Tampere University

ADAMANTIOS NTAKARIS

Mid-Price Movement Prediction in Limit Order Books Using Feature Engineering and Machine Learning

Tampere University Dissertations 146

Tampere University Dissertations 146

ADAMANTIOS NTAKARIS

Mid-Price Movement Prediction in Limit Order Books Using Feature Engineering and Machine Learning

ACADEMIC DISSERTATION
To be presented, with the permission of
the Faculty of Information Technology and Communication Sciences
of Tampere University,
for public discussion in the Lecture room TB109
of the Tietotalo building, Korkeakoulunkatu 1, Tampere,
on Friday, 25 October 2019, at 12 o'clock.

ACADEMIC DISSERTATION
Tampere University,
Faculty of Information Technology and Communication Sciences
Finland

Responsible supervisor and Custos
Professor
Moncef Gabbouj
Tampere University
Finland

Supervisors
Professor
Juho Kanniainen
Tampere University
Finland

Adjunct Professor
Alexandros Iosifidis
Tampere University
Finland

Associate Professor
Aarhus University
Denmark

Pre-examiners
Professor
Alec N. Kercheval
Florida State University
USA

Associate Professor
Pekka Malo
Aalto University
Finland

Opponent
Professor
Peter Sarlin
Hanken School of Economics
Finland

The originality of this thesis has been checked using the Turnitin Originality Check service.

Cover design: Roihu Inc.

ISBN 978-952-03-1287-9 (print)
ISBN 978-952-03-1288-6 (pdf)
ISSN 2489-9860 (print)
ISSN 2490-0028 (pdf)
http://urn.fi/URN:ISBN:978-952-03-1288-6

PunaMusta Oy – Yliopistopaino
Tampere 2019

This dissertation is dedicated to my mother

ABSTRACT

The increasing complexity of financial trading in recent years revealed the need for methods that can capture its underlying dynamics. An efficient way to organize this chaotic system is by contracting limit order book ordering mechanisms that operate under price and time filters. Limit order book can be analyzed using linear and non-linear models.

The thesis develops novel methods for the identification of limit order book characteristics which provide traders and market makers an information edge in their trading. A good proxy for traders and market makers is the prediction of mid-price movement, which is the main target of this thesis. The contributions of this thesis are categorized chronologically into three parts. The first part refers to the introduction in the literature of the first publicly available limit order book dataset for high-frequency trading for the task of mid-price movement prediction. This dataset comes together with the development of an experimental protocol that utilizes methods inspired by ridge regression and a single layer feed-forward neural network as classifiers. These classifiers use state-of-the-art limit order book features as inputs for the target task.

The next contribution of this thesis is the use and development of a wide range of technical and quantitative indicators for the task of mid-price movement prediction via an extensive feature selection process. This feature selection process identifies which features improve predictability performance. The results suggest that the newly introduced quantitative feature based on an adaptive logistic regression model for online learning was selected first according to several criteria. These criteria operate according to entropy, linear discriminant analysis, and least mean square error.

The third contribution is the introduction of econometric features as inputs to deep learning models for the task of mid-price movement prediction. An extensive comparison against other state-of-the-art hand-crafted features and fully automated

feature extraction processes is provided. Furthermore, a new experimental protocol is developed for the task of mid-price prediction, to overcome the problem of time irregularities, which characterizes high-frequency data. Results suggest that advanced hand-crafted features such as econometric indicators can predict movements of proxies, such as mid-price.

PREFACE

The work presented in this thesis has been carried out at Tampere University, Finland, during the years 2016-2019.

First and foremost, I would like to thank my supervisor, Professor Moncef Gabbouj, for his endless support, patience, and help throughout this project. His academic genius and rigorous teaching have not only been invaluable but also inspirational.

I would also like to thank my second supervisor Professor Juho Kanniainen, for his constant support as well as his technical supervision for the numerous researches that we have conducted.

I would also like to acknowledge a special debt to my advisor and immensely talented Professor Alexandros Iosifidis, who helped me to run the extra mile during this project.

I also gratefully acknowledge the funding that I received from Marie Sklodowska-Curie Innovative Training Network for the entire duration of my Ph.D. studies.

To my co-authors, Giorgio Mirone and Martin Magris: thank you for your generous collaboration on our shared projects. I would also like to thank BigDataFinance ESRs, MUVIS group researchers, Milla Mäkinen, Jaakko Valli, Kestutis Baltakys, Milla Siikanen, Jenni Raitoharju, Nikolaos Passalis, Vivi Nousi, Avraam Tsantekidis, and George Kanellis for all the discussions we had during these three years of my doctoral education.

I also want to express my profound appreciation to my colleague, Morteza Zabihi, for his help.

My family deserves all the credits as the steady base that has given me the spiritual and material means to accomplish my goals. I feel deeply grateful to my parents,

Maroula and Yannis, as well as my grandparents, Despina and Kostas. My sister and my best friend, Dr. Virginia Dakari, has always been there for me, believing in me and cheering me up at times when I felt my courage failing. As for my Alina, for her enduring love, affection, and sunny optimism, I thank her deeply.

22/05/19 Adamantios Ntakaris

CONTENTS

List of Symbols

A	Reaction-Diffusion Particle
α	Limit Orders Arrival Rate
$\mathcal{A}(t)$	Set Of Sell Orders
$\mathcal{B}(t)$	Set Of Buy Orders
B	Reaction-Diffusion Particle
C_i^t	LSTM's Cell Gate
$e^{(i)}$	Error Function at time instance i
$E[\cdot]$	Expected Value
f_i^t	LSTM's Forget Gate
$f_T(t)$	Poisson Process at time t
$\hat{f}_h(\cdot)$	Kernel Density Function
g_i^t	LSTM's Input Gate
$\mathbf{h}$	Activation's Function Vectors
h	Bandwidth
$\mathbf{H}^\dagger$	Moore-Penrose Pseudoinverse Matrix
$\mathbf{H}$	Network's Hidden Layer Outputs Matrix
$\mathbf{h}_t$	LSTM's Hidden State Output
k	Time Series Lag
l_x	Maximal Output Value
$n^b(\cdot)$	Bid-Side Depth
$n^a(\cdot)$	Ask-Side Depth
o_i^t	LSTM's Output Gate
$\mathbf{o}$	Neural Network Output
p_{x-}^a	Before Change Ask Price
p_x	New Order's Price
p_{x-}^b	Before Change Bid Price
p_x^a	New Ask Price
p_x^b	New Bid Price
p_j^a	Ask Agent's LOB Price
p_j^b	Bid Agent's LOB Price
$\bar{p}$	Upper Bounded Price Particle
$r_{X,k}$	Autocorrelation Function

R_i	Stationarity Process
$\tilde{R}_i$	Stationarity Process After Price Change
S	Random Variable
t_x	New Order's Arrival Time
$\mathbf{T}$	Targets $\mathbf{t}_i$ Matrix
$\mathbf{t}_i$	Target Vector At Time Instance
v^a_{x-}	Before Change Ask Volume
v^b_{x-}	Before Change Bid Volume
V^b_{x-}	Before Change Bid Volume
v_t	New Order's Volume
V^a_{x-}	Before Change Ask Volume
V^a_x	New Ask Volume Order
V^b_x	New Bid Volume Order
$\mathbf{V}$	Hidden's Layer Weights
$\mathbf{W}$	Weights' Matrix
X	Time Series
$\mathbf{X}$	Samples $\mathbf{x}_i$ Matrix
x_i	Sample At Time Instance i
x	New Orders
$\mathbf{x}_i$	Set Of Vectors
$Z(t)$	Statistical Criterion
z	Cut-off Point
$\Delta p(t)$	Price Variation At Time t
η	Market Orders Arrival Rate
λ	Regularization Penalty
μ	Mean
π	Tick Size
σ	Smallest Traded Amount
σ_g	Sigmoid Function
σ^2_z	Constant Variance
ω	Poisson's Counting Process Rate

List of Abbreviations

ADX	Average Directional Index
AE	Autoencoder
CNN	Convolutional Neural Network
DM	Directional Movement
EG	Engle-Granger Statistical Test
ELM	Extreme Learning Machine
FN	False Negatives
FP	False Positives
FR	Feature Representation
HF	High Frequency
i.i.d.	Independent and Identical Distributed
IV	Integrated Variance
LDA	Linear Discriminant Analysis
LMS	Least Mean Squares
LOB	Limit Order Book
LRL	Linear Regression Line
LSTM	Long Short-Term Memory
MB	Message Book
MLP	Multilayer Perceptron
PA	Pre-averaged Variance
QV	Quadratic Variation
RBFN	Radial Basis Function Neural Network
ReLU	Rectified Linear Activation Function
RK	Realized Kernel
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RV	Realized Variance
RR	Ridge Regression
SAR	Stop And Reverse
SLFN	Single-Hidden Layer Feedforward Neural Network
TF	True Negatives
TP	True Positives
TR	True Range

LIST OF PUBLICATIONS

[P 1] A. Ntakaris, M. Magris, J. Kanniainen, M. Gabbouj and A. Iosifidis. Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting* 37.8 (2018), 852–866.

[P 2] A. Ntakaris, J. Kanniainen, M. Gabbouj and A. Iosifidis. Mid-Price Prediction Based on Machine Learning Methods with Technical and Quantitative Indicators. *PLOS ONE* (under review).

[P 3] A. Ntakaris, G. Mirone, J. Kanniainen, M. Gabbouj and A. Iosifidis. Feature Engineering for Mid-Price Prediction With Deep Learning. *IEEE Access* 7 (2019), 82390–82412.

Preliminary results of [P1] were presented at the 10^{th} *International Conference on Computational and Financial Econometrics (CFE 2016)* in Seville, Spain, on December 11^{th} 2016.

1 INTRODUCTION

Trading started several centuries ago. The Dutch East India was the first publicly traded company in 1602, while the New York Stock Exchange (NYSE) was created in 1817 and became the most dominant stock ex-change in the USA. Nasdaq, the second-biggest stock exchange after NYSE, was founded in 1971 and acquired other stock exchanges such as the Finnish stock exchange in Helsinki[1] in 2003. However, it was not until 1967 that electronic trading platforms were introduced as an alternative trading system by Instinet in New York. Since then, electronic and automated trading increased their impact on the trading floor and nowadays, this type of trading is the only possible way to exchange stocks. Electronic platforms created a very complex and remarkably fast system which offers instant access to several stock exchanges. This dynamic system is able to trade large volumes in milliseconds or even nanoseconds. Transaction speed and complexity of this trading universe created the need for efficient analysis of its dynamics.

One way to formulate and organize such a complex system is the so-called limit order book (LOB). LOB is the way that stock exchanges organize their trading activity. It is a mechanism that sorts trades on a price-time sequence, which means that LOB prioritizes orders first according to the price level and next according to arrival time. LOB has two sides: the ask side and the bid side. Ask refers to the minimum price someone is willing to sell a given stock at, while bid refers to the maximum price that someone is willing to pay for a stock. Every LOB's ask and bid side is divided into several price levels. Every price level comes together with the volume/number of available stock pieces. Both LOB sides (i.e., ask and bid) have their best ask and bid prices, respectively. The average of the best ask and bid price constitutes the so-called mid-price. Even though mid-price cannot be traded directly, many investors use it as a proxy to identify potential movements of the actual ask and bid stock prices.

[1] Helsinki Stock Exchange started trading since October 1912

Movement prediction of proxies like LOB's mid-price is a challenging task due to several reasons such as data complexity and availability, identification of factors that explain the source of uncertainty, and assumption elimination of every possible underline process. Another challenge that makes LOB's mid-price movement prediction even more complicated is the size of intraday trading activity in a high-frequency environment. Under these challenges, a critical question arises: is LOB's mid-price predictable? The present thesis proposes a coherent answer to this question by connecting machine and deep learning methods with state-of-the-art feature engineering as part of several experimental protocols.

Machine and deep learning methods are capable of analyzing linear and non-linear complex systems. An example of a complex system is the LOB. Such methods were extensively used in image and video applications for data parsing (e.g., filtering) via numerous model options, mainly based on functions-kernels used for classification or prediction. Machine and deep learning aim to learn from the data (e.g., LOB dataset), and, together with recent computational advancements, they offer efficient analysis methods for financial applications, such as mid-price movement prediction. Besides their ability to learn from data, these methods are competent enough to utilize representations effectively as inputs. These inputs are usually handcrafted features, and their purpose is to transform raw data into meaningful signals. That being said, development of state-of-the-art handcrafted features comes with challenges as they require an in-depth analysis of data and understanding of any underlying process that may be attached to it.

In this thesis, such in-depth analysis is proposed as a result of the extensive experimental basis for the task of LOB's mid-price movement prediction. First, an experimental protocol for high-frequency LOB data is developed and made publicly available. What is more, state-of-the-art LOB hand-crafted features and linear and non-linear classifiers utilized and developed for LOB dynamics analysis. The solutions have proven that mid-price's movements can be predictable. Next, feature engineering exploitation based on technical and quantitative indicators as extensions of the existing LOB features provides insight via an extensive study, whose features are suitable for the task of mid-price movement prediction. This study is based on the conversion of entropy, linear discriminant analysis, and least mean square as feature sorting methods. This analysis reveals that mid-price movements could be predicted by means of the best subset selection of advanced hand-crafted features. Last, one

more hand-crafted econometric-based feature set is presented as input to a wide range of deep learning models. Its aim is to offer a fair evaluation against LOB, technical, quantitative and fully automated features. The output of this analysis led to highly accurate estimation of when the next mid-price movement will happen.

1.1 Objectives and Outline of the Thesis

In this thesis, the focus is placed on capturing, via feature engineering, LOB dynamics by utilizing machine and deep learning methods. The objectives of this thesis addressing the research problem stated above are the following:

- To investigate whether state-of-the-art features can predict, with the use of machine learning models, mid-price movements based on high-frequency LOB data.
- To develop state-of-the-art handcrafted features and scrutinize their effectiveness for the task of mid-price movement prediction through the development of experimental protocols which they can be publicly available.
- To determine the effectiveness of deep learning for online mid-price movement prediction via feature engineering.

The thesis is organized as follows. In Chapter 2 an overview of the key concepts of interest are provided. This background section contains the main ideas of High-frequency (HF) Data properties (Section 2.1), Limit Order Books (Section 2.2), Machine and Deep Learning (Section 2.3), and Feature Engineering (Section 2.4). Chapter 3 presents the datasets, experimental protocols and metrics used in the analysis. The contributions of this thesis are discussed in Chapter 4. Section 4.1 presents a novel approach to predicting mid-price movements based on state-of-the-art handcrafted features. In Section 4.2 an extensive analysis for feature engineering is developed together with a novel set of handcrafted features for online learning for the task of mid-price movement prediction. In Section 4.3 the development of econometric features for deep learning is introduced via an experimental protocol suitable for online learning for the task of mid-price movement prediction. Finally, Chapter 5 summarizes the most important findings of the thesis along with concluding remarks and topics for future research.

1.2 Publications and Author's Contribution

In [P1], a benchmark LOB dataset was published for the task of mid-price movement prediction. The major contribution of this paper is the very first detailed experimental protocol for a LOB benchmark dataset based on Nordic stocks. State-of-the-art handcrafted features were tested through linear and non-linear classifiers for the prediction of several projected horizons of future mid-price movements. Baseline models set up on ridge regression and a single-hidden layer feedforward neural network (SLFN) with k-means were suggested in this task. The candidate is the main author of the publication and is responsible for developing and implementing the suggested models and protocol as well as writing the initial draft of the paper.

