained in Chapter 3. the model is not simply composed by a neural network, but by 6 neural networks, 3 for the bid side, and 3 for the ask side, that work together to forecast the next price move.

5.1 Standard Neural Network for classification

The basic neural network for classification is a highly nonlinear parameterized function that takes an input $x \in X$ and produces a probability distribution on the finite discrete space Y. This will be the baseline model that we will be comparing our model with.
Given an input x, the output of the k-th layer of a neural network is:

$$R^{d_k} \ni f_{\theta,k}(x) = g_k(W_k f_{\theta,k1}(x) + b_k), \quad k = 1, ..., K,$$

where $W_k \in R^{d_k} R^{d_{k-1}}$ is a linear combination of the previous layer' output , the bias $b_k \in R^{d_k}$, $f_{\theta,0}(x) = x$, and $d_K = |Y|$.

For $k = 1, ..., K1$, the nonlinear transformation g is defined as:

$$g(z) = (\sigma(z_1), ..., \sigma(z_{d_k})) \quad for\ z \in R^{d_k} \quad and\ z_1, ..., z_{d_k} \in R.$$

The functions σ are nonlinear; typical choices are sigmoidal functions, tanh, rectified linear units (ReLU), and clipped rectified linear units. The function g_L for the final layer L is the softmax function g.

$$g(z) = \left(\frac{e^{z_1}}{\sum_{i=1}^{d_K} e^{z_i}}, .., \frac{e^{z_K}}{\sum_{i=1}^{d_K} e^{z_i}} \right) \quad z \in R^{d_K}.$$

In our application, the output of the neural network is a probability distribution on Y conditional on the features x. Therefore, for every class y_i, the estimated probability that Y will assume the value of the k-th class, conditioned on the observed feature x:

$$f_\theta^k(x) = P_{est}(Y = y_k) \quad k = 1, .., N_{classes},$$

where $f_\theta^k(x)$ is the k-th component of the output vector $f_\theta(x)$.

A very commonly used loss function is the categorical loss-entropy, which measures the distance between the estimated probability distribution, and the observed probability distribution. Given the set of all features-observation couples $D = \{(x_i, y_i\}_{i=1,..,N_{samples}}$, the overall loss $L(D)$ is:

$$L(D) = \frac{1}{N_{samples}} \sum_{i}^{N_{samples}} \sum_{k}^{N_{classes}} log\ f_\theta^k(x)) 1_{y_k = y_i}.$$

The standard neural network for classification has two major drawbacks in this particular application. The classes refers to possible movements $Y^{A/B}$ of either the best bid or the best ask price. Therefore $Y^{A/B} \in (.., -2, 1, 0, 1, 2..)$ and so they can assume an infinite amount of values.

However, the standard neural network for classification can only handle a finite number of classes. Therefore, the possible movements of $Y^{A/B}$ have to be truncated within a finite range in order for the model to be trained. This is inconvenient because large movements of the best bid and best ask are important for risk management purposes.

Moreover, one incurs in an inconvenient trade-off when determining the number of classes $N_{classes}$. A realistic model should have a reasonable amount of classes, but the more classes one chooses to include, the greater the training time is, since the dimension of the output is increased. This is greatly amplified if one wants to model the joint distribution of bid of ask, in which case $f_\theta(x) \in (.., -2, 1, 0, 1, 2..)^2$.

The real problem of truncating the number of classes is that, in the case of the standard neural network, the truncation must be done before training. Once specified and trained, the standard architecture can only be used to predict within that pre-defined range, and so has little flexibility in a production environment. If for instance, the trained model is being used to predict real-time data, and the market happens to be particularly volatile, the range of possible movements to predict might need to be extended. However, the standard neural network does not offer the flexibility for this to be done in a timely manner. As we will see later in this Chapter, this is not the case for our model.

A second drawback of a standard neural network ,as previously mentioned, is that it fails to model spatial correlation of $Y^{A/B}$. If the order of the classes is shuffled, little changes in training the model. However, the events that $Y^{A/B}$ moves by $y_1 = 1$ levels or $y_2 = 2$ levels are more greatly correlated than Y^A moving by $y_1 = 1$ levels or $y_{17} = 17$ levels, so the order of the classes is crucial.

5.2 Spatial Neural Network

The architecture proposed by Sirignano (2019), solves all the drawbacks of the standard neural network for classification. It does this by modelling a conditional probability of a local event rather the full probability. We use two models of this type, one for both possible direction of the next bid price.

Definition 5.2.1 (**Upper spatial network**)**.** It outputs the probability of the next price movement will be equal to y conditioned on the event that it will be greater or equal than y

$$f^{spatial,up}(x, y) = P_{est}(Y^{A/B} = y | Y^{A/B} \geq y) \quad y \in (1, 2, 3, ..).$$

Definition 5.2.2 (**Lower spatial network**)**.** It outputs the probability of the next price movement will be equal to y conditioned on the event that it will be lower or equal than y

$$f^{spatial,down}(x, y) = P_{est}(Y^{A/B} = y | Y^{A/B} \leq y) \quad y \in (.., -3, -2, -1).$$

By doing this, one ensures to make use of spatial relationship in the data, such as the one explored in Chapter 3. Moreover by including the y as a feature, the model can keep track of spatial correlation between different events.

Spatial Neural Network dataset preparation

The most obvious advantage of this architecture is that it can practically take into account all observed movements y. Given a vector of movements $Y^{A/B}$, it being either the best ask or best bid movements, the dataset for positive movements $Y^{A/B} = y|Y \geq y$ is assembled as follows:

Algorithm 1 Spatial Network data preparation

```
y_max = max(Y)
train_data = [ ]
y ← 1
i ← 1
while y ≤ y_max do
    while i ≤ N_samples do
        if Y[i] ≥ y then
            if Y[i] = y then
                train_data.append(1, x, y)
            else
                train_data.append(0, x, y)
            end if
        end if
        i ← i + 1
    end while
    y ← y + 1
end while
```

where x represent the feature relative to the event $Y^{A/B} = y|Y \geq y$.
In Figure 5.1, a visualization of the input is shown. The whole dataset is scanned to include all realization of $Y^{A/B} = y|Y^{A/B} \geq y$ for which a label 1 is given, and $Y^{A/B} > y \mid Y^{A/B} \geq y$ for which a value of label 0 is given. This is done iteratively for all possible values of y, therefore without truncating in the training phase.

1	x_1	y=1
0	x_2	y=1
..	...	...
1	x_k	y=k
...	...	...
0	x_{n-1}	y=y_max
1	x_n	y=y_max

Figure 5.1: Visualisation of the Spatial neural network input data.

5.3 Our model

As previously mentioned, starting from probability of events of the type $P(Y^{A/B} = y|Y \geq y)$ it is possible to reconstruct $P(Y^{A/B} = y|Y \geq 0)$, and similarly for the negative side, by using Lemma 3.3.1.
Therefore,for each side of the order book, i.e. bid and ask, there's a need for 3 models, 2 for the tails and one required to "stitch" together $Y > 0$ and $Y < 0$.

Definition 5.3.1 (**Middle model**)**.** It outputs the probability that the next price movement will be either positive, negative, or null.

$$f^{mid}(x, A) = P_{est}(Y^{A/B} \in A|X = x) \quad A \in (\{Y^{A/B} = 0\}, \{Y^{A/B} > 0\}, \{Y^{A/B} < 0\}).$$

.

Definition 5.3.2 (**Upper tail model**)**.** It outputs the probability that the next price movement will amount to exactly y levels, conditioned on the fact of y being positive.

$$f^{tail,up}(x, y) = P_{est}(Y^{A/B} = y|Y^{A/B} > 0, X = x) \quad y \in (1, 2, 3, ..).$$

.

Definition 5.3.3 (**Lower tail model**)**.** It outputs the probability that the next price movement will amount to exactly y levels, conditioned on the fact of y being negative.

$$f^{tail,down}(x,y) = P_{est}(Y^{A/B} = y | Y^{A/B} < 0, X = x) \quad y \in (..,-3,-2,-1).$$

.

In Figure 5.2 it is shown a visualisation of how our models reconstruct the estimation of the next price move. It is can simply done by multiplication:

- $y > 0$: $P_{est}(Y^{A/B} = y, X = x) = f^{tail,up}(x,y) f^{mid}(x, \{Y^{A/B} > 0\})$
- $y = 0$: $P_{est}(Y^{A/B} = y, X = x) = f^{mid}(x, \{Y^{A/B} = 0\})$
- $y < 0$: $P_{est}(Y^{A/B} = y, X = x) = f^{tail,down}(x,y) f^{mid}(x, \{Y^{A/B} < 0\})$

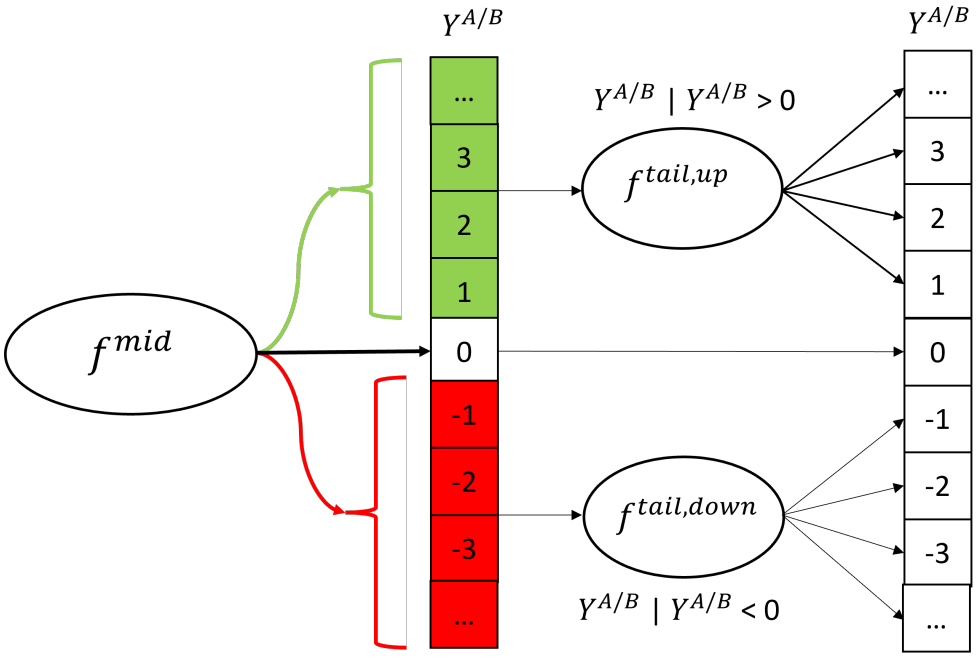


Figure 5.2: Visualisation of the way our model reconstruct the probabilities of the movements of either the best bid or the best ask.