In [P2], an extensive feature selection evaluation was performed for the task of mid-price prediction. The evaluation was based on the majority of technical indicators together with quantitative indicators and LOB features. Under a wrapper method, the best sets of features were suggested as optimal selection for the classification task of mid-price movement prediction. Three methods, entropy, linear discriminant analysis (LDA), and least mean squares (LMS), converted into feature selection criteria where the majority of them selected first the newly introduced adaptive logistic regression feature. Testing the majority of basic and advanced technical indicators, the development of an adaptive logistic feature and the conversions of entropy, LDA, and LMS as feature selection criteria constitue the main contribution of this publication. The candidate developed and implemented the suggested models and protocol and wrote the initial draft of the paper.

In [P3], econometrics features were suggested as inputs to several deep learning models, like multilayer perceptrons (MLP), convolutional neural networks (CNN), and long short-term memory (LSTM) neural networks, for the task of mid-price movement prediction. Introduction of econometrics features is one the main contributions of this publication since they performed, in some cases, much better than technical, quantitative, LOB and fully automated features. Another contribution of this publication is the development of a new experimental protocol to overcome the problem of time irregularities of HF data. The candidate is the first author of this publication and is responsible for developing and implementing the suggested models and protocol. He also wrote the first draft of the paper.

2 BACKGROUND WORK

This thesis introduces feature engineering for LOB high-frequency trading data by utilizing machine and deep learning for the task of mid-price movement prediction. We will first introduce the main concepts of High-Frequency Data, Limit Order Book, Feature Engineering, and Machine and Deep Learning.

2.1 High Frequency Data

Algorithmic trading is a computer-based trading mechanism of selling and buying stocks depending on computerized instructions. High-frequency trading is a type of algorithmic trading that operates under low-latency setting. This low-latency setting refers to a system that operates under a single digit millisecond round-trip time for a packet of data/orders. These data packets, named high-frequency data, provide records of the intraday trading activity (i.e., message book (MB)). These records contain information regarding:

- Security's ID code
- Timestamp
- Price
- Volume
- Side (Ask or Bid)
- Type of Order (Submission, Cancellation, Execution)
- Full or Partial Cancellation or Execution

These details are an example (e.g., Table 2.1) of the available trading information and can be extended according to the level of the subscription fee. The flow of intraday information creates the need for analysis of its dynamics.

Table 2.1 Amazon's message list example from 11:27:45:289 to 11:27:45:305 on 22.09.15, where time is expressed in UNIX format and price is multiplied by 10,000

Timestamp	Id	Price	Quantity	Event	Side
1442921265289000	119132763	5345100	100	Cancellation	Bid
1442921265289000	119137068	5345200	100	Submission	Bid
1442921265297000	119137122	5342100	40	Submission	Ask
1442921265299000	119137068	5345200	100	Cancellation	Bid
1442921265299000	119137132	5345100	100	Submission	Bid
1442921265305000	119117680	5340100	32	Cancellation	Ask

2.1.1 High-Frequency Data Properties

High-frequency data creates the need for analysis due to its complex nature and dynamics. The analysis can be based on the following general properties of high-frequency data:

- **Negative Autocorrelation**
  Asset's returns[1] in a high frequency environment exhibit negative first-order autocorrelation. Autocorrelation measures the cor- relation between a time series with a lagged version of itself. More specifically, for an N-sample time series $X = \{x_1, x_2, ..., x_N\} \in \mathbb{R}^N$ of N samples the autocorrelation with lag k at time instance i is defined as:

$$r_{X,k} = \frac{E[(x_i - \mu)(x_{i+k} - \mu)]}{\sqrt{E[(x_i - \mu)^2]E[(x_{i+k} - \mu)^2]}} \tag{2.1}$$

  where $E[(x_i - \mu)(x_{i+k} - \mu)]$ is the covariance between the time series and its lagged version, and $E[(x_i - \mu)^2]$ and $E[(x_{i+k} - \mu)^2]$ are the corresponding variances.

  Many authors (e.g. [3], [15], [28]) found evidence of the existence of negative autocorrelation in high-frequency data. An example of the presence of negative correlation can be seen in Figure 2.1 which is based on Amazon trading on 9/22/2015. One can clearly see the presence of negative first-order (i.e., lag 1) autocerralation which gradually disappears as the lag increases.

[1] Asset's return is the percentage change of the stock price based on stock's initial value

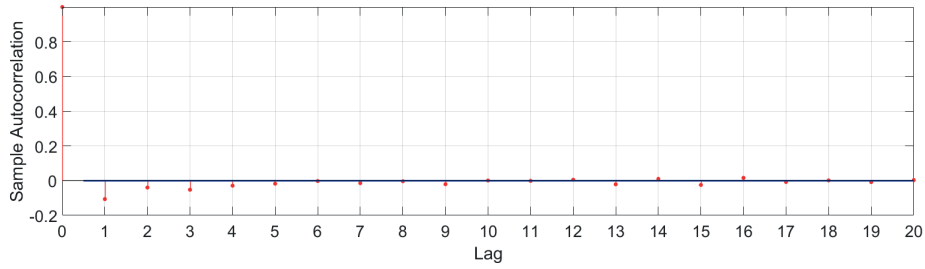


Figure 2.1 Correlogram up to 20 lags based on Amazon returns for the trading session on 9/22/15. Correlogram shows the existence of negative first-order autocorrelation with tight confidence intervals (blue lines).

- **Absence of Normality or Lognormality with Heavy Tails**
  Stock prices in financial markets follow a lognormal distribution (e.g., [49]) under the condition that sequential price changes are normally distributed. This does not hold for the case of high-frequency data where the differencing interval[2] becomes shorter (e.g., [3], [15], [24]) and data loses its normality in the case of a heavy tailed distribution. Figure 2.2 shows that returns' histograms are not normally distributed and a non-parametric statistical distribution describes better the data (i.e., stock returns). The non-parametric statistical distribution is based on the following kernel density estimator:

$$\hat{f}_h(x) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) \tag{2.2}$$

  where h is the bandwidth adjusted to the data (i.e., Amazon returns) and $K(\cdot)$ is the Gaussian kernel smoothing function $K(x, x_i) = e^{-\frac{(x-x_i)^2}{2h^2}}$. Heavy tails can be seen in Figure 2.3, where a Q-Q[3] provides evidence regarding the existence of heavy tails in both extremes.

- **Bid-Ask Bounce**
  Bid and ask data arrive asynchronously, which means that the quoting process is exposed to noise. The quoting process in tick-by-tick data that creates the bid-ask spread, which is the difference between the bid quote and the ask quote

[2]Differencing interval is defined as the space between two consecutive data points.

[3]Q-Q (i.e., Quantile-Quantile) plot compares the quantiles of a given dataset and a set of quantiles derived from a probability distribution.

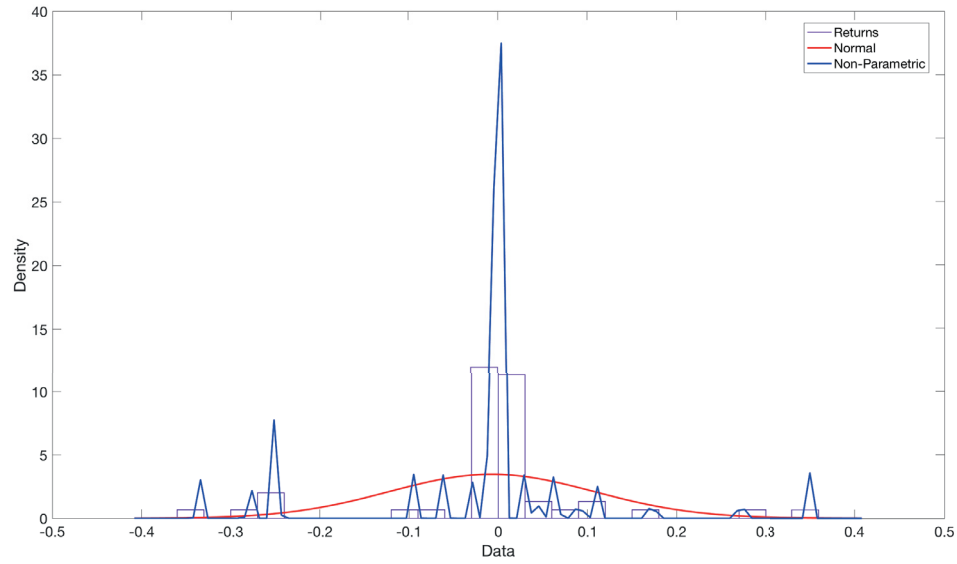


Figure 2.2 High-frequency data-based Amazon returns can be better described by a non-parametric distribution compared to the normal distribution.

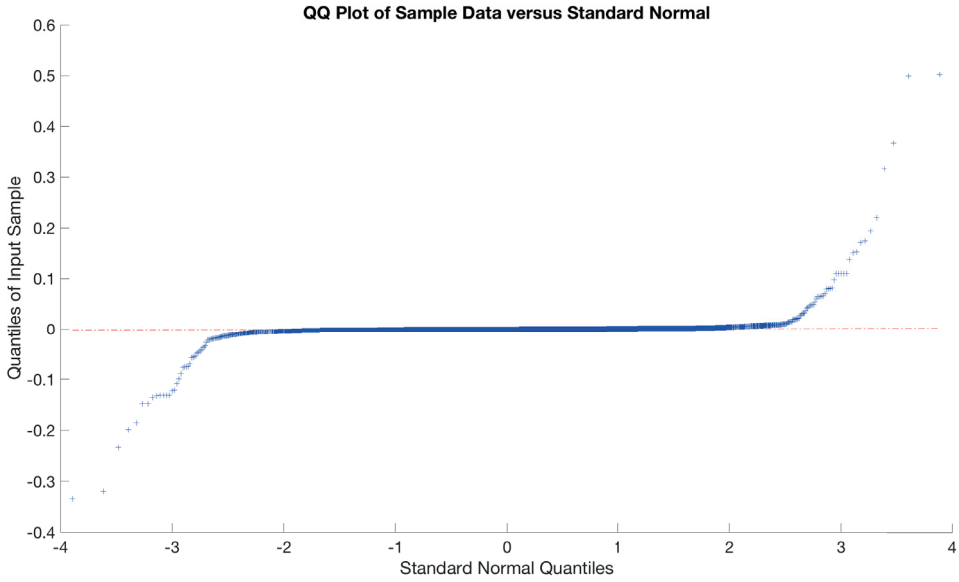


Figure 2.3 Q-Q plot of Amazon returns against normal distribution where presence of heavy tails is obvious in both extremes.

at any time. Bid-ask spread plays a critical role in stock price forecasting since it exhibits a different behavior in liquid and illiquid stocks (i.e., liquid stocks

have a smaller bid-ask spread compared to illiquid stocks where the spread is higher). This spread represents the profit and some commission fees market maker firms make. This spread creates room for the bid-ask bounce effect. Bid-ask bounce (e.g., Figure 2.4) is the bouncing of any trade between the bid and ask price. This oscillation can lead to bias in high-frequency data analysis. For instance, authors in [33] suggest that the price movement that takes place inside the bid-ask spread is not an actual price movement but leads to idiosyncratic volatility increase.[4]

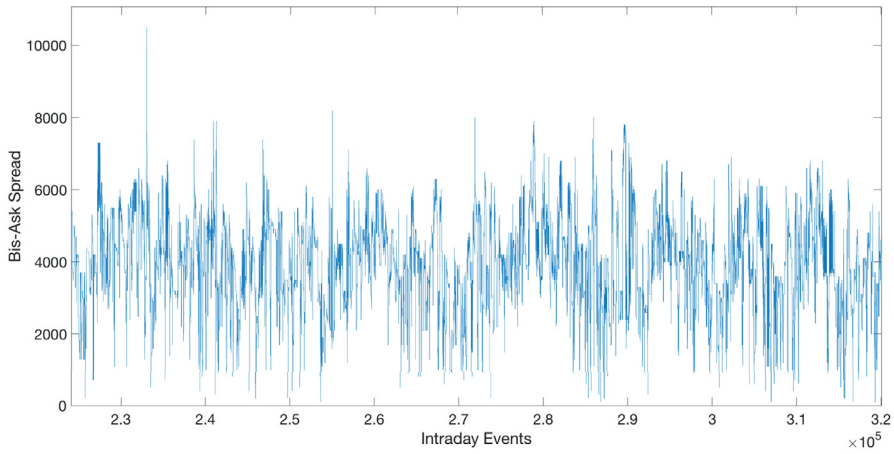


Figure 2.4 Amazon's intraday (9/22/15) bid-ask spread fluctuations

2.2 Limit Order Book

Modern financial markets operate under a double auction mechanism called limit order book (LOB). LOB accepts two types of orders, limit and market orders. A limit order is an order to sell or buy a stock at a specified price, whereas a market order is traded instantly under the current market price. As a result, LOB acts as a recording mechanism for the unexecuted trading activity where time and price priority are the primary filters for queues creation. These queues represent, at any time t , the price levels of the bid and ask sides and vary according to order arrivals

[4] Volatility is a risk measure and is based on the standard deviation of asset returns.

and liquidity [5] levels.

Orders, following [30], are defined as $x = (p_i, v_i, t_t)$ where p_i, v_i, and t_i represent the price (i.e., bid or ask), the volume and the order submission time at time i, respectively. All orders arrive with size $v_i \in \{\pm k\sigma | k = 1, 2, ...\}$ with σ the smallest traded amount[6]. The set of all active orders, $\mathcal{L}(t)$, is a càdlàg process since for every new limit order x, the following holds: $x \in \mathcal{L}(t_i)$ and $x \notin \lim_{t' \uparrow t_x} \mathcal{L}(t')$. This activity defines the depth size for every LOB level in both bid and ask sides. The depth size, especially in the best bid and ask level, is a key element for the price formation (see e.g., [9], [18]). More specifically, the available bid-side depth $n^b(p, i)$ (equivalently, the ask-side depth $n^a(p, t)$) at time i is defined as:

$$n^b(p, i) := \sum_{\{x \in \mathcal{B}(i) | p_i = p\}} v_i \tag{2.3}$$

where $\mathcal{B}(i)$ is the set of all active buy orders (equivalently $\mathcal{A}(i)$ is the set of all sell orders).

The depth of each level in both (bid and ask) sides is modified constantly due to limit orders, market orders, and cancellations. Limit orders increase the size depth while market orders and cancellations remove liquidity from LOB. Instead of following this trading activity, the trader can handle the bid and ask queues, as presented in [14], as stationary processes namely $(R_i)_{i \geq 1}$ after a price increase and $(\tilde{R}_i)_{i \geq 1}$ when there is a price decrease for the i^{th} event. Both stationary processes exhibit the same distribution for the order book depth in case of a price increase or decrease. For instance, in the case of reduced-form model, for a new order or cancellation arrival, in the bid side, of size V_i^b at time instance t_i^b there are two scenarios:

- if $v_{i-}^b + V_i^b \geq 0$ then there is no price change and the order will be satisfied
- if $v_{i-}^b + V_i^b < 0$ then the size of the bid level is reduced together with the price by one 'tick' of size π. Based on the updated $\tilde{R}_i = (\tilde{R}_i^b, \tilde{R}_i^a)$ values for the bid

[5] Liquidity is defined according to three factors: (i) tightness of a liquid market (i.e., the position's alternation cost over a short period of time), (ii) depth (i.e., order-flow innovations for price changes), and (iii) resiliency (i.e. recovering time from random, uninformative shocks) [39].

[6] σ together with π, which is the smallest price interval between orders, are the so-called LOB's resolution parameters.

and ask side, the new LOB state is:

$$(p_i^b, v_i^b, v_i^a) = (p_{i-}^b, v_{i-}^b + V_i^b, v_{i-}^a)\mathbb{1}_{\{v_{i-}^b \geq -V_i^b\}} + (p_{i-}^b - \pi, \tilde{R}_i^b, \tilde{R}_i^a)\mathbb{1}_{\{v_{i-}^b < -V_i^b\}} \tag{2.4}$$

where $\mathbb{1}$ is the indicator function, p_{i-}^b is the best bid price before the update, v_{i-}^b and v_{i-}^a are the volume sizes for the best bid and ask sides, respectively.