Note that the two f^{tail} models are not neural networks, but rather composition of outputs of spatial neural networks, according to Lemma 3.3.1.

For instance,

$$f_{ask}^{tail,up}(x,y) = f_{ask}^{spatial,up}(x,y)\prod_{k=1}^{y-1}(1 - f_{ask}^{spatial,up}(x,k)), \tag{5.1}$$

where $f_{ask}^{spatial,up}(x,y)$ is a neural network that will be specified shortly.

The modularity of our model, i.e. being composed by three neural networks per side [1], is motivated by making use of spatial relationship in the data. However, as we will see later in this section, it offers greater flexibility in predicting several scenarios of interest. Of course, this come at the increased computational cost of having three models to train instead of just one.

5.3.1 Mid neural network

The first neural network to come into play has the most crucial role, i.e. to determine the general direction of the next price movement. We use a standard neural network for classification to do this with a 3-dimensional output, but using the engineered features presented in Section 3.

The input for the bid side Y^B is similar to the input for the ask side Y^A. The same type of features are used .

- **Bid side**: For predicting the next bid movement Y^B, we use the features describing the global state of the order book.

$$f_{bid}^{mid}(x,\cdot) = f_{bid}^{mid}(\textsc{BookFeatures}).$$

- **Ask side**: For predicting the next ask movement Y^A, along with the book features, we also use the observed bid movement Y^B, so that we are able to model the joint distribution of bid and ask.

[1] Three networks for predicting $P(Y^A = y)$ and three for $P(Y^B = y)$

$$f_{ask}^{mid}(x,\cdot) = f_{ask}^{mid}(\textsc{BookFeatures}, Y^B).$$

Both of these functions are identical neural networks. Similarly to Sirignano (2019) use $K = 4$ layers, 250 neurons per layer, and *tanh* activations functions.

5.3.2 Tail neural networks

There are four spatial neural networks to train, one for both the two sides, and the ask and bid.

- **Upper Bid side**: The features for estimating the probability of $Y^B = y|Y^B \geq y$ are the order sizes on the ask side, as mentioned in section two, so that:

$$f_{bid}^{spatial,up}(x,y) = f_{bid}^{spatial,up}(V_y^A, y).$$

- **Upper Ask side** The features for estimating the probability of $Y^A = y|Y^A \geq y$ are the order sizes on the ask side, as well as the observed bid movement Y^B:

$$f_{ask}^{spatial,up}(x,y) = f_{ask}^{spatial,up}(V_y^A, Y^B, y).$$

- **Lower Bid side**: The features for estimating the probability of $Y^B = y|Y^B \leq y$ are the order sizes on the bid side, so that:

$$f_{bid}^{spatial,down}(x,y) = f_{bid}^{spatial,down}(V_y^B, y).$$

- **Lower Ask side** The features for estimating the probability of $Y^A = y|Y^A \leq y$ are the order sizes on the bid side, as well as the observed bid movement Y^B:

$$f_{ask}^{spatial,down}(x,y) = f_{ask}^{spatial,down}(V_y^B, Y^B, y).$$

All of these functions are identical neural networks. Similarly to Sirignano (2019) use $K = 4$ layers, 250 neurons per layer, and *tanh* activations functions.

5.4 Benchmark model

In order to evaluate our model, we compare it with the standard neural network for classification, as visualized in Figure 5.3. To do so, as mentioned earlier, we need to truncate the number of classes. To do so, we fix a symmetric range in which to predict, namely y_{max}, so that the number of classes $N_{classes} = 2y_{max} + 1$.

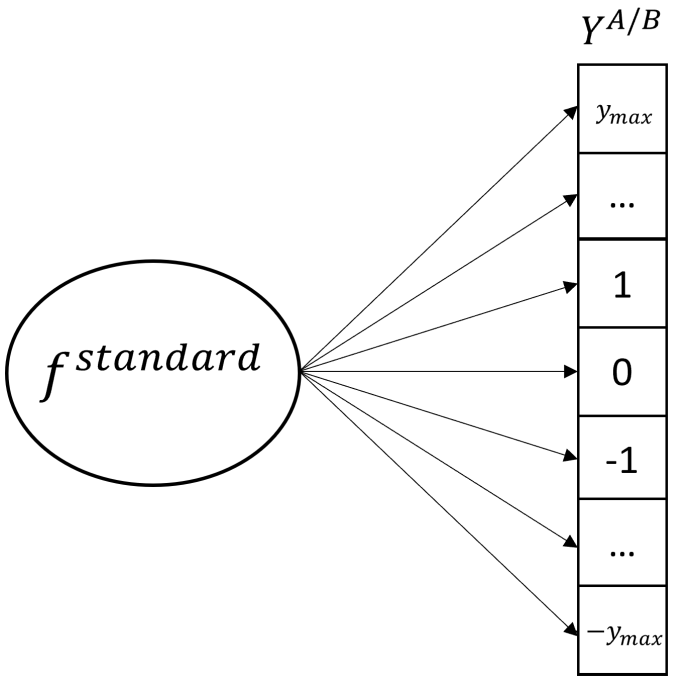


Figure 5.3: Visualization of the standard neural network for classification. It directly outputs the probability for all possible movements, within a range, of either the best bid or the best ask.

For the benchmark model, we use again two separate models for the bid and the ask side. The two models share the same feature, i.e. all the order sizes from the book, up to the desired depth y_{max}. However, as for our model, we include the best bid movement as a feature for predicting the best ask movement.

- **Bid side**: For predicting the next bid movement Y^B, we use all order sizes both from the bid size and the ask size.

$$f_{bid}^{standard}(x) = f_{bid}^{standard}((V_y^B, V_y^A)_y)) \quad y = 1, .., y_{max}.$$

- **Ask side**: For predicting the next ask movement Y^A, along with the book order sizes, we also use the observed bid movement Y^B, so that we are able to model the joint distribution of bid and ask.

$$f_{ask}^{standard}(x) = f_{ask}^{standard}((V_y^B, V_y^A)_y), Y^B) \quad y = 1, .., y_{max}.$$

We use the same networks parameters as all our model, except of course the number of classes, so we use $K = 4$ layers, 250 neurons per layer, and *tanh* activations functions for both of these networks.

5.5 Predictions using our model

As mentioned, our model, when it comes to predicting new data, offers greater flexibility than the standard model. The upper and lower model are not trained neural networks, but rather output their probability estimations from the trained spatial neural network according to equations of the the same type of Equation 5.1.

In this section, we will analyze some features that characterises making predictions with our modular model, and how they compare with the standard neural network for classification.

5.5.1 Predicting in a greater range than training

The functions $f^{tail}(x, y)$ can be used with every y, once the networks $f^{spatial}(x, y)$ are trained. This means that the range of values to predict in, which is called y_{max} during fitting, does not need to be the same than the one used in the training phase.
In other terms, one can compute, for instance:

$$P_{est}(Y = \overline{y}|Y > 0, X = x) = f^{spatial,up}(x, \overline{y}) \prod_{k=1}^{\overline{y}-1} (1 - f^{spatial,up}(x, k)),$$

where we allow $\overline{y}$ to be greater than the y_{max} used during training.

This is not possible for the standard neural network, which is forced to predict values in the same range as it was trained in.

Anyway, for computational reasons it is necessary to fix a range in which to predict, after training. From now on, we will be referring to this range as y_{range}. Since $Y^{A/B} \in Z$, nothing guarantees that the truncated distribution will still resembles a probability distribution, i.e such that:

$$\sum_{y=-y_{range}}^{y_{range}} P_{est}(Y^{A/B} = y) = 1.$$

However, that sum can get arbitrarily close to 1 by choosing y_{range} large enough, i.e. the model is well posed. This was shown in Sirignano (2019). All our estimated probabilities output always sum to 1 by using y_{range} values around 25.

5.5.2 Scenario Predicting

Moreover, many scenarios that might be of interest can be evaluated in a straightforward manner by our model. For example, a broker willing to execute a large sell order might be interested in only one side of the bid queue, i.e. by how much the best bid will increase, *conditioned* on the fact that it will increase. These scenarios include:

- **Best Ask increasing**:

$$P_{est}(Y^A = y | Y^A > 0, X = x) = f_{ask}^{tail,up}(x, y) \ .$$

- **Best Bid increasing**:

$$P_{est}(Y^B = y | Y^B > 0, X = x) = f_{bid}^{tail,up}(x, y) \ .$$

- **Best Ask decreasing**:

$$P_{est}(Y^A = y | Y^A < 0, X = x) = f_{ask}^{tail,down}(x, y) \ .$$

- **Best Bid decreasing**:

$$P_{est}(Y^B = y | Y^B < 0, X = x) = f_{bid}^{tail,down}(x, y) \ .$$

Of course, the standard neural network can also be used to calculate these probabilities, by composing the probabilities of the relevant singleton events. However, the 4 different spatial networks $f^{spatial}$, from which the 4 different tail functions f^{tail} are derived, are trained specifically for these types of events, and are therefore more reliable.

5.5.3 Predicting joint events

With our model, we are able to model the full probabilities of bid and ask, i.e.

$$P(Y^A = y) \text{ and } P(Y^B = y), \;\; y \in (-y_{range}, .., -1, 0, 1, .., y_{range}),$$

but we are not modelling the joint distribution directly, i.e. we are not computing:

$$P_{est}[(Y^A, Y^B) = (y^a, y^b)], \;\; (y^a, y^b) \in (-y_{range}, .., -1, 0, 1, .., y_{range})^2.$$

We also cannot obtain $P_{est}[(Y^A, Y^B) = (y^a, y^b)]$ by multiplicating the probabilities, as we have supposed non-indipendence between bid and ask, and we are using the bid movements as a feature for the ask movement.

On the contrary, we predict joint events in a two different steps: we first predict Y^B_{pred} by selecting the most likely scenario from the bid distribution, and then us it as a feature in order to predict Y^A_{pred}.

This allows us to compute an accuracy metric on the joint events, but not to produce a loss with respect to the joint probability distribution.

Chapter 6

Experiments

In this chapter we will explain how we have set our experiments, as well as presenting the out of sample results of our models.

6.1 Training Neural Networks

The problem of training is equivalent to the problem of minimizing the loss function, i.e. given a set of training features and labels $\{(x_i, y_i)\}_{i=1,..,N_{samples}}$, find weights $\overline{\theta}$, such that when applied to the neural network $f(x)$:

$$\overline{\theta} = argmin_{\theta} L(\{(f_{\theta}, x_i), y_i\}_{i=1,..,N_{samples}},$$

where L is the loss function.