In a similar fashion, for a new arrival to the ask side, of size V_i^a, LOB's state will be:

- if $v_{i-}^a + V_i^a \geq 0$ then there is no price change and the order will be satisfied
- if $v_{i-}^a + V_i^a < 0$ then the size of the ask level is reduced together and the price will be increased by one 'tick' of size π. Based on the updated $\tilde{R}_i = (\tilde{R}_i^b, \tilde{R}_i^a)$ values for the bid and ask side, the new LOB state is:

$$(p_i^a, v_i^b, v_i^a) = (p_{i-}^b, v_{i-}^b, v_{i-}^a + V_i^a)\mathbb{1}_{\{v_{i-}^a \geq -V_i^a\}} + (p_{i-}^b + \pi, \tilde{R}_i^b, \tilde{R}_i^a)\mathbb{1}_{\{v_{i-}^a < -V_i^a\}} \tag{2.5}$$

where $\mathbb{1}$ is the indicator function, p_{i-}^b is the best bid price before the update, v_{i-}^b and v_{i-}^a are the volume sizes for the best bid and ask sides before the update, respectively.

2.2.1 Modelling Limit Order Book Dynamics

The constant LOB state updates create a dynamical system that has attracted the attention of many researchers and practitioners. Modeling of LOB dynamics is based on the assumptions that must be proven by the data. Several models have been suggested as potential solutions to the description of LOB dynamics.

Cut-Off Strategies

Driven mainly by statistics which are related to $\mathcal{L}(t)$ and order flow.[7] Cut-off strategies formulate a decision-making trading approach. The main idea behind this type of modeling is close to a statistical hypothesis test. For instance, following [30], a trader will place an order when the spread[8] is smaller than 5π. Usually, this type of LOB modeling exhibits the difference between informed and uninformed traders.

[7] Limit orders, together with market orders and cancellations, are the critical components of the so-called order flow.

[8] Spread is defined as the difference between the best bid and ask price.

Definition *A cut-off strategy, between decisions D_1 and D_2, is defined according to the comparison of a statistic $Z(t)$ and a cut-off point z:*

$$\begin{cases} D_1, & if\ Z(t) \leq z, \\ D_2, & otherwise. \end{cases} \tag{2.6}$$

Diffusion Models

Diffusion models belong to a modeling approach where the arrival of orders, cancellations, and executions are described by a stochastic process. More specifically, the first model of this type was introduced by [5], where the authors suggested that the interaction between order flow and LOB follows the reaction-diffusion model $A + B \rightarrow \emptyset$ in Physics. These two types of particles are inserted in a pipe of size $\tilde{p}$ and move randomly by a step of size one. Every time these two particles collide, they are annihilated, and as a result, two new particles are created. This process can describe orders' arrivals in the LOB. The authors make the following analogies: 1) Particle → Orders, 2) Finite Pipe → Order Book, and 3) Collision → Transaction.

They define the stock price at time t as $p(t) \in \{0, ..., \bar{p}\}$, where $\bar{p}$ is an upper bounded discrete price. Every simulation considers half of the agents in the bid side and the other half in the ask side asking for one share of the stock each:

$$p_j^b(0) \in \left\{0, \frac{\bar{p}}{2}\right\},\ j = 1, ..., \frac{N}{2} \tag{2.7}$$

and

$$p_j^a(0) \in \left\{\frac{\bar{p}}{2}, \bar{p}\right\},\ j = 1, ..., \frac{N}{2} \tag{2.8}$$

where N is the number of noise traders[9]. The trading activity revision at each time step t is based on an equal probability of a stock going up or down by one tick size. As a result, every new price for the agent is:

$$\begin{aligned} bid\ side &: p_j^b(t+1) = p_j^b \pm 1 \\ ask\ side &: p_j^a(t+1) = p_j^a \pm 1 \end{aligned} \tag{2.9}$$

[9]These are the traders whose current volatility depends on the recent changes in the market and they imitate others in buying and selling stocks.

The matching or transaction achieved if exists $(i,j) \in \{1,...,\frac{N}{2}\}$ such that:

$$p_i^b(t+1) = p_j^a(t+1). \tag{2.10}$$

This dynamical system follows, as authors suggest, the price variation $\Delta p(t)$ at time t as this presented by [6]:

$$\Delta p(t) \sim t^{\frac{1}{4}}\Big(ln\big(\frac{t}{t_0}\big)\Big)^{\frac{1}{2}}, \tag{2.11}$$

where t_0 represents the initial trading time. The idea that a reaction-diffusion approach can model LOB dynamics has some drawbacks. For instance, simulations suggest a Hurst exponent of $H = 1/4$ and no fat tails exist in the returns' distribution.

Continuous-time Models

Continuous-time models' main idea is that order arrivals (i.e., market orders, limit orders, and cancellations) are independent Poisson processes, as this can be seen in models like [16], [17], [19], [20], [53]. The basic model (i.e., [20]) considers the order flow and the relevant adjustments to $\mathcal{L}(t)$ as a queueing system where:

- market orders arrive in chunks of size V^m at a rate of ν per unit time,
- limit orders arrive in chunks of size V^l at a rate of α per unit time,
- offers are placed, with a uniform probability, on the price grid $\{1,...n\}$ with tick size π resolution.

The motivation for choosing independent Poisson processes is their simplicity and the fact that they can describe orders' arrival independently. Following [27], a Poisson process is an arrival process as defined below:

Definition *An arrival process is an increasing sequence of random variables* $0 < S_1 < S_2 < ...$, *with* $S_{i+1} - S_i$ *be always a positive random variable.*

Definition *A Poisson process is a renewal*[10] *process where the interarrival intervals follow an exponential distribution function with* $\omega > 0$ *and each of the interarrival times* T_i *has a density*

$$f_T(t) = \omega e^{-\omega t},\ t \geq 0. \tag{2.12}$$

[10] A renewal process is an arrival process during which the sequence of interarrival times is a sequence of i.i.d. random variables.

This model has several drawbacks since Poisson processes are not able to describe the different arrival rates (i.e., intensity periods of high arrival rates cluster in time) as suggested in [21], [31], and [61].

2.3 Machine and Deep Learning

The recent rise of computational power facilitated fields such as machine and deep learning to grow. Machine learning is an automated process where an algorithm performs a task based on a set of filters and conditions. There are several tasks a machine learning algorithm can solve via supervised or unsupervised learning. A supervised task occurs when the algorithm has access to the so-called ground truth, which is a 'proof' of an event occurrence[11]. The second type of tasks machine learning algorithms are suitable for are the ones without ground truth. These methods are based only on the given data and operate according to a self-feedback loop during the learning process. Both methods require tuning depending on the dataset and the problem under consideration. Many applications (e.g., image and voice recognition, time-series analysis, etc.) are based on big datasets, which demand analysis of their dynamics. The focal point of the present thesis is the use of machine learning methods for time-series analysis under the scope of classification and regression for LOB. The machine learning methods presented in this thesis vary from statistics and regression models to neural networks and deep learning.

2.3.1 Ridge Regression

Ridge regression [54] is based on a linear mapping, expressed by a matrix $\mathbf{W} \in \mathbb{R}^{D\times C}$, that optimally maps a set of vectors $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1,\ldots,N$ to another set of vectors (noted as target vectors) $\mathbf{t}_i \in \mathbb{R}^C$, $i = 1,\ldots,n$, by optimizing the following criterion:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{i=1}^{n} ||\mathbf{W}^T\mathbf{x}_i - \mathbf{t}_i||_2^2 + \lambda||\mathbf{W}||^2, \tag{2.13}$$

where the matrix notation is:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} ||\mathbf{W}^T\mathbf{X} - \mathbf{T}||^2 + \lambda||\mathbf{W}||^2. \tag{2.14}$$

[11] Ground truth is always subject to expert's opinion.

$\mathbf{X} = [\mathbf{x}_i, \ldots, \mathbf{x}_n]$ and $\mathbf{T} = [\mathbf{t}_i, \ldots, \mathbf{t}_n]$ are matrices formed by the samples $\mathbf{x}_i$ and $\mathbf{t}_i$ as columns, respectively, and λ is the regularization penalty. Every sample $\mathbf{x}_i$ corresponds to an event with dimensions of that of the input data (e.g., number of different features). For instance, in a three-class classification problem, the elements of vectors $\mathbf{t}_i \in \mathbb{R}^C$ ($C = 3$ in this case) take values $t_{ik} = 1$, if $\mathbf{x}_i$ belongs to class k, and, if $t_{ik} = -1$, otherwise. The solution of Eq. 2.14 is given by:

$$\mathbf{W} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{T}^T, \tag{2.15}$$

or

$$\mathbf{W} = \left(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}\right)^{-1}\mathbf{X}\mathbf{T}^T, \tag{2.16}$$

where $\mathbf{I}$ is the identity matrix of appropriate dimensions[12]. After the calculation of $\mathbf{W}$, a new (test) sample $\mathbf{x} \in \mathbb{R}^D$ is mapped to its corresponding representation in space $\mathbb{R}^C$, i.e. $\mathbf{o} = \mathbf{W}^T\mathbf{x}$, and is classified according to the maximal value of its projection, i.e.:

$$l_{\mathbf{x}} = \arg\max_k \; o_k. \tag{2.17}$$

2.3.2 Single-Hidden Layer Feedforward Neural Network

Single-Hidden Layer Feedforward Neural Network (SLFN) is a type of extreme learning machines (ELM) [35] and belongs to the feedforward type of artificial neural networks. Their topology is based on three layers - input, hidden, and output layers (see Fig. 2.5). SLFN exhibits a good generalization performance and fast learning speed, which makes it suitable for bid datasets.

There are several ways to train this type of neural network. The most common way is to use the proposed method in [35] where weights of the hidden layers are determined randomly, but in this thesis, these weights follow [36] where clustering applied on the training data was used. The clustering is based on K-means algorithm where K prototype vectors will be determined, and which are used as the network's hidden layer weights.

Since the network's hidden layer weights $\mathbf{V} \in \mathbb{R}^{D\times K}$ are ready, the input data $\mathbf{x}_i$, $i = 1, \ldots, N$ are non-linearly mapped to vectors $\mathbf{h}_i \in \mathbb{R}^K$, expressing the data

[12]When data is big, $\mathbf{W}$ should be computed using Eq. 2.16, since the calculation of Eq. 2.15 is computationally very expensive.

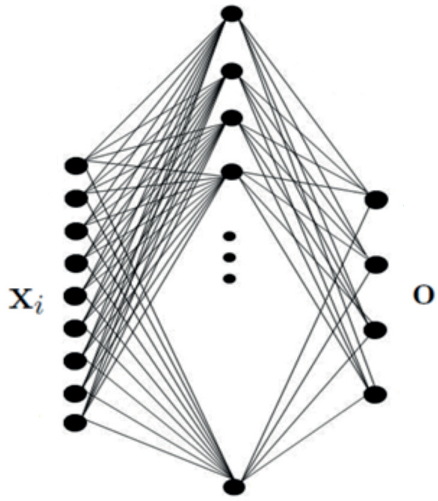


Figure 2.5 SLFN example with several input units (left), one hidden layer (middle) and an output layer with four units(right) [P1].

representations in the feature space determined by the network's hidden layer outputs $\mathbb{R}^K$. Then a radial basis function (RBFN) is used as the activation function, i.e. $\mathbf{h}_i = \phi_{RBF}(\mathbf{x}_i)$, calculated element-wise, as follows:

$$h_{ik} = \exp\left(\frac{||\mathbf{x}_i - \mathbf{v}_k||_2^2}{2\sigma^2}\right),\ k = 1,\ldots,K, \tag{2.18}$$

where σ is a hyper-parameter denoting the spread of the RBF neuron and $\mathbf{v}_k$ corresponds to the k-th column of $\mathbf{V}$.

The output weights $\mathbf{W} \in \mathbb{R}^{K\times C}$ are subsequently determined by:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} ||\mathbf{W}^T\mathbf{H} - \mathbf{T}||^2 + \lambda||\mathbf{W}||^2, \tag{2.19}$$

where $\mathbf{H} = [\mathbf{h}_1,\ldots,\mathbf{h}_N]$ is a matrix formed by the network's hidden layer outputs for the training data and $\mathbf{T}$ is a matrix formed by the network's target vectors $\mathbf{t}_i$, $i = 1,\ldots,n$. The network's output weights will be:

$$\mathbf{W} = \left(\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I}\right)^{-1}\mathbf{H}\mathbf{T}^T. \tag{2.20}$$

After obtaining the network's parameters $\mathbf{V}$ and $\mathbf{W}$, a new (test) sample $\mathbf{x} \in \mathbb{R}^D$ is mapped to its corresponding representations in spaces $\mathbb{R}^K$ and $\mathbb{R}^C$, i.e. $\mathbf{h} = \phi_{RBF}(\mathbf{x})$

and $\mathbf{o} = \mathbf{W}^T\mathbf{h}$, respectively. It is classified according to the maximal network output, i.e.:

$$l_{\mathbf{x}} = \arg\max_k \ o_k. \tag{2.21}$$

2.3.3 Multilayer Perceptron

Multilayer Perceptron (MLP) ([10], [48], [55]) is a set of functions which filter the input data in several steps. MLP (see Fig. 2.6) is similar to SLFN but with more hidden layers. Every hidden layer consists of nodes that will determine whether the information (i.e., input data from the previous layer) will continue to the next layer. This type of neural network shows a high degree of connectivity, and its weights define its strength. Usually, an MLP is used for supervised tasks where adjustment of weights takes place via several iterations between the input and the output layers through backpropagation.

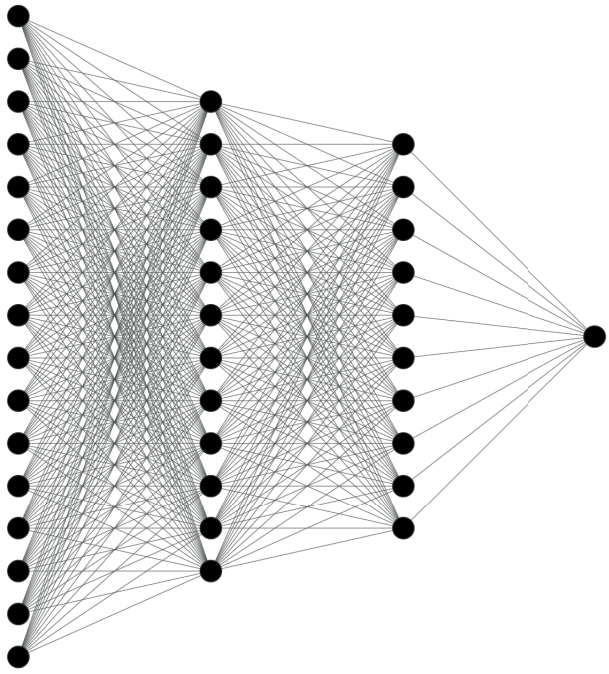


Figure 2.6 MLP with two hidden layers (the graph was created in NN-SVG).

One way to train an MLP is to use a sequential data feeding process called batch learning. Batch learning is a process during which the neural network adjusts the

synaptic weights after the presentation of all the samples $\mathbf{J} = \{\mathbf{x}(i), \mathbf{d}(i)\}_{i=1}^{N}$ in the training process, where: $\mathbf{x}(i)$ is the multi-dimensional input vector and $\mathbf{d}(i)$ is the response vector of the supervised problem at instance i where the error function at instance i is:

$$e^{(i)} = d^{(i)} - y^{(i)} \tag{2.22}$$

2.3.4 Convolutional Neural Network

Convolutional neural network (CNN) ([26], [29], [40]) is a type of neural network that exploits spatial correlations between neurons. CNN has the ability to share the so-called tied weights and equivariant representations properties. More specifically, sparse connectivity can be achieved by using a kernel smaller than the sample input to create a new filtered representation of the given signal. Based on this property, CNN reduces the amount of memory required during training. The second advantage that CNN has is the use of tied weights. Tied weights are shared among the inputs since the same amount of weights are applied to the inputs at each layer.

CNN topology varies according to the number of the main building blocks which are: the convolution layer, the pooling layer, and the fully connected layer. An example of a CNN architecture can be seen in Fig. 2.7.

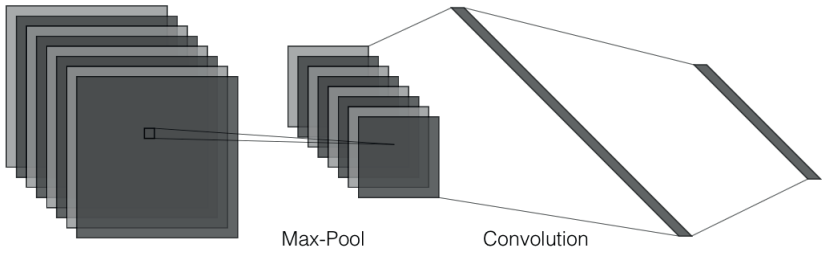


Figure 2.7 Example of a basic CNN architect (the graph was created in NN-SVG)

The main advantage of CNN is its ability to extract features from the multi-dimensional input signal which is usually expressed as a tensor or matrix. This process creates linear activations that run via a non-linear activation function (e.g., rectified linear activation function (ReLU), Leaky ReLU, etc). Then a pooling layer

based on a summary statistic related to the local outputs (e.g., max-pooling) will convert the local output. The last step of the process is the connection to the fully connected layers that will perform the classification (or regression) task. This task is based on discrete time series inputs, which formulate the (forward) convolution layer calculation as follows:

$$y_{i^{l+1},j^{l+1},d} = \sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{d=1}^{D} f_{i,j,d} \times x^{l}_{i^{l+1}+i,j^{l+1}+j,d} \tag{2.23}$$

where H, W, and D are the row, columns and depth dimension of the input tensor $\mathbf{x} \in \mathbb{R}^{H^l \times W^l \times D^l}$ respectively. $\mathbf{f} \in \mathbb{R}^{H^l \times W^l \times D^l}$ is the filter bank, and the indexing $(i^{l+1}+i, j^{l+1}+j, d)$ refers to the iterative local convolution of the filter bank on the suggested input for the l-layer. The last step is the use of fully connected layers.

2.3.5 Long Short-Term Memory

Prediction of sequential representations, such as time series, is a common objective in finance. The presence of temporal behavior of time series needs to be taken into consideration when it comes to model selection. A model which can effectively manage these dependencies is long- short-term memory neural (LSTM) network ([25], [34]). LSTM belongs to the family of recurrent neural networks (RNN) introduced in [50], which have feedback loops to help identify patterns in the data. LSTM is an extension of RNN where the difference lies in the fact an LSTM contains several internal gates that filter the input information more effectively. The motivation for choosing LSTM is its ability to create connections through time and account for the problem of vanishing (or exploding) gradients. Instead of just applying element-wise non-linear input transformations, LSTM units contain processes which take into consideration the sequential nature of time series. The primary function that explains LSTM's output is:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t; \eta) \tag{2.24}$$

where $\mathbf{h}$ and x are the state and the input at time t and η are the shared parameters for a transition function f at time t. LSTM cell is equipped with gates that will filter the information flow by applying weights internally via the sigmoid function

σ_g. The first pass is the forget gate vector f_i^t:

$$f_i^{(t)} = \sigma_g\Big(\sum_j W_{i,j}^f h_j^{(t-1)} + \sum_j U_{i,j}^f x_j^{(t)} + b_i^f\Big) \tag{2.25}$$

where $\mathbf{x}_i^{(t)}$ and $\mathbf{h}_i^{(t)}$ are the current input and hidden state vectors of cell i at time t, respectively. The corresponding weight matrices to these vectors are W^f and U^f for the forget gate vector with b^f the bias term. The next pass is related to the information that is going to be saved to the so-called "cell state". The cell state can be divided in two parts the input vector and a $tanh$ layer as follows:

$$C_i^{(t)} = f_i^{(t)} C_i^{(t-1)} + g_i^{(t)} tanh\Big(\sum_j W_{i,j}^C h_j^{(t-1)} + \sum_j U_{i,j}^C x_j^{(t)} + b_i^C\Big) \tag{2.26}$$

where $g^{(t)}$ is the input gate:

$$g_i^{(t)} = \sigma_g\Big(\sum_j W_{i,j}^g h_j^{(t-1)} + \sum_j U_{i,j}^g x_j^{(t)} + b_i^g\Big) \tag{2.27}$$

The last part is the filtered output. More specifically, the LSTM output/hidden state will be formulated by the output gate vector $o_i^{(t)}$, which can be calculated as follows:

$$o_i^{(t)} = \sigma_g\Big(\sum_j W_{i,j}^o h_j^{(t-1)} + \sum_j U_{i,j}^o x_j^{(t)} + b_i^o\Big) \tag{2.28}$$

and the final output $h_i^{(t)}$ is equal to:

$$h_i^{(t)} = o_i^{(t)} * tanh(C_i^{(t)}). \tag{2.29}$$

An example of the internal operations of an LSTM can be seen in Fig. 2.8

2.4 Feature Engineering

Feature engineering is the field of study that deals with data analysis and data transformation. This study facilitates the process of extracting information relevant to the problem/task under consideration. There are two main categories for feature extraction methods: handcrafted features and fully automated feature extraction methods.

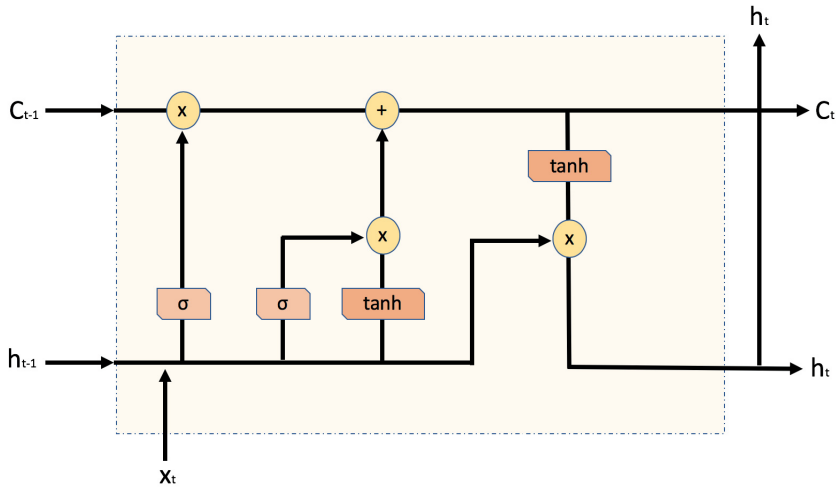


Figure 2.8 LSTM internal structure which shows how the information flows through gates.

Both categories require in-depth knowledge of data analysis and model development. More specifically, the former case refers to the process that features a set of vector representation which converts raw data into informative signal based on experts' opinion, whereas in the latter case a model (e.g., neural network) will extract itself the transformed representations.

Since the problem under consideration in the present thesis focuses on financial data (i.e., the mid-price prediction based on high-frequency data), relevant features/-factors are related to financial concepts. More specifically, these features will be LOB features, technical indicators, quantitative analysis, econometrics, and fully automated financial processes. The following list of features is by no means exhaustive, demonstrating only a few exemplary cases.

2.4.1 Limit Order Book Features

LOB is an ordering mechanism for the trading activity of the submitted limit and market orders. This trading activity creates a complex system in which relevant indicators/features should capture its dynamics. The work presented in [38] is an example of capturing LOB dynamics by handcrafted feature development. LOB features extraction is based on the prediction of mid-price. There are three main categories of feature set whose description can be seen in Table. 2.2

Table 2.2 LOB feature sets (obtained from [P1])

Feature Set	Description	Details
Basic	$u_1 = \{P_i^{ask}, V_i^{ask}, P_i^{bid}, V_i^{bid}\}_{i=1}^{n}$	10(=n)-level LOB Data
Time-Insensitive	$u_2 = \{(P_i^{ask} - P_i^{bid}), (P_i^{ask} + P_i^{bid})/2\}_{i=1}^{n}$	Spread & Mid-Price
	$u_3 = \{P_n^{ask} - P_1^{ask}, P_1^{bid} - P_n^{bid}, \lvert P_{i+1}^{ask} - P_i^{ask}\rvert, \lvert P_{i+1}^{bid} - P_i^{bid}\rvert\}_{i+1}^{n}$	Price Differences
	$u_4 = \left\{\frac{1}{n}\sum_{i=1}^{n} P_i^{ask}, \frac{1}{n}\sum_{i=1}^{n} P_i^{bid}, \frac{1}{n}\sum_{i=1}^{n} V_i^{ask}, \frac{1}{n}\sum_{i=1}^{n} V_i^{bid}\right\}$	Price & Volume Means
	$u_5 = \left\{\sum_{i=1}^{n}(P_i^{ask} - P_i^{bid}), \sum_{i=1}^{n}(V_i^{ask} - V_i^{bid})\right\}$	Accumulated Differences
Time-Sensitive	$u_6 = \left\{dP_i^{ask}/dt, dP_i^{bid}/dt, dV_i^{ask}/dt, dV_i^{bid}/dt\right\}_{i=1}^{n}$	Price & Volume Derivation
	$u_7 = \left\{\lambda_{\Delta t}^1, \lambda_{\Delta t}^2, \lambda_{\Delta t}^3, \lambda_{\Delta t}^4, \lambda_{\Delta t}^5, \lambda_{\Delta t}^6\right\}$	Average Intensity per Type
	$u_8 = \left\{\mathbf{1}_{\lambda_{\Delta_t}^1 > \lambda_{\Delta_T}^1}, \mathbf{1}_{\lambda_{\Delta_t}^2 > \lambda_{\Delta_T}^2}, \mathbf{1}_{\lambda_{\Delta_t}^3 > \lambda_{\Delta_T}^3}, \mathbf{1}_{\lambda_{\Delta_t}^4 > \lambda_{\Delta_T}^4}, \mathbf{1}_{\lambda_{\Delta_t}^5 > \lambda_{\Delta_T}^5}, \mathbf{1}_{\lambda_{\Delta_t}^6 > \lambda_{\Delta_T}^6}\right\}$	Relative Intensity Comparison
	$u_9 = \{d\lambda^1/dt, d\lambda^2/dt, d\lambda^3/dt, d\lambda^4/dt, d\lambda^5/dt, d\lambda^6/dt\}$	Limit Activity Accelaration

These three sets of handcrafted features provide useful insight regarding the evolution of price rate changes, or volume rate changes, but they are limited in terms of other LOB characteristics. Several additional factors/features should also be considered, such as volatility estimation, price trends, volume imbalance, etc.

2.4.2 Technical Analysis

Technical analysis is a trading universe where traders make investment predictions based on the idea that stock price is the primary source of information. The central component of technical analysis is the utilization of charts of volume and price trends. Several indicators developed through the years (e.g., candlestick charts exist since 1600) and fuse with other methods, such as statistics. Some examples of technical indicators are:

Average Directional Index

Average directional index (ADX) indicator [56] developed in order to identify the strength of a current trend. ADX is calculated as follows:

$$TR = max(H_t - L_t, |H_t - CL_{t-1}|, |L_t - CL_{t-1}|) \tag{2.30}$$

$$(+DM) = H_t - H_{t-1} \tag{2.31}$$

$$(-DM) = L_t - L_{t-1} \tag{2.32}$$

$$TR_{14} = TR_{t-1} - (TR_{t-1}/14) + TR \tag{2.33}$$

$$(+DM_{14}) = (+DL_{t-14}) - ((+DL_{t-14})/14) + (+DM) \tag{2.34}$$

$$(-DM_{14}) = (-DL_{t-14}) - ((-DL_{t-14})/14) + (-DM) \tag{2.35}$$

$$(+DI_{14}) = 100 \times ((+DM_{14})/(+TR_{14})) \tag{2.36}$$

$$(-DI_{14}) = 100 \times ((-DM_{14})/(-TR_{14})) \tag{2.37}$$

$$DI_{diff_{14}} = |(+DM_{14}) - (-DM_{14})| \tag{2.38}$$

$$DI_{sum_{14}} = |(+DM_{14}) + (-DM_{14})| \tag{2.39}$$

$$DX = 100 \times ((DI_{diff_{14}})/(DI_{sum_{14}})) \tag{2.40}$$

$$ADX = (ADX_{t-1} \times 13) + DX)/14 \tag{2.41}$$

where: TR is the true range, H_t is the current 10-block's highest MB price, L_t is the current 10-block's lowest MB price, CL_t is the previous 10-block's closing MB price, $(+DM)$ is the positive Directional Movement (DM), $(-DM)$ is the negative DM, TR_{14} is the TR based on the previous 14-blocks, TR_{t-1} is the previous TR price, $(+DM_{14})$ is DM based on the previous 14 $(+DM)$ blocks, $(-DM_{14})$ is the DM based on the previous 14 $(-DM)$ blocks, $(+DM_{t-14})$ is the (+DM) of the previous 14 $+DM$ blocks, $DI_{diff_{14}}$ is the directional indicator (DI) of the difference between $(+DM_{14})$ and $(-DM_{14})$, $DI_{sum_{14}}$ is the DI of the sum between $(+DM_{14})$ and $(-DM_{14})$, DX is the directional movement index and ADX_{t-1} is the previous average directional index.

Parabolic Stop and Reverse

Parabolic SAR (PSAR) [57] is a trend following indicator that protects profits. There

are two main modules for its calculation, the Rising SAR and the Falling SAR, and they are calculated as follows:

- Rising SAR

$$EP = H_{High_5} \tag{2.42}$$

$$SAR = SAR_{t-1} + AF_{t-1}(EP_{t-1} - SAR_{t-1}) \tag{2.43}$$

- Falling SAR

$$EP = L_{Low_5} \tag{2.44}$$

$$SAR = SAR_{t-1} - AF_{t-1}(EP_{t-1} - SAR_{t-1}) \tag{2.45}$$

where AF is the acceleration factor, and EP the extreme point with highest high (H_{High_5}) and lowest low (L_{Low_5}) for the last five 10-MB blocks.

Linear Regression Line

Linear regression line (LRL) is a basic statistical method that provides information for a future projection wherein trading is used to capture overextended price trends. The basic calculation steps are as follows:

$$PV = c_1 + c_2 \times MB_{prices} \tag{2.46}$$

$$c_2 = r \times (std_{PV}/std_{MB_{prices}}) \tag{2.47}$$

$$r = \frac{\left(\sum_{i=1}^{10}(MB_{prices}(i) - \overline{MB_{prices}})(PV(i) - \overline{PV})\right)}{\left(\sqrt{\sum_{i=1}^{10}(MB_{prices}(i) - \overline{MB_{prices}})^2(\sum_{i=1}^{10}(PV(i) - \overline{PV})^2}\right)} \tag{2.48}$$

$$c_1 = \overline{PV} - c_2 \times \overline{MB_{prices}} \tag{2.49}$$

where PV are the predicted values, r is the correlation coefficient, and $\overline{MB_{prices}}$ and $\overline{PV}$ are the mean of MB prices and predicted values, respectively.