6.1.1 Back Propagation

The task of minimizing the loss function is usually done via an algorithm known as back-propagation. Back-propagation works by propagating the loss from the last layer to the first, each time evaluating the derivatives of the loss function with respect to the nodes of a given layer. At the end of this procedure, the weights are updated.

6.1.2 Updating the weights

Updating the weights is a matter of optimization, and many different algorithms are available. Most of these, however, are a generalization of the basic

Gradient Descend algorithm. The gradient descend works by evaluating the gradient of the loss function and updating the weights in a additive manner, similarly to Newton Methods:

$$\theta_{n+1} = \theta_n + \alpha \nabla L((f_\theta, x_i), y_i)). \qquad for\ i\ \in 1, .., N_{samples}$$

where α is the descend step.

Therefore, each weight is updated after every sample in the dataset. A straightforward generalization of *Gradient Descend*, that works by updating the weights not after a single sample but after a batch of samples. In this work a batch size of 20 is used.

Nowadays, much more sophisticated optimizer are available. In our case, we used the Adam optimizer, as introduced in Kingma and Ba (2014).

6.1.3 Other training parameters

There are several other concepts that contribute to a successful training of a neural network.

- **Epochs**: The maximum number of iterations over the entire dataset. We use 75 epochs.
- **Data splitting**: The dataset is divided into three groups: train set, which is used for the training, validation set which is useful for early stopping, and test set, on which the model is evaluated.
- **Early stopping**: To avoid overfitting, after every epoch the model is used on the validation data and evaluated. Once the validation performance stops increasing, for N epochs in a row, the training is stopped. We use $N = 15$.
- **Dropout**: Another technique to reduce overfitting,it consists in "forgetting" the value of a weight of a given layer, after every epoch, every weight is set to zero with a given probability p. We use $p = 0.3$
- **Data shuffling and repeating**: To increase the number of different batches available, we randomly shuffle the dataset before proceeding with batching. After this, the dataset is repeated, so that it has infinite length.

- **Evaluation interval**: Having an infinite repeating dataset means that we have to specify and evaluation interval N_e, i.e. the number of samples the model trains on in a single epoch. We use $N_e = 10000$

6.2 Out of sample prediction

The test set contains data that were not used for training. It is possible to use the features in the train set to predict the price movements and compare them with the observed ones.
It is then possible to then compute the loss on these out of sample values.
Along with the loss, we also use another metric to evaluate the model, namely accuracy.
Accuracy measures the frequency with which our model correctly predict the observed value.

6.2.1 Equity Out of sample results

We try our models on the test set of all the stocks in Set B, and compare it with the results obtained by the benchmark model.
We use the same parameters across all differents stocks, most notably we predict in a $y_{max} = 25$ range. This value was handpicked looking at the empirical distribution of the stocks in Set B. It isn't always enough to cover all possible realizations, but in order to produce a fair comparison with the standard model, we chose it not to increase too much the standard models output dimension.

For all stocks, we compute the joint accuracy. We do this in a two-steps way as explained in Chapter 5, i.e. predicting first the bid, and then the ask. Then, we can compute the accuracies on the joint events. However, we cannot produce a loss on the joint probability distribution, as we are not modelling that directly. Instead, we compute the sum of the two marginal losses.

Results are shown in Table 6.1 below. For better readibility, we used 'MM' to indicate our Modular Model, as well as 'SNN' to indicate the Standard Neural Network.

	MM Accuracy	SNN Joint Accuracy	MM Loss	SNN Loss
AMZN	0.57	0.57	3.12	3.43
FB	0.52	0.49	2.39	3.52
NVDA	0.57	0.54	2.24	3.07
AAPL	0.56	0.57	2.36	3.29
TSLA	0.60	0.53	4.30	4.51
NFLX	0.48	0.48	2.64	2.91
MSFT	0.63	0.59	2.21	2.25
GOOGL	0.63	0.56	2.17	2.98
DIS	0.73	0.73	1.31	1.33

Table 6.1: Results Summary

For both models, accuracies are quite high. This is because, even when sampled with a 1s sampling time, most moves are 0-moves, i.e. the best bid and the best ask not moving. Therefore, both models tend to assign a high probability to the $(0, 0)$ event, predicting it correctly most of the times.
Another possible metric is the top-k-accuracy, which measures the frequency with which the observed value lies within the top-k most likely events. We use $k = 9$. In Table 6.2, results are shown for our model, and in Table 6.3 for the Standard Network.

	AMZN	FB	NVDA	AAPL	TSLA	NFLX	MSFT	GOOGL	DIS
1	0.57	0.52	0.57	0.56	0.60	0.48	0.63	0.63	0.73
2	0.64	0.67	0.69	0.69	0.67	0.59	0.73	0.70	0.84
3	0.67	0.78	0.78	0.78	0.71	0.66	0.78	0.74	0.90
4	0.69	0.85	0.83	0.86	0.75	0.72	0.82	0.77	0.94
5	0.69	0.89	0.87	0.91	0.77	0.75	0.87	0.78	0.96
6	0.71	0.92	0.90	0.94	0.79	0.79	0.91	0.79	0.97
7	0.72	0.95	0.92	0.96	0.81	0.81	0.93	0.80	0.98
8	0.73	0.96	0.94	0.97	0.82	0.84	0.94	0.81	0.99
9	0.73	0.97	0.95	0.98	0.84	0.86	0.95	0.82	0.99

Table 6.2: Modular Model Accuracies

	AMZN	FB	NVDA	AAPL	TSLA	NFLX	MSFT	GOOGL	DIS
1	0.57	0.49	0.54	0.57	0.53	0.48	0.59	0.56	0.73
2	0.64	0.63	0.65	0.61	0.62	0.55	0.76	0.64	0.84
3	0.67	0.73	0.78	0.75	0.67	0.63	0.91	0.68	0.87
4	0.67	0.79	0.81	0.82	0.70	0.65	0.95	0.72	0.90
5	0.69	0.86	0.83	0.89	0.72	0.69	0.98	0.74	0.94
6	0.72	0.89	0.85	0.92	0.74	0.72	0.99	0.75	0.96
7	0.73	0.93	0.87	0.95	0.76	0.76	0.99	0.77	0.98
8	0.74	0.94	0.90	0.96	0.78	0.78	0.99	0.78	0.98
9	0.75	0.96	0.92	0.98	0.79	0.81	0.99	0.79	0.99

Table 6.3: Standard Neural Network Accuracies

Looking at the top-k-accuracies, it is clear that both models do quite a good job at predicting future events. This is due to the fact that, at this sampling time, big price moves are increasingly less likely to happen, and most moves tend to be just a few levels around the $(0, 0)$ move. Therefore, as we will see later, any naive model producing decreasing probabilities in a neighborhood of $(0, 0)$, with no particular criteria, will achieve high top-k-accuracies.

Something that it is interesting to see is that, for both model, sometimes the top-9-accuracy reaches almost 100%, whereas other times does not. This maybe due to the fact that different stocks have different volatilities, and for the most volatile ones it may be appropriate to use a larger output range. Again, this can easily be done by our model but not by the standard model. but for the sake of these results we used the same range in both models.

Results concerning the comparison between our modular model and the standard model are somewhat mixed: our modular model is outperformed only in one stock, namely Apple. For the rest of the stocks, our model barely outperforms the standard model.

There's another, more trivial, benchmark that could be used, namely the empirical distribution.

Definition 6.2.1 (**Empirical Distribution**)**.** The estimated probability equals the historical frequency:

$$P_{est}(Y^{A/B} = y) = p_y^{emp},$$

where $p_y^{emp} = \sum_{i=0}^{N_{samples}} 1_{Y=y}$.

In particular, we compute the empirical distribution in the validation set. Unfortunately, and dissimilarly from Sirignano (2019) -which this work is based on-, the standard neural network never outperforms the empirical distribution. We didn't report results from the empirical distribution because they are identical to the results from the standard neural network. Therefore, the standard model doesn't manage to learn any fruitful information from the data.

This is surely due to the low amount of data, compared to Sirignano (2019), as well as the high dimensionality of the input. The features of both standard neural networks are all the order sizes of the book. In contrast, the features of the two spatial neural networks are one size at the time, according to the way the architecture is conceived.

It maybe also due to the high dimensionality of the output of the standard model, it being a wide range of possible movements. These classes will be terribly unbalanced, with the 0-class -i.e. no price movement- taking up the large majority of observations, and all others classes with increasingly less observations, up to two orders of magnitude less. In contrast, in our modular models, we never use more than 3 classes in output.

Despite this, we feel that the comparison with the standard model is somewhat invalidated by it not being able to outperform the null model. It very well may be that, given enough data, the standard model will be able to do so and end up performing equally to our model.

What is clear however, is that our modular model needs less data to outperform the null model. However, it barely does so for the majority of the stocks, and surely wouldn't be useful in a production environment, i.e. being used to engage in high-frequency trading.

6.2.2 Tether's out of sample results

We perform the same experiments, with the same parameters and networks with data from the cryptocurrency known as Tether. As mentioned in Chapter 5 the format in which this data comes into is equivalent to the format of our processed data from the NASDAQ exchange, save for the sampling. Therefore, little needs to be changed in our input pipeline in order for the models to be applied to this type of data as well.

In Table 6.4, results are summarised. Again, we use 'MM' to indicate our modular model, as well as 'SNN' to indicate the standard neural network for classification.

	MM	SNN
Loss	3.18	3.18
k=1	0.29	0.29
k=2	0.69	0.60
k=3	0.95	0.88
k=4	0.99	0.94
k=5	0.99	0.99

Table 6.4: Tether Results Summary

Differently from the accuracies achieved by the models trained on equity data, the top-1-accuracy is quite low. This indicates that, once sampled, Tether tends to move more, and a null model won't be able to perform as good. This is not unexpected however: we specifically picked 280s as sampling time to try to maximise movements (without reducing too much the dataset size), without having to account for multiple assets as in the case of the data from NASDAQ. Moreover, we also didn't have a benchmark, since in the equity case we ended up using the same sampling time as in Sirignano (2019), although after performing due diligence to justify its use.

However, the behaviour with respect to the top-k-accuracies is quite similar to the stocks. This is because the same reasoning applies: probabilities of large moves are rapidly decreasing, and any model producing decreasing probabilities will have relatively high top-k-accuracies.

Unfortunately, the results in the case of Tether show that our modular model is not able to outperform the standard neural network, and with it the null empirical model.