2.4.3 Quantitative Analysis

Quantitative analysis ([1], [2]) belongs to a trading universe where a decision is made based on mathematical models and statistics. This type of analysis can effectively be used not only for pattern identification in massive datasets but also to mitigate investment risks. One of the goals of the thesis is to use quantitative analysis for pattern recognition. Examples of features that utilize mathematical concepts and statistics follow:

Autocorrelation and Partial Correlation
Autocorrelation and partial correlation [11], [23] are key features in the development of time series analysis. Under the assumption of stationary stochastic processes, their local behavior is:

- Autocorrelation

$$r_{X,k} = \frac{E[(x_i - \mu)(x_{i+k} - \mu)]}{\sqrt{E[(x_i - \mu)^2]E[(x_{i+k} - \mu)^2]}} \tag{2.50}$$

  where x_i and x_{i+k} are the time series of lag k at time instance i, $\mu = E[x_i] = \int_{-\infty}^{\infty} x\,p(x)dx$ and $\sigma_x^2 = E[(x_i - \mu)^2] = \int_{-\infty}^{\infty}(x-\mu)^2 p(x)dx$ are the constant mean and constant variance, respectively.

- Partial Correlation
  - For the general case of an autoregressive model $AR(p)$, we have:

$$x_{i+1} = \phi_1 x_i + \phi_2 x_{i-1} + ... + \phi_p x_{i-p+1} + \xi_{i+1} \tag{2.51}$$

    of lag 1 up to p follows:

$$< x_i x_{i+1} >= \sum_{j=1}^{p} (\phi_j < x_i x_{i-j+1} >) \tag{2.52}$$

$$< x_{i-1} x_{i+1} >= \sum_{j=1}^{p} (\phi_j < x_{i-1} x_{i-j+1} >) \tag{2.53}$$

$$< x_{i-k+1} x_{i+1} >= \sum_{j=1}^{p} (\phi_j < x_{i-k+1} x_{i-j+1} >) \tag{2.54}$$

$$< x_{i-p+1}x_{i+1} >= \sum_{j=1}^{p} (\phi_j < x_{i-p+1}x_{i-j+1} >) \tag{2.55}$$

by diving with $N-1$ and autocovariance of zero separated periods (where the autocovariance function is even), all the lag periods above will be:

$$r_i = \sum_{j=1}^{p} \phi_j r_{j-1} \tag{2.56}$$

where $r_i, i = 1,2,...,k,...,p$ for the matrix operations $\boldsymbol{R}\boldsymbol{\Phi} = \boldsymbol{r}$, $\boldsymbol{R} \in \mathbb{R}^{p\times p}$, $\boldsymbol{\Phi} \in \mathbb{R}^{p\times 1}$ and $\boldsymbol{r} \in \mathbb{R}^{p\times 1}$. The symmetric and full rank $\boldsymbol{\Phi}$ are as follows: $\hat{\boldsymbol{\Phi}} = \boldsymbol{R}^{-1}\boldsymbol{r}$.

- *Yule-Walker* equations calculation for a lagged interval $1 \leqslant i \leqslant p$:

$$\hat{\boldsymbol{\Phi}} = \left(\boldsymbol{R}^{(i)}\right)^{-1} \boldsymbol{r}^{(i)} = \begin{bmatrix} \hat{\phi}_1 \\ \hat{\phi}_2 \\ \vdots \\ \hat{\phi}_i \end{bmatrix} \tag{2.57}$$

Cointegration

Investigation of time-series equilibrium [22], [32] can be tested through the cointegrated hypothesis. Utilizing the cointegration test will help ML traders avoid the problem of spurious regression. For this reason an Engle-Granger (EG) test is utilized for the multivariable case of LOB ask (A_t) and bid (B_t) times series. The formulation of the EG test for the ask and bid LOB prices are as follows:

- A_t and $B_t \sim I(d)$, where $I(d)$ represents the order of integration
- Cointegration equation based on the error term: $u_t = A_t - \alpha B_t$, where ordinary least squares (OLS) is performed for the estimation of α
- EG Hypothesis: $u(t) \sim I(d)$, $d \neq 0$
- Perform OLS for the estimation of $\hat{\alpha}$ and unit root test for: $\hat{u} = A_t - \hat{\alpha} B_t$

2.4.4 Econometrics

Econometrics ([46], [51], [59]) is the field of study that covers topics related to financial applications and assumptions under the scope of statistics and mathematics. This field is vast and covers methods from regression and probability theory to asymptotic theory and volatility estimation. These topics are directly connected to high-frequency financial data, which come with variation in prices, known in the financial literature as volatility. Since the thesis deals with this type of data, some features will be presented in this section as examples of feature engineering under econometric theory.

Volatility Measures

Features based on volatility measures estimate either the integrated variance (IV), that is, the process

$$IV_t = \int_0^t \sigma_s^2 \mathrm{d}s \tag{2.58}$$

or, more generally, the quadratic variation (QV)

$$[X,X]_t = \int_0^t \sigma_s^2 \mathrm{d}s + \sum_{0<s\leq t} (\zeta_s \mathrm{d}N_s)^2. \tag{2.59}$$

Here, X is the logarithmic price of a given asset. Assuming that X_t follows an Itô semimartingale; that is,

$$X_t = X_0 + \int_0^t b_s \mathrm{d}s + \int_0^t \sigma_s \mathrm{d}W_s + \int_o^t \zeta_s \mathrm{d}N_s, \tag{2.60}$$

where b is locally bounded, σ is cádlág and predictable, and W_t is a standard Wiener process, ζ is a thin (i.e., finite) process mapping the jump size, and N is the counting process associated to the jump times of X. Δ_n defines the time elapsed between two adjacent observations; specifically, assuming that the observations are equidistant in time: $\Delta_n = \lfloor \frac{t}{n} \rfloor$. Since calendar time is not present, then: $\Delta_n = \frac{1}{n}$.

- Realized variance
  The realized variance [4] is the most natural estimator of the quadratic varia-

tion process and is equal to:

$$RV_t = \sum_{i=1}^{n} (r(X)_i)^2, \tag{2.61}$$

where $r(X)_i$ are the log-returns of the X_i logarithmic stock price.

- Realized kernel
  Realized kernels [7] are used in obtaining a noise robust estimate of QV as follows:

$$RK_t = \gamma_0(X_{\Delta_n}) + \sum_{h=1}^{H} k\left(\frac{h}{H}\right)\{\gamma_h(X_{\Delta_n}) + \gamma_{-h}(X_{\Delta_n})\}, \tag{2.62}$$

  with H the kernel bandwidth, $\gamma_h(X_{\Delta_n})$ the autocovariation process, k is the kernel function of choice. For a non-flat-top Parzen the implementation follows closely [8].

- Realized pre-averaged variance
  The pre-averaged realized variance [37] is akin to the realized kernel estimator (in fact they are asymptotically equivalent). As for the realized kernel, the pre-averaged realized variance is used in retrieving a noise-free measurement of the quadratic variation of the price process and is calculated as follows:

$$PA-RV_t = \frac{\sqrt{\Delta_n}}{\theta\psi_2} \sum_{i=0}^{n-H+1} (\bar{X}_i^n)^2 - \frac{\psi_1 \Delta_n}{2\theta^2\psi_2} \sum_{i=0}^{N} (r(X))^2. \tag{2.63}$$

  As before, H is the kernel bandwidth and θ the pre-averaging horizon. Further, given a nonzero real-valued function $g : [0,1] \to \mathbb{R}$ with $g(0) = g(1) = 0$ and which is further continuous and piecewise continuously differentiable such that its derivative g' is piecewise Lipschitz. Then:

$$\psi_1 = \int_0^1 (g'(s))^2 \mathrm{d}s \tag{2.64}$$

  and

$$\psi_2 = \int_0^1 (g(s))^2 \mathrm{d}s. \tag{2.65}$$

  Following [12] and for $H = \theta\sqrt{n}$ and $\theta = 1$, $g(x) = x \wedge (1-x)$. Hence $\psi_1 = 1$

and, $\psi_2 = \frac{1}{12}$.

2.4.5 Fully Automated Feature Extraction

Feature extraction can be performed not only by hand-crafted features but also through fully automated processes. An example of these processes is the so-called autoencoders (AE). Autoencoders [41] are neural networks that operate in an unsupervised manner. They do not require any labeling system since they operate under a semi-supervised protocol. More specifically, they use the input data as label and the model training is calculated through this mechanism. The main parts of an AE are: the encoding part, the latent representation, and the decoding part as seen in Fig. 2.9

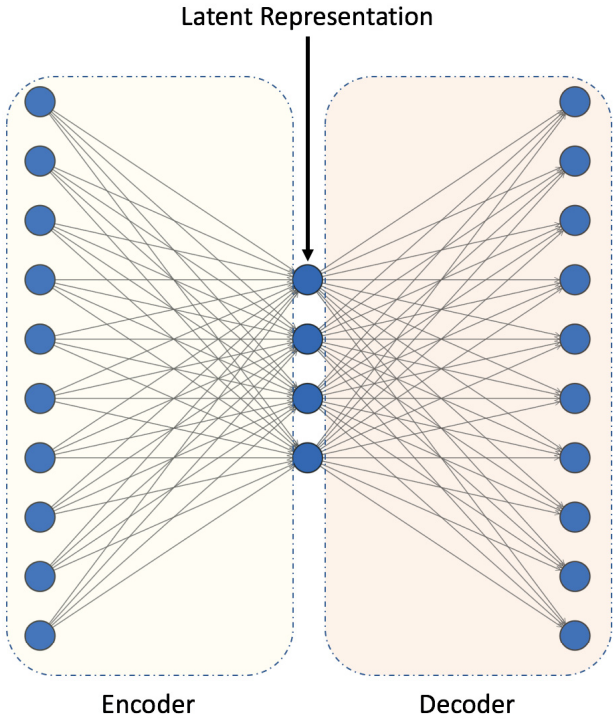


Figure 2.9 Example of a symmetrical undercomplete AE

The basic structure of AE is defined as a mapping from the encoder to the decoder

with the main objective being the following minimization problem:

$$f, g = \underset{f,g}{\operatorname{argmin}} ||X - (f \circ g)X||^2 \tag{2.66}$$

where $f : X \to F$ and $g : F \to X$ are the transition functions and $\circ$ is the function composition operation.

3 DATASETS, PROTOCOLS AND METRICS

In this chapter, a description of the experimental setup will be presented. The present thesis deals with the task of LOB's mid-price prediction. Datasets used for this task are two TotalView-ITCH sets based on Nordic and US stocks. More specifically, experiments were conducted on five Nordic stocks (i.e., Kesko Oyj, Outokumpu Oyj, Sampo Oyj, Rautaruukki, and Wartsila Oyj) and two US stocks (i.e., Amazon and Google). These datasets contain daily trading activity in the form of MB events. Processing this flow of information requires intensive computational power together with data processing such as filtering and normalization. Furthermore, one of the critical components of predicting mid-price is the development of the experimental protocols that formulate the problem under consideration correctly and specify how the experiments are conducted.

3.1 Datasets

Nordic

The Nordic dataset is obtained from NASDASQ Nordic and consists of five Nordic stocks (Table 3.1) with 4603143 trading events in millisecond resolution for the entire trading period, which is from 06/01/2010 until 06/14/2010.

Table 3.1 Nordic Stocks

Stock	Sector	Industry	ISIN Code
Kesko Oyj	Consumer Defensive	Grocery Stores	FI0009000202
Outokumpu Oyj	Basic Materials	Steel	FI0009002422
Sampo Oyj	Financial Services	Insurance - Property & Casualty	FI0009003305
Rautaruukki	Basic Materials	Steel	FI0009003552
Wartsila Oyj	Industrials	Diversified Industrials	FI0009000727

Trading activity in terms of trade side (i.e., bid and ask) and trade type (i.e., trades,

order, and cancellations) for the diversified Nordic dataset is presented in Table 3.2. For every stock the proportions of trades:orders:cancellations is the same for the entire dataset.

Table 3.2 Trading activity of the five Nordic stocks

Stock	Number of Events	Bid : Ask	Trades : Orders : Cancellations
Kesko Oyj	341913	146472 : 195441	8216 : 177479 : 156218
Outokumpu Oyj	827666	423372 : 404294	30923 : 416139 : 380604
Sampo Oyj	734055	363185 : 370870	24299 : 368675 : 341081
Rautaruukki	1015838	519280 : 496558	18660 : 510647 : 486531
Wartsila Oyj	1683671	846104 : 837567	16210 : 842840 : 824621

US

The US dataset was obtained from NASDASQ Nordic and consists of two stocks (Table 3.3) with 4603143 trading events for the entire trading period, which is from 09/22/15 to 10/05/15.

Table 3.3 US Stocks

Stock	Sector	Industry	ISIN Code
Amazon	Consumer Cyclical	Specialty Retail	US0231351067
Google	Technology	Internet Content & Information	US02079K3059

Trading activity in terms of trade side (i.e., bid and ask) and trade type (i.e., trades, order, and cancellations) for the US dataset is presented in Table 3.4. For every stock the proportions of trades:orders:cancellations is the same for the entire dataset.

Table 3.4 Trading activity of the two US stocks

Stock	Number of Events	Bid : Ask	Trades : Orders : Cancellations
Amazon	5144115	2813419 : 2330696	413752 : 2467835 : 2262528
Google	4135017	2161713 : 1973304	112946 : 2047697 : 1974374

3.2 Normalization

Normalization is the process where the input representations are scaled in order to operate under a uniform range. This is particularly helpful in the case of neural networks. More specifically, normalization facilitates the convergence, during training,

of biases and weights. Experiments presented in this thesis utilized three different normalization setups: z-score, min-max, and decimal precision normalization, for every i data sample.

Z-score

Z-score, in particular, is the normalization process through which the mean is subtracted from the input data for each feature separately and divided by the standard deviation of the given sample:

$$\mathbf{x}_i^{(Z_{score})} = \frac{\mathbf{x}_i - \bar{\mathbf{x}}}{\sqrt{\frac{1}{N}\sum_{j=1}^{N}(\mathbf{x}_j - \bar{\mathbf{x}})^2}}, \tag{3.1}$$

where $\bar{\mathbf{x}}$ is the sample mean:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_j \tag{3.2}$$

.

Min-Max

On the other hand, min-max scaling, as described by:

$$\mathbf{x}_i^{(MM)} = \frac{\mathbf{x}_i - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}}, \tag{3.3}$$

is the process of subtracting the minimum value from each feature and dividing it by the difference between the maximum and minimum value of that feature sample.

Decimal Precision

The third scaling setup is the decimal precision approach. This normalization method is based on moving the decimal points of each of the feature values. Calculations follow the absolute value of each feature sample:

$$\mathbf{x}_i^{(DP)} = \frac{\mathbf{x}_i}{10^k}, \tag{3.4}$$

where k is the integer that will give us the maximum value for $|\mathbf{x}^{(DP)}| < 1$.

3.3 Protocols

Experimental protocols define the objective and the main steps of the experiment. This thesis makes use of three different experimental protocols. All protocols deal with the task of mid-price movement prediction, as follows:

Anchored Forward Cross-Validation

A day-based prediction framework is developed following an anchored forward cross-validation format. More specifically, the training set increases by one day in each fold and stops after $n-1$ days (i.e., after nine days with $n = 10$). On each fold, the test set corresponds to one day of data, which moves in a rolling window format (i.e., every next day works as the testing set). The experimental setup is illustrated in Fig. 3.1.