This is of course due to the low amount of data points at our disposal after sampling, namely 9240 datapoints. What is interesting to note, however, is that the top-k accuracy increases at a greatest rate for the modular model than for the standard neural network. This doesn't really seem to happen for the NASDAQ data.

Chapter 7

Conclusions and further work

In this Chapter, we draw some conclusions and list some possible ideas for future work.

7.1 Conclusions

As explored in Chapter 6, our modular model somewhat outperforms the standard neural network, which in turn is not able to outperform the null model, i.e. the empirical distribution.

In the case of the cryptocurrency USDT, our model performs the same as the the standard neural network. This is of course due to the very low amount of training data remaining after sampling, with around 10-15 times less available data than the average stock.

In any case, our model performing the same, or not significantly better, than a standard model it's still an acceptable result: the advantages in our model are not necessarily related to its accuracy, but lie also in its flexibility and usability.

The decisions taken in setting up are architecture were driven by a in-depth analysis of order book data: exploring the local behaviour of the order book, the correlation between bid and ask, feature selection, possible choices of sampling time, and general empirical properties of LOB data.

This kind of analysis would be crucial for any other work willing to model LOB price movements with a classification model.

7.1.1 Limits of this work

In producing this work, we were ampered by several problems.
First of all, the low amount of data. Of course, this problem affects nearly all deep learning works. However in our case, we have a particular mismatch between the size of available data, namely a limit order book, and the size of available data after sampling. This is most notable in the case of USDT data. This mismatch makes it difficult to store many limit order books in memory, only for them to be reduced in size by several orders of magnitude before training.

Another problem is what data is available for free on the internet, namely some selected end-of-month data. This means that most data are scattered in time, and the same stock could be having wildly different properties -such as stock price, volatility, etc- between different months. It would be much more convenient to have the same amount of days worth of data, or even less, provided that they are close in time or even contiguous. As we will explain in the next section, the non-contiguity of our data doesn't allow us to propose and try some relevant architectures.

One last problem, that affects all LOB-related forecasting problems, is the presence of hidden liquidity in the limit order book, as explained in Chapter 2. Stock forecasting is already a difficult problem. If, on top of that, models are not allowed to access all possible information, and there's no way to tell if the observed quantities used to predict a given data point are actually very different from the true quantities, as hidden orders may be hampering that.

7.2 Direction of future work

As mentioned in the last section, access to data was quite restricted for us, in comparison to some of the relevant literiture on the topic, such as the work from which this thesis is inspired from. Therefore, it would be interesting to see how our model would perform once trained on extensively more data.

Having access to better quality data would also allows us to try some other architecture. Namely, in exploring possible architectures for this work, a Recurrent Neural Network (RNN) was long considered. In particular, a Long Short Term Memory (LSTM) network, which has techniques in place to handle vanishing gradient problems related to RNN applications to time series forecasting. The LSTM is built in a way to be able to recognize long-term dependencies of the data. When used on a single day of data, the LSTM outperformed the standard networks. However, when used on several days, this wasn't the case anymore. This may have been due to the fact that we did not have access to contiguous days, but on singular days scattered around 2019 and 2018. So, there were few long-term dependencies to be found in data, and the LSTM was abandoned for the sake of this work.

It would be interesting to see how a LSTM-based network would perform given extensive and high-quality data. Moreover, the first trials with this latter architecture showed some degree of ability to transfer-learning. The model, first trained on a single stock, would be able to be trained on another stock and perform significantly better on the latter stock, without first training on the former stock.

Any transfer-learning application in NASDAQ LOB data could be incredibly consequential, given the format that NASDAQ compressed data comes into. As mentioned in Chapter 5, a single .ITCH50 files contains data from all the stock listed on the NASDAQ exchange. Therefore, whereas there could be little 'vertical' training data -i.e. in time-, there is enormous potential for 'horizontal' -i.e. across different stocks- training.

Appendix A

Data specifications

In this appendix, we will briefly list the stocks and the exact days[1] used in this thesis. As mentioned, we use two different sets of stocks depending on the nature of each analysis. If the analysis involves using only the best bid and best ask price, we use the stocks listed in Set A. If the whole book is needed, we use the stocks listed in Set B.

- **Set A**: it's composed by the stocks of the 43 companies with largest market capitalization listed at the NASDAQ, at the time of this work.

  The complete list of the stocks is:

  AAPL (Apple), ABBV (AbbVie), ABT (Abbott Laboratories), ADBE (Adobe), AMZN (Amazon), ASML (ASML Holding), BABA (Alibaba), BAC (Bank of America), CMCSA (Comcast), CRM(salesforce.com), CSCO (Cisco), CVX (Chevron), DIS (Disney), FB (Facebook), GOOGL (Google), HD (Home Depot), INTC (Intel), JNJ (Johnson&Johnson), JPM (JP Morgan), KO (Coke), LLY (Eli Lilly), MA (Mastercard), MSFT (Microsoft), NFLX (Netflix), NKE (Nike), NVDA (Nvidia), NVS (Novartis), ORCL (Oracle), PEP (Pepsi), PFE (Pfizer), PG (Procter&Gamble), PYPL (PayPal), T (AT&T), TM (Toyota), TMO (Thermo Fisher Scientific), TSLA (Tesla), TSM (Taiwan Semiconductor), UNH (United Health), V (Visa), VZ (Verizon), WMT (Walmart), XOM (Exxon Mobil).