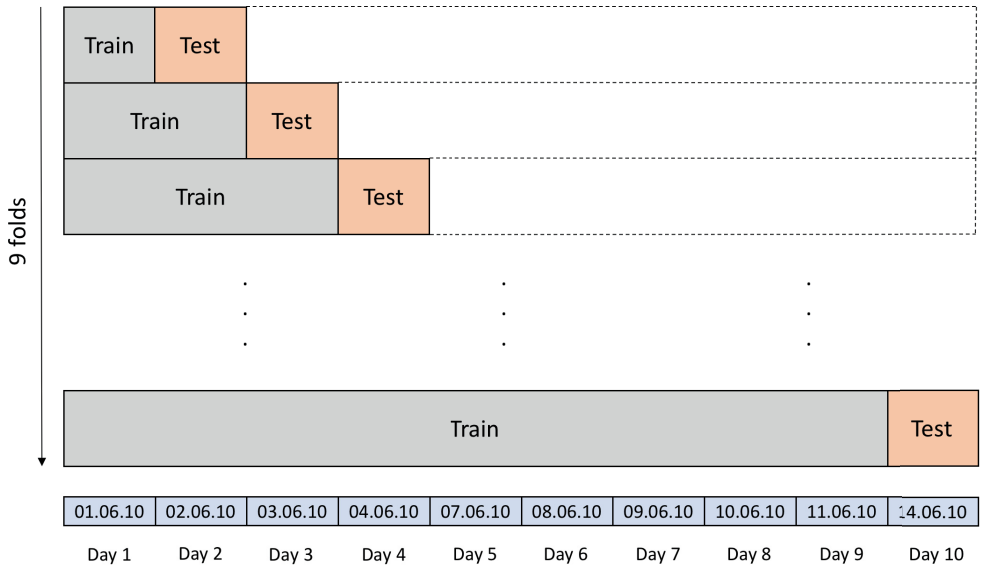


Figure 3.1 Experimental Setup Framework based on an Anchored Forward cross-validation format ([P1]).

Independent Feature Blocks

Independent Feature Blocks are based on independent 10-event blocks for extraction of feature representations as seen in Fig. 3.2. Feature representations are based on independent 10-event blocks. The protocol considers the problem of mid-price movement prediction as a three-class classification problem, wherein mid-price's states are up, down, and stationary conditions. These changes in mid-price are defined

through the following calculations:

$$l_t = \begin{cases} 1, & \text{if} \frac{m_a(t)}{MP(t)} > 1+\alpha \\ -1, & \text{if} \frac{m_a(t)}{MP(t)} < 1-\alpha \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

where $MP(t)$ is the mid-price at time t, $m_a(t) = \frac{1}{r}\sum_{i=1}^{r} MP(t+1)$ is the average of the future mid-price events with window size $r = 10$, and α determines the significance of the mid-price movement, which is equal to 2×10^{-5}.

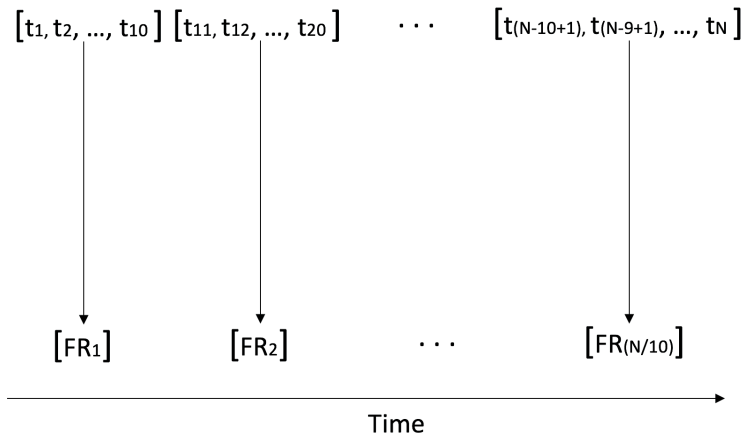


Figure 3.2 Experimental Setup Framework based on independent feature blocks (i.e., $t_1, ..., t_{10}$, $t_{11}, ..., t_{20}$, ..., etc) format. This format constructs the input feature representations (FR) ([P3]).

Online Learning

Online learning protocol utilizes all the events in an online sequence. Feature representations extraction takes place over a sliding window of ten events with an overlap of nine events. As a result, there is no missing value during the feeding process of a classifier such as neural networks. A pictorial representation of the feature extraction is shown in Fig. 3.3. The task of this protocol is to predict the movement of mid-price (i.e., classification: up or down) together with the number of events it takes for that movement to occur in the future (i.e., regression: number of events until next mid-price's movement change).

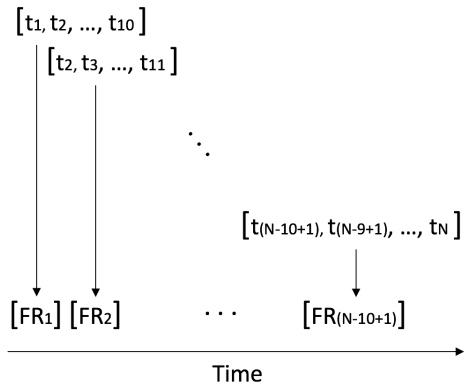


Figure 3.3 Experimental Setup Framework based on online learning format ([P3]). This format is constructing the input feature representations (FR).

3.4 Performance Evaluation

Performance evaluation is a critical component in machine learning. The task of mid-price prediction is treated as a classification and regression problem. The classification task refers to the prediction of mid-price's direction (i.e., up, down or stationary condition), while the regression part is related to the forecasting of the number of trading events that will take place until the next mid-price movement change. For the classification part, the metric of F1 score is used whereas for the regression task, the root mean squared error (RMSE) is employed. The advantages of these metrics are summarized as follows: i) F1 score can only be affected in one direction by skewed distributions for unbalanced classes, such as the datasets in these experiments, and ii) RMSE, which is used to measure the distance between the regression line and the data points, is easily interpretable since it retains the error measurement under the same unit as the data points. A description of these performance measures follows:

F1 Score

Performance is based on the calculation of the mean accuracy, recall, precision, and F1 score. These metrics are defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.6}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.7}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.8}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.9}$$

where TP and TF represent the true positives and true negatives, respectively, of the mid-price prediction label compared with the ground truth, while FP and FN represent the false positives and false negatives, respectively. The focal point is F1 score performance.

Root Mean Squared Error

Root mean squared error measures the average magnitude of error. It is suitable for continuous processes, wherein measurement units are comparable to units of the variable of interest. It is calculated as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}}, \tag{3.10}$$

where P_i and O_i are the predicted and observed values of n samples, respectively.

4 CONTRIBUTIONS

In this chapter, the contributions related to this thesis will be briefly summarized. First, the Nordic benchmark dataset and the baseline performance will be described for the task of mid-price movement prediction in Section 4.1. Next, a feature selection process will be presented in Section 4.2, where an extensive list of features derived from technical and quantitative analysis together with LOB features are compared with a newly introduced feature based on logistic regression. This new feature, which is designed for online learning, was selected first among the other suggested features. The selection process was tested on several sorting methods such as entropy, linear discriminant analysis, and least mean square. Furthermore, in Section 4.3, a feature engineering approach will be discussed in terms of econometric features. Finally, the effectiveness of using deep learning for the task of mid-price prediction will be presented in the same section.

4.1 Nordic Benchmark Dataset

HF data is a collection of millions of trading events with time irregularities. This collection can be effectively organized in LOB filtered by price and time. Proper analysis of LOB dynamics (see Section 2.2) requires the development of a suitable experimental protocol as well as an adequate model selection and parameterization in order to effectively make predictions. To our knowledge, there are no publicly available LOB datasets ready for machine learning tasks. This therefore, is one of the main contributions of the present thesis (see [P1]), as it provides a publicly available[1] dataset equipped with three normalizations setups (i.e., z-score, min-max, and decimal precision) together with annotations for five classification problems depending on the forecasting horizon (see Section 3.1 and Section 3.3). Characteristics of other

[1]LOB dataset can be found in https://etsin.avointiede.fi/dataset/urn-nbn-fi-csc-kata20170601153214969115

datasets are provided in Table 4.1.

Table 4.1 HF Dataset Examples ([P1])

	Dataset	Public Avl.	Unit Time	Period	Asset Class / Num. of Stocks	Size	Annotations
1	Dukascopy	✓	ms	up-to-date	various	≈ 20,000 events/day	x
2	truefx	✓	ms	up-to-date	15 FX pairs	≈ 300,000 events/day	x
3	NASDAQ	AuR	ms	2008-09	Equity / 120	-	x
4	NASDAQ	AuR	ms	10/07 & 06/08	Equity / 500	≈ 55,000 events/day	x
5	NASDAQ	x	ms	-	Equity / 5	2,000 data points	x
6	Euronext	AuR	-	-	Several Products	-	x
7	NASDAQ	x	ns	01/14-08/15	Equity / 489	50 TB	x
8	Our - NASDAQ	✓	ms	01-14/06/10	Equity / 5	4 M samples	✓

The dataset is divided into two main categories depending on the presence or absence of the auction period. Since the anchored cross-validation method for ten days for five stocks is followed, there are nine (cross-fold) datasets for each normalization set-up for training and testing. Every train and test dataset contains information from all the five Nordic stocks. For instance, the first fold contains one day of training and one day of testing for all the five stocks. The second fold contains the training dataset for two days and the testing dataset for one day and so on.

Another contribution of [P1] is the provided baseline models, which are grounded on linear and non-linear regressions methods, and more specifically, on RR and SLFN with k-means (see Section 2.3.1 and Section 2.3.2, respectively). Experimental results are presented for four cases: one is based on feature representations without normalization filtering and three cases with normalized representations (i.e., z-score, min-max, and decimal precision). As seen in Table 4.2 and Table 4.3 for five different classification tasks (i.e., Labels 1, 2, 3, 5, and 10)[2], the outcome suggests that normalization-depending horizon projection offers significant performance improvements for both RR and SLFN.

[2]Labels 1, 2, 3, 5 and 10 represent five different classification problems related to the forecasting horizon. For instance: Label 1 represents the next mid-price movement, Label 2 represents the mid-price movements in two events from the current state, Label 3 represents the mid-price movements three events away from the current state etc.

Table 4.2 Results based on Unfiltered Representations ([P1])

Labels	$RR_{Accuracy}$	$RR_{Precision}$	RR_{Recall}	RR_{F1}
1	0,637 ± 0,055	0,505 ± 0,145	0,337 ± 0,003	0,268 ± 0,014
2	0,555 ± 0,064	0,504 ± 0,131	0,376 ± 0,023	0,320 ± 0,050
3	0,489 ± 0,061	0,423 ± 0,109	0,397 ± 0,031	0,356 ± 0,070
5	0,429 ± 0,049	0,402 ± 0,113	0,425 ± 0,038	0,400 ± 0,093
10	0,453 ± 0,054	0,400 ± 0,105	0,400 ± 0,030	0,347 ± 0,066
Labels	$SLFN_{Accuracy}$	$SLFN_{Precision}$	$SLFN_{Recall}$	$SLFN_{F1}$
1	0,636 ± 0,055	0,299 ± 0,075	0,335 ± 0,002	0,262 ± 0,015
2	0,536 ± 0,069	0,387 ± 0,132	0,345 ± 0,009	0,260 ± 0,035
3	0,473 ± 0,074	0,334 ± 0,080	0,357 ± 0,005	0,270 ± 0,021
5	0,381 ± 0,038	0,342 ± 0,058	0,370 ± 0,020	0,327 ± 0,043
10	0,401 ± 0,039	0,284 ± 0,102	0,356 ± 0,020	0,290 ± 0,070

Table 4.3 Results based on Z-score Normalization ([P1])

Labels	$RR_{Accuracy}$	$RR_{Precision}$	RR_{Recall}	RR_{F1}
1	0,480 ± 0,040	0,418 ± 0,021	0,435 ± 0,029	0,410 ± 0,022
2	0,498 ± 0,052	0,444 ± 0,025	0,443 ± 0,031	0,440 ± 0,031
3	0,463 ± 0,045	0,438 ± 0,027	0,437 ± 0,033	0,433 ± 0,034
5	0,439 ± 0,042	0,436 ± 0,028	0,433 ± 0,028	0,427 ± 0,041
10	0,429 ± 0,046	0,429 ± 0,028	0,429 ± 0,043	0,416 ± 0,044
Labels	$SLFN_{Accuracy}$	$SLFN_{Precision}$	$SLFN_{Recall}$	$SLFN_{F1}$
1	0,643 ± 0,056	0,512 ± 0,037	0,366 ± 0,019	0,327 ± 0,046
2	0,556 ± 0,066	0,550 ± 0,029	0,378 ± 0,011	0,327 ± 0,030
3	0,512 ± 0,069	0,497 ± 0,024	0,424 ± 0,047	0,389 ± 0,082
5	0,473 ± 0,036	0,468 ± 0,024	0,464 ± 0,028	0,459 ± 0,031
10	0,477 ± 0,048	0,453 ± 0,056	0,432 ± 0,025	0,410 ± 0,040

4.2 Feature Selection for Technical and Quantitative Indicators

Feature selection is a key component in experiments with multidimensional datasets. There are three main reasons why a trader, for instance, should consider performing feature selection: 1) to reduce computational complexity, 2) to improve perfor-

mance, and 3) to gain a better insight to any underlying process in the problem under consideration. Feature selection, together with feature engineering, has the potential to provide an information edge in trading, the benefits of which can be seen in [P2]. Below is a brief discussion of the newly introduced features based on technical and quantitative indicators and the feature selection criteria.

The experiments in [P2] evaluate an extensive list of technical indicators[3] and several quantitative indicators together with LOB features. A new quantitative feature based on logistic regression (see Table 4.4), which is suitable for online learning, is further introduced. The new feature was selected first in terms of several feature sorting methods, such as 1) entropy, 2) linear discriminant analysis (LDA), and 3) least mean square (LMS). The selection process follows a wrapper methodology, wherein the performance (i.e., F1 score) is reported in terms of the best subset of features for every iteration (e.g., the best single feature, the best pair of features, ...). The wrapper method is described in Algorithm 1.

Algorithm 1 Wrapper-Based Feature Selection ([P2])

procedure $l_{opt} = Feature\_Select(X, labels, criterion)$
  $l = [1:D]$
  $l_{opt} = [\ ]$
  $X_{opt} = [\ ]$, $[D, N] = size(X)$
  for $d = 1:D$ **do**
    $crit\_list = [\ ]$
    for $i = 1:D-d+1$ **do**
      $curr\_X = [X_{opt}; X(i,:)]$
      $crit\_list[i] = crit(curr\_X)$
    $[best\_d, best\_crit] = opt(crit\_list)$
    $l_{opt}[d] = best\_d$
    $l[best\_d] = [\ ]$
    $X_{opt} = [X_{opt}; X(best\_d,:)]$
    $X(best\_d,:) = [\ ]$

[3]This list is by far the most extensive in the literature.

Table 4.4 Feature list of the three groups (description and calculations in [P2]).