[1] Stock data is stored into files each worth a day of data.

- **Set B**: it is composed by a subset of Set A. In Table A.1 below, we list the ticker of stocks along with the number of datapoints used for training.

MSFT	151960
AMZN	144450
FB	152204
NVDA	141444
AAPL	160224
TSLA	140062
GOOGL	136852
DIS	121906
NFLX	132620

Table A.1: Set B stocks and correspondent number of datapoints

- **Available days**: the list of days-worth of data used in this work is the following:

  21/01/2012, 28/12/2018, 30/01/2019, 27/03/2019, 30/05/2019, 30/07/2019, 30/08/2019, 30/10/2019

Bibliography

Frédéric Abergel and Aymen Jedidi. A mathematical approach to order book modeling. *International Journal of Theoretical and Applied Finance*, 16 (05):1350025, 2013.

Michael J Aitken, Henk Berkman, and Derek Mak. The use of undisclosed limit orders on the australian stock exchange. *Journal of Banking & Finance*, 25(8):1589–1603, 2001.

Marco Avellaneda and Sasha Stoikov. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224, 2008.

Marco Avellaneda, Josh Reed, and Sasha Stoikov. Forecasting prices from level-i quotes in the presence of hidden liquidity. *Algorithmic Finance*, 1 (1):35–43, 2011.

Aurelio Fernández Bariviera. The influence of liquidity on informational efficiency: The case of the thai stock market. *Physica A: Statistical Mechanics and its Applications*, 390(23-24):4426–4432, 2011.

Martino Bernasconi-De-Luca, Luigi Fusco, and Ozrenka Dragić. martinobdl/itch: Itch50converter, 2021. URL https://zenodo.org/record/5209267.

Hendrik Bessembinder, Marios Panayides, and Kumar Venkataraman. Hidden liquidity: an analysis of order exposure strategies in electronic stock markets. *Journal of Financial Economics*, 94(3):361–383, 2009.

Jose Blanchet and Xinyun Chen. Continuous-time modeling of bid-ask spread and price dynamics in limit order books. *arXiv preprint arXiv:1310.1103*, 2013.

Jean-Philippe Bouchaud, Marc Mézard, and Marc Potters. Statistical properties of stock order books: empirical results and models. *Quantitative finance*, 2(4):251, 2002.

Charles Cao, Oliver Hansch, and Xiaoxin Wang. The information content of an open limit-order book. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 29(1):16–41, 2009.

Rene Carmona and Kevin Webster. High frequency market making. *arXiv preprint arXiv:1210.5781*, 2012.

Alvaro Cartea, Ryan Donnelly, and Sebastian Jaimungal. Enhancing trading strategies with order book signals. *Applied Mathematical Finance*, 25(1): 1–35, 2018.

Rama Cont. Statistical modeling of high-frequency financial data. *IEEE Signal Processing Magazine*, 28(5):16–25, 2011. doi: 10.1109/MSP.2011.941548.

Rama Cont and Adrien De Larrard. Order book dynamics in liquid markets: limit theorems and diffusion approximations. *Available at SSRN 1757861*, 2012.

Rama Cont and Adrien De Larrard. Price dynamics in a markovian limit order market. *SIAM Journal on Financial Mathematics*, 4(1):1–25, 2013.

Rama Cont, Sasha Stoikov, and Rishi Talreja. A stochastic model for order book dynamics. *Operations research*, 58(3):549–563, 2010.

Rama Cont, Arseniy Kukanov, and Sasha Stoikov. The price impact of order book events. *Journal of financial econometrics*, 12(1):47–88, 2014.

Jonathan Donier, Julius Bonart, Iacopo Mastromatteo, and J-P Bouchaud. A fully consistent, minimal model for non-linear market impact. *Quantitative finance*, 15(7):1109–1121, 2015.

Parameswaran Gopikrishnan, Vasiliki Plerou, Xavier Gabaix, and H Eugene Stanley. Statistical properties of share volume traded in financial markets. *Physical review e*, 62(4):R4493, 2000.

Martin D Gould and Julius Bonart. Queue imbalance as a one-tick-ahead price predictor in a limit order book. *Market Microstructure and Liquidity*, 2(02):1650006, 2016.

Heike Hofmann, Karen Kafadar, and Hadley Wickham. Letter-value plots: Boxplots for large data. Technical report, had.co.nz, 2011.

Alec N Kercheval and Yuan Zhang. Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, 15 (8):1315–1329, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680, 2006.

Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Temporal bag-of-features learning for predicting mid price movements using high frequency limit order book data. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(6):774–785, 2018.

Faisal I Qureshi. Investigating limit order book characteristics for short term price prediction: a machine learning approach. *arXiv preprint arXiv:1901.10534*, 2018.

Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459, 2019.

Justin A Sirignano. Deep learning for limit order books. *Quantitative Finance*, 19(4):549–570, 2019.

Laura A Tuttle. Hidden orders, trading costs and information. *Trading Costs and Information (November 29, 2003)*, 2003.

Tzu-Wei Yang and Lingjiong Zhu. A reduced-form model for level-1 limit order books. *Market Microstructure and Liquidity*, 2(02):1650008, 2016.

Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, 2019.

Ruoqing Zhu, Donglin Zeng, and Michael R Kosorok. Reinforcement learning trees. *Journal of the American Statistical Association*, 110(512):1770–1784, 2015.