Feature Sets	**Description**
Technical Analysis	
Accumulation Distribution Line	Awesome Oscillator
Accelerator Oscillator	Average Directional Index
Average Directional Movement Index Rating	Displaced Moving Average based on Williams
Alligator Indicator	Absolute Price Oscillator
Aroon Indicator	Aroon Oscillator
Average True Range	Bollinger Bands
Ichimoku Clouds	Chande Momentum Oscillator
Chaikin Oscillator	Chandelier Exit
Center of Gravity Oscillator	Donchian Channels
Double Exponential Moving Average	Detrended Price Oscillator
Heikin-Ashi	Highest High and Lowest Low
Hull MA	Internal Bar Strength
Keltner Channels	Moving Average Convergence/Divergence Oscillator
Median Price	Momentum
Variable Moving Average	Normalized Average True Range
Percentage Price Oscillator	Rate of Change
Relative Strength Index	Parabolic Stop and Reverse
Standard Deviation	Stochastic Relative Strength Index
T3-Triple Exponential Moving Average	Triple Exponential Moving Average
Triangular Moving Average	True Strength Index
Ultimate Oscillator	Weighted Close
Williams %R	Zero-Lag Exponential Moving Average
Fractals	Linear Regression Line
Digital Filtering: Rational Transfer Function	Digital Filtering: Savitzky-Golay Filter
Digital Filtering: Zero-Phase Filter	Remove Offset and Detrend
Beta-like Calculation	
Quantitative Analysis	
Autocorrelation	Partial Correlation
Cointegration based on Engle-Granger test	Order Book Imbalance
Logistic Regression for Online Learning	
LOB Features	
n levels of LOB Data	Spread & Mid-Price
Price Differences	Price & Volume Means
Accumulated Differences	Price & Volume Derivation
Average Intensity per Type	Relative Intensity Comparison
Limit Activity Acceleration	

4.2.1 Adaptive Logistic Regression Feature

An adaptive logistic regression model is built for the task of mid-price movement prediction. Based on [52], the idea of this new feature is the local behavior of LOB levels. More specifically, an adaptive learning rate is developed based on the Hessian matrix and the ratio of the logistic coefficients. This ratio considers the relationship between the first six LOB levels and the ones deeper in LOB. The feature is constructed as follows: since $0 \leqslant h_\theta(V) \leqslant 1$ and V are the stock volumes for the first six levels of the LOB then the logistic function (i.e. hypothesis function) is:

$$h_\theta(V) = \frac{1}{1+e^{-\theta^T V}} \tag{4.1}$$

where $\theta^T V = \theta_0 + \sum_{j=1}^{n} \theta_j V_j$. Parameter estimation is considered by calculating the parameter's likelihood:

$$L(\theta) = \prod_{i=1}^{m} (h_\theta(V^{(i)}))^{y^{(i)}} (1 - h_\theta(V^{(i)}))^{1-y^{(i)}} \tag{4.2}$$

for m training samples and the cost function, based on this probabilistic approach, is as follows:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \big[-y^{(i)} log(h_\theta(V^{(i)})) - (1-y^{(i)}) log(1 - h_\theta(V^{(i)})) \big]. \tag{4.3}$$

The next step is the process of choosing θs for optimizing (i.e. minimizing) $J(\theta)$ and follows the Newton's update method:

$$\theta^{(s+1)} = \theta^{(s)} - H^{-1} \nabla_\theta J, \tag{4.4}$$

where the gradient is:

$$\nabla_\theta J = \frac{1}{m} \sum_{1}^{m} (h_\theta(V^{(i)}) - y^{(i)}) V^{(i)} \tag{4.5}$$

and the Hessian matrix is:

$$H = \frac{1}{m}\sum_{i=1}^{m}\left[h_\theta(V^{(i)})\left(1-h_\theta(V^{(i)})\right)V^{(i)}(V^{(i)})^T\right] \tag{4.6}$$

with $V^{(i)}(V^{(i)})^T \in \mathbb{R}^{(n+1)\times(n+1)}$ and $y^{(i)}$ are the labels calculated as the differences of the best[4] level's ask (and bid) prices. The suggested labels describe a binary classification problem: one for change in the best ask price and another one for no change in the best ask price. The above calculations were extracted in an online manner. The online process considers the 9^{th} element of every 10-MB block multiplied by the θ coefficient first-order tensor to obtain the probabilistic behavior (i.e., first-order tensor filtered through the hypothesis function) of the 10^{th} event of the 10-MB block. The output is the feature representation expressed as a scalar (i.e. probability) of the bid and ask price separately.

4.2.2 Selection Criteria

Entropy

Entropy [47], is a measure of signal complexity, wherein the signal is the time series of the multidimensional dual-mode tensor with dimensions $\mathbb{R}^{p\times n}$, p is the number of features and n number of samples, as a measure of feature relevance. The bits calculation for every feature in the feature set follows an iterative manner and reports the order. Entropy is measured as follows:

$$H(X) = -\sum_{i=1}^{p} p(x_i)\ log\ p(x_i), \tag{4.7}$$

where $p(x_i)$ is the probability of the frequency per feature for the given data samples.

Linear Discriminant Analysis

LDA [13] is used for classification and dimensionality reduction. However, instead of performing these tasks, LDA is converted into a feature selection algorithm. The feature selection performance is measured through two metrics. One is the classification rate (LDA1) and the other is based on the error term (LDA2), which is defined as the ratio of the within-class scatter matrix and the between-class scatter matrix. The

[4]Best LOB level is defined as the highest price for the bid size and the lowest price for the ask side.

main objective of LDA is to identify the projection matrix $\mathbf{W} \in \mathbb{R}^{m \times \#cl-1}$, where m is the sample dimension, and $\#cl$ is the number of classes, such that $Y = W^T X$ maximizes the class separation. For a given sample set X of m p-dimensional samples such that $X = \{X_1, X_2, ..., X_m\}$, where $X_i = (X_{i1}, X_{i2}, ..., X_{i\#cl})$ belonging to the i^{th} class, with $i = 1, 2, .., \#cl$. Then $\mathbf{W}$ maximizes Fisher's ratio

$$J(\mathbf{W}) = \frac{trace\left(\mathbf{W}^T S_B \mathbf{W}\right)}{trace\left(\mathbf{W}^T S_W \mathbf{W}\right)}, \tag{4.8}$$

where

$$S_B = \sum_{i=1}^{\#cl} N_i (\mu_i - \mu)(\mu_i - \mu)^T \tag{4.9}$$

and

$$S_W = \sum_{i=1}^{\#cl} \sum_{\mathbf{x} \in X_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^T \tag{4.10}$$

are the $between$-class and $within$-class scatter matrices, respectively, with

$$\mu_i = \frac{1}{N_i} \sum_{i \in \#cl} X_i \tag{4.11}$$

and

$$\mu = \frac{1}{N} \sum_{i}^{\#cl} N_i \mu_i, \tag{4.12}$$

for N_i number of samples in the $\#cl^{th}$ class. Same calculations are conducted for the projected samples $\mathbf{y}$ with

$$\widetilde{\mu_i} = \frac{1}{N_i} \sum_{i \in \#cl} Y_i \tag{4.13}$$

and

$$\widetilde{\mu} = \frac{1}{N} \sum_{i}^{\#cl} Y_i, \tag{4.14}$$

while the scatter matrices (i.e. $within$ and $between$ scatter matrices, respectively) are

$$\widetilde{S_W} = \sum_{i=1}^{\#cl} \sum_{y \in \#cl} (y - \tilde{\mu_i})(y - \tilde{\mu_i})^T \tag{4.15}$$

and

$$\widetilde{S_B} = \sum_{i=1}^{\#cl} N_i (\widetilde{\mu_i} - \widetilde{\mu})(\widetilde{\mu_i} - \widetilde{\mu})^T. \tag{4.16}$$

These two metrics constitute the evaluation of the hand-crafted features incrementally. The two evaluation metrics are based on the classification rate and the ratio of the $within$-class and $between$-class scatter matrices of the projected space Y.

Least-Mean-Square

LMS, a fitting method, produces an approximation that minimizes the sum of squared differences between the given data and the predicted values. LMS is used in a similar fashion (such as LDA) in the feature selection task. Two different approaches are presented, namely LMS1 (based on the classification rate) and LMS2 (based on the $\mathscr{L}_2$-norm). A hand-crafted feature assessment is performed sequentially based on LMS. More specifically, each of the features is evaluated according to a classification rate, the $\mathscr{L}_2$-norm of the predicted labels and the ground truth. The evaluation process is performed as follows: $\mathbf{HW} = \mathbf{T}$, where $\mathbf{H} \in \mathbb{R}^{p_i \times n}$ is the input data with feature dimension p_i where it is calculated incrementally for the number of training samples n, $\mathbf{W} \in \mathbb{R}^{p_i \times \#cl}$ are the weighted coefficients for the number of features p_i of the number(#) of classes (i.e. up, down, and stationary labelling), and $\mathbf{T} \in \mathbb{R}^{\#cl \times n}$ represents the target labels of the training set. The weight coefficient matrix $\mathbf{W}$ is estimated via the following formula: $\mathbf{W} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{H}^{\dagger}$ is the Moore-Penrose pseudoinverse matrix.

4.2.3 Feature Selection - Performance

Feature selection in terms of technical and quantitative indicators under the formation above provides useful information to traders and market makers. This is verified by the results of an extensive performance evaluation over the Nordic dataset. More specifically, the wrapper topology reported an average F1 score close to 50% with a best subset selection of close to five features (see Figure 4.1). Furthermore, the newly introduced feature based on the adaptive logistic regression model ranked first among the majority of the five selected sorting criteria (see Table 4.5).

Table 4.5 List of the first 10 best features for the 5 sorting methods ([P2])

Feature Sets	Description
Entropy	
1	Autocorrelation
2	Donchian Channels
3	Highest High
4	Center of Gravity Oscillator
5	Heikin-Ashi
6	Linear Regr. - Regression Coeffic.
7	Linear Regr. - Correlation Coeffic.
8	T3
9	TEMA
10	TRIMA
LMS1	
1	**Logistic Regr. - Local Spatial Ratio**
2	Best LOB Level - Bid Side Volume
3	Second Best LOB Level - Ask Volume
4	Price and Volume Derivation
5	Best LOB Level - Ask Side
6	Linear Regr. - Corr. Coeffic.
7	**Logistic Regr. - Logistic Coeffic.**
8	**Logistic Regr. - Extended Spatial Ratio**
9	Autocorrelation for Log Returns
10	Partial Autocorrelation
LMS2	
1	**Logistic Regression - Spatial Ratio**
2	Cointegration - Boolean Vector
3	Cointegration - Test Statistics
4	Price and Volume Means
5	Average Type Intensity
6	Average Type Intensity
7	Spread & Mid-Price
8	Alligator Jaw
9	Directional Index
10	Fractals
LDA1	
1	**Logistic Regression - Spatial Ratio**
2	Second Best LOB Level - Ask Volume
3	Price & Volume derivation
4	Spread & Mid-Price
5	Partial Autocorrelation for Log Returns
6	Linear Regression Line - Squared Corr. Coeffic.
7	Order Book Imbalance
8	Linear Regression - Corr. Coeffic.
9	Linear Regression - Regr. Coeffic.
10	Third Best LOB Level - Ask Volume
LDA2	
1	**Logistic Regression - Probability Estimation**
2	**Logistic Regression - Spatial Ratio**
3	Bollinger Bands
4	Alligator Teeth
5	Cointegration - Test Statistics
6	Best LOB Level - Bid Side Volume
7	Cointegration - p Values
8	Price & Volume Means
9	Price & Volume derivation
10	Price differences

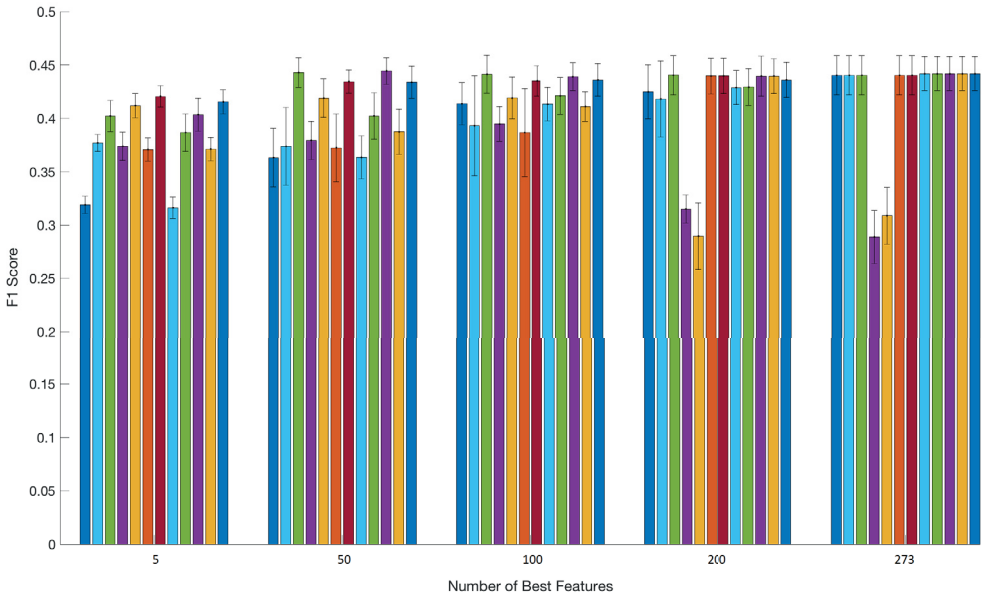


Figure 4.1 Bar plots with variance present the average (i.e. average F1 performance for the 9-fold protocol for all the features) F1 score of the 12 different models for the cases of 5, 50, 100, 200, and 273 number of best features. The order of the models from the left to the right column is (1) feature list sorted based on entropy and classified based on LMS, (2) feature list sorted based on LMS1 and classified based on LMS, (3) feature list sorted based on LMS2 and classified based on LMS, (4) feature list sorted based on LDA1 and classified based on LDA, (5) feature list sorted based on LDA2 and classified based on LDA, (6) feature list sorted based on LDA1 and classified based on LMS, (7) feature list sorted based on LDA2 and classified based on LMS, (8) feature list sorted based on LDA2 and classified based on LMS, (9) feature list sorted based on entropy and classified based on RBFN, (10) feature list sorted based on LMS2 and classified based on RBFN, (11) feature list sorted based on LDA1 and classified based on RBFN, and (12) feature list sorted based on LDA2 and classified based on RBFN ([P2]).

4.3 Econometric Features for Online Deep Learning

The introduction of econometric features as inputs to deep learning models (see Sections 2.3.3, 2.3.4, and 2.3.5) together with the development of an experimental protocol for online learning robust to time irregularities, which are present in HF data constitute another important contribution of the thesis (i.e., [P3]). Based on a fair evaluation of econometric features against fully automated (see Section 2.4.5) and LOB features, and against technical and quantitative indicators, the experimental results suggest that hand-crafted features provide good insight on when the mid-price

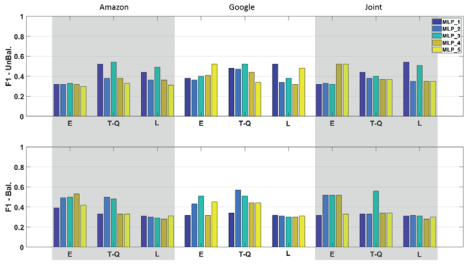
will change. Econometric features covered in this thesis contain information regarding statistical features, volatility measures (see Section 2.4.4), noise and uncertainty measures, and price discovery. A list of the econometric features against other handcrafted features can be seen in Table 4.6.

4.3.1 Econometric Features Performance

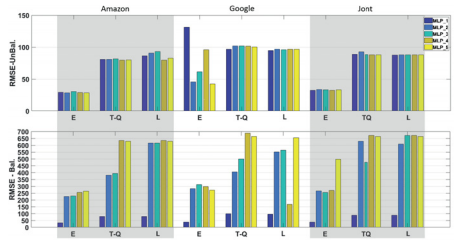
Econometric features were tested over liquid and illiquid stocks (i.e., based on US and Nordic stocks) via deep learning exploration. Several deep learning models are used for the task of the mid-price prediction. More specifically, five MLP, two CNN, and two LSTM neural networks were used in the experiments. The reported performance of these models was based on an extensive grid search where shallow and deep architectures were examined. Details of these models can be found in [P3]. The problem under consideration was the prediction of when and in which direction that mid-price movement will happen. As a result, the loss function is measured simultaneously through cross-entropy (classification task) and mean-square error (regression task). The results showed that econometrics features in some cases (i.e., selection of stock and model) lead to prediction of mid-price in the next millisecond. For instance, Figure 4.2 shows low RMSE of 28 events while F1 score is close to 60% for Amazon and Joint (Amazon and Google trained together) cases.

Table 4.6 Feature list of the three feature sets: Description for the newly introduced, Econometrics-based, handcrafted features can be found in [P3]; where description for the Tech & Quant and LOB feature sets can be found in [P2]

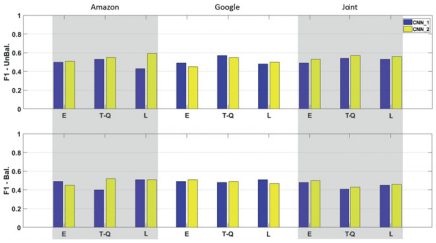
Econometric features	Tech & Quant features	LOB features
Statistical Features	**Technical Indicators**	**Basic**
Mid-Price	Accumulation Distribution Line	n LOB Levels
Financial Duration	Awesome Oscillator	
Average Mid-Price Financial Duration	Accelerator Oscillator	**Time-Insensitive**
Log-Returns	Average Directional Index	Spread & Mid-Price
	Average Directional Movement Index Rating	Price Differences
Volatility Measures	Displaced Moving Average	Price& Volume Means
Realized Volatility	Absolute Price Oscillator	Accumulated Differences
Realized Kernel	Aroon Indicator	
Realized Pre-Averaged Variance	Aroon Oscillator	**Time-Sensitive**
Realized Semi-Variance	Average True Range	Price & Volume Derivation
Realized Bipower Variation	Bollinger Bands	Average Intensity per Type
Realized Bipower Variation (lag 2)	Ichimoku Clouds	Relative Intensity Comparison
Realized Bipower Semi-Variance	Chande Momentum Oscillator	Limit Activity Acceleration
Jump Variation	Chaikin Oscillator	
Spot Volatility	Chandelier Exit	
Average Spot Volatility	Center of Gravity Oscillator	
	Donchian Channels	
Noise and Uncertainty Measures	Double Exponential Moving Average	
Realized Quarticity	Detrended Price Oscillator	
Realized Quarticity Tripower	Heikin-Ashi	
Realized Quarticity Quadpower	Highest High and Lowest Low	
Noise Variance [45]	Hull MA	
Noise Variance [60]	Internal Bar Strength	
	Keltner Channels	
Price Discovery Features	Moving Average Convergence/Divergence Oscillator	
Weighted Mid-Price by Order Imbalance	Median Price	
Volume Imbalance	Momentum	
Bid-Ask Spread	Variable Moving Average	
Normalized Bid-Ask Spread	Normalized Average True Range	
	Percentage Price Oscillator	
	Rate of Change	
	Relative Strength Index	
	Parabolic Stop and Reverse	
	Standard Deviation	
	Stochastic Relative Strength Index	
	T3-Triple Exponential Moving Average	
	Triple Exponential Moving Average	
	Triangular Moving Average	
	True Strength Index	
	Ultimate Oscillator	
	Weighted Close	
	Williams %R	
	Zero-Lag Exponential Moving Average	
	Fractals	
	Linear Regression Line	
	Digital Filtering: Rational Transfer Function	
	Digital Filtering: Savitzky-Golay Filter	
	Digital Filtering: Zero-Phase Filter	
	Remove Offset and Detrend	
	Beta-like Calculation	
	Quantitative Indicators	
	Autocorrelation	
	Partial Correlation	
	Cointegration based on Engle-Granger test	
	Order Book Imbalance	
	Logistic Regression for Online Learning	



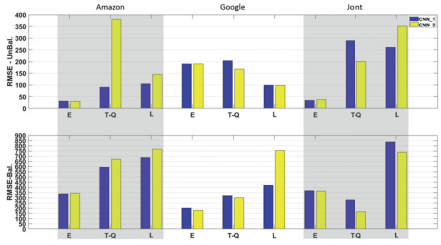
(a) F1 scores based on the unbalanced (top) and balanced (bottom) sets



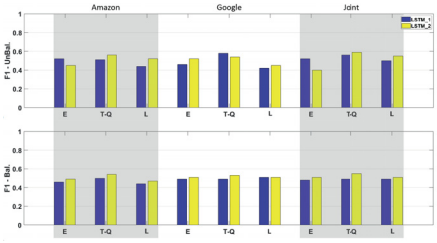
(b) RMSE scores based on the unbalanced (top) and balanced (bottom) sets



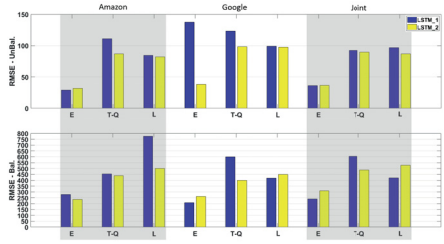
(c) F1 scores based on the unbalanced (top) and balanced (bottom) sets



(d) RMSE scores based on the unbalanced (top) and balanced (bottom) sets



(e) F1 scores based on the unbalanced (top) and balanced (bottom) sets



(f) RMSE scores based on the unbalanced (top) and balanced (bottom) sets

Figure 4.2 F1 (left column plots) and RMSE (right column plots) scores for the nine deep learning models based on the US data ([P3])

5 CONCLUSIONS

The main contribution of this thesis lies in answering the critical question of whether LOB mid-price movement is predictable by discovering and exploring the advantages of feature engineering in the era of machine and deep learning. A thorough investigation of different state-of-the-art features and experimental protocols together with the development of new and more advanced ones were conducted for the task of mid-price movement prediction. More specifically, the thesis is developed around three main pillars for the task above. The first pillar is related to the introduction of the first publicly available HF LOB dataset together with baseline models and state-of-the-art hand-crafted features. The second pillar is associated with an extended evaluation of the effectiveness of technical and advanced quantitative indicators for the same task under a wrapper method. Last, the third pillar is linked to a head-to-head comparison of hand-crafted econometric features against several state-of-the-art features via deep learning models. These three pillars are developed parallel to three publications that provide insight into the difficulties and solutions to the problem of LOB mid-price movement prediction.

LOB is the translation of daily trading activity to an organized system. This system requires an analysis of its dynamics. One of the main challenges researchers and practitioners face is accessing to publicly available LOB datasets. The present thesis introduces the first publicly available LOB dataset in the HF universe (see [P1]). This publicly available dataset comes with a baseline analysis that utilizes state-of-the-art LOB features and linear and non-linear classifiers (i.e., RR and SLFN), which were tuned and extensively tested over HF data for the first time. Even though results highlighted the impact that hand-crafted features have in predicting mid-price movements, the dataset is limited to 144 hand-crafted LOB features. This dataset under the anchored cross-validation protocol invites further probe into investigation of more advanced methods and state-of-the-art features. The performance of the F1 score can be improved by considering more advanced hand-crafted features and validate their

effectiveness in line with the best feature subset selection methods.

These drawbacks are addressed in [P2], where an extensive list of technical indicators together with quantitative and LOB features were tested via a wrapper method based on five different sorting metrics. This list is the most extensive one in the literature and contains the majority of technical indicators and some advanced quantitative features. Furthermore, a new feature was proposed, which is based on an adaptive logistic regression model suitable for online learning. The new proposed feature was selected first among the majority of the sorting metrics. Another contribution of the thesis is the conversion of entropy, LDA, and LMS as selection criteria for the best-provided feature list. Despite the extensive evaluation protocol, the proposed hand-crafted feature lists were tested only on five Nordic stocks which, have lower intraday activity compared, for instance, to US stocks. Another improvement would have been the development of an experimental protocol that considers every trading event (i.e., online learning) and not only features extracted from independent event blocks (i.e., features are extracted every ten independent events). This process creates a lag of ten events in the prediction mechanism.

The proposed analysis presented in [P3] addresses these issues by introducing a new dataset based on two liquid US stocks (i.e., Amazon and Google) and a new experimental protocol suitable for online learning. These two additions, together with another set of econometrics-based hand-crafted features, constitue the most significant contribution of this publication. Econometrics were tested extensively, through deep learning classifiers and regressors, against three other features sets introduced in [P2] and features extracted by means of fully automated processes derived from LSTM AE. The results suggest that the use of advanced hand-crafted features can outperform fully automated processes and basic feature engineering. [P3] opens avenues for more advanced feature development. For instance, more advanced fully automated feature extraction methods should be tested for the same task. An interesting addition to the existing protocol would be the utilization of a more extensive number of stocks.

In conclusion, the contributions presented in this thesis offer a new approach to feature engineering universe under the scope of machine and deep learning for the critical task of LOB mid-price movement prediction. This thesis showed that advanced features, methods, and proper experimental protocol development could effectively provide insight into future trends of LOB metrics. Several questions arise

and should be addressed in the future. A feature selection mechanism for the five feature lists presented here should be utilized in order to shed light on which ones convey more information for the task under consideration. The problem of mid-price prediction can also be treated as an image segmentation problem. More specifically, time series can be converted to images and then fed into state-of-the-art classifiers found in image recognition literature.

REFERENCES

[1] A. N. Akansu, S. R. Kulkarni and D. M. Malioutov. *Financial Signal Processing and Machine Learning*. John Wiley & Sons, 2016.

[2] A. N. Akansu and M. U. Torun. *A Primer for Financial Engineering: Financial Signal Processing and Electronic Trading*. Academic Press, 2015.

[3] I. Aldridge. *High-frequency trading: a practical guide to algorithmic strategies and trading systems*. Vol. 604. John Wiley & Sons, 2013.

[4] T. G. Andersen and T. Bollerslev. Answering the skeptics: Yes, standard volatility models do provide accurate forecasts. *International Economic Review* (1998), 885–905.

[5] P. Bak, M. Paczuski and M. Shubik. Price variations in a stock market with many agents. *Physica A: Statistical Mechanics and its Applications* 246.3-4 (1997), 430–453.

[6] G. T. Barkema, M. J. Howard and J. L. Cardy. Reaction-diffusion front for $A + B \to \emptyset$ in one dimension. *Physical Review E* 53.3 (1996), R2017.

[7] O. E. Barndorff-Nielsen et al. Designing realized kernels to measure the ex post variation of equity prices in the presence of noise. *Econometrica* 76.6 (2008), 1481–1536.

[8] O. E. Barndorff-Nielsen et al. Realized kernels in practice: Trades and quotes. *The Econometrics Journal* 12.3 (2009), C1–C32.

[9] B. Biais, P. Hillion and C. Spatt. An empirical analysis of the limit order book and the order flow in the Paris Bourse. *Journal of Finance* 50.5 (1995), 1655–1689.

[10] C. M. Bishop et al. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[11] G. E. Box et al. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

[12] K. Christensen, R. C. Oomen and M. Podolskij. Fact or friction: Jumps at ultra high frequency. *Journal of Financial Economics* 114.3 (2014), 576–599. ISSN: 0304-405X.

[13] P. Cohen, S. G. West and L. S. Aiken. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Psychology Press, 2014.

[14] R. Cont and A. De Larrard. Order book dynamics in liquid markets: limit theorems and diffusion approximations (2012). *arXiv preprint arXiv:1202.6412* ().

[15] R. Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance* 1 (2001), 223–236.

[16] R. Cont and A. De Larrard. Order book dynamics in liquid markets: limit theorems and diffusion approximations. *Available at SSRN 1757861* (2012).

[17] R. Cont and A. De Larrard. Price dynamics in a Markovian limit order market. *SIAM Journal on Financial Mathematics* 4.1 (2013), 1–25.

[18] R. Cont, A. Kukanov and S. Stoikov. The price impact of order book events. *Journal of Financial Econometrics* 12.1 (2014), 47–88.

[19] R. Cont, S. Stoikov and R. Talreja. A stochastic model for order book dynamics. *Operations research* 58.3 (2010), 549–563.

[20] M. G. Daniels, J. D. Farmer, L. Gillemot, G. Iori and E. Smith. Quantitative model of price diffusion and market friction based on trading as a mechanistic random process. *Physical Review Letters* 90.10 (2003), 108102.

[21] A. Ellul et al. *Determinants of Order Choice on the New York Stock Exchange*. 2003. URL: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.114.6359`.

[22] R. F. Engle and C. W. Granger. Dynamic model specification with equilibrium constraints: Co-integration and error-correction. *Econometrica* 55.3 (1987), 251–276.

[23] G. Eshel. The Yule Walker equations for the AR coefficients. *Internet resource* 2 (2003), 68–73.

[24] E. Fama. *Foundations of Finance: Portfolio Decisions and Securities Prices*. Basic Books, 1976.

[25] S. Fernández, A. Graves and J. Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. *International Conference on Artificial Neural Networks*. Springer. 2007, 220–229.

[26] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* 36.4 (1980), 193–202.

[27] R. Gallager. *Discrete Stochastic Processes. RES.6-262*. Massachusetts Institute of Technology: MIT OpenCourseWare. Spring 2011. URL: `https://ocw.mit.edu`. License: Creative Commons BY-NC-SA.

[28] R. Gençay et al. *An introduction to High-Frequency Finance*. Elsevier, 2001.

[29] I. Goodfellow, Y. Bengio and A. Courville. *Deep Learning*. MIT Press, 2016.

[30] M. D. Gould et al. Limit order books. *Quantitative Finance* 13.11 (2013), 1709–1742.

[31] A. D. Hall and N. Hautsch. Order aggressiveness and order book dynamics. *High Frequency Financial Econometrics*. Springer, 2008, 133–165.

[32] J. D. Hamilton. Time series analysis. *Economic Theory. II, Princeton University Press, USA* (1995), 625–630.

[33] Y. Han and D. Lesmond. Liquidity biases and the pricing of cross-sectional idiosyncratic volatility. *The Review of Financial Studies* 24.5 (2011), 1590–1629.

[34] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation* 9.8 (1997), 1735–1780.

[35] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew. Extreme learning machine: theory and applications. *Neurocomputing* 70.1-3 (2006), 489–501.

[36] A. Iosifidis, A. Tefas and I. Pitas. Approximate kernel extreme learning machine for large scale data classification. *Neurocomputing* 219 (2017), 210–220.

[37] J. Jacod et al. Microstructure noise in the continuous case: the pre-averaging approach. *Stochastic Processes and their Applications* 119.7 (2009), 2249–2276.

[38] A. N. Kercheval and Y. Zhang. Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance* 15.8 (2015), 1315–1329.

[39] A. S. Kyle. Continuous auctions and insider trading. *Econometrica: Journal of the Econometric Society* (1985), 1315–1335.

[40] Y. LeCun et al. Object recognition with gradient-based learning. *Shape, Contour and Grouping in Computer Vision*. Springer, 1999, 319–345.

[41] C.-Y. Liou et al. Autoencoder for words. *Neurocomputing* 139 (2014), 84–96.

[42] A. Ntakaris, J. Kanniainen, M. Gabbouj and A. Iosifidis. Mid-Price Prediction Based on Machine Learning Methods with Technical and Quantitative Indicators. *PLOS ONE* (under review).

[43] A. Ntakaris, M. Magris, J. Kanniainen, M. Gabbouj and A. Iosifidis. Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting* 37.8 (2018), 852–866.

[44] A. Ntakaris, G. Mirone, J. Kanniainen, M. Gabbouj and A. Iosifidis. Feature Engineering for Mid-Price Prediction With Deep Learning. *IEEE Access* 7 (2019), 82390–82412.

[45] R. C. A. Oomen. Properties of Realized Variance under Alternative Sampling Schemes. *Journal of Business & Economic Statistics* 24.2 (2006), 219–237. ISSN: 07350015. URL: `http://www.jstor.org/stable/27638871`.

[46] R. H. I. Palgrave. *The New Palgrave: A Dictionary of Economics*. Vol. 1. Macmillan, 1987.

[47] J. S. Richman and J. R. Moorman. Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology* 278.6 (2000), H2039–H2049.

[48] F. Rosenblatt. *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms*. Tech. rep. Cornell Aeronautical Lab Inc, Buffalo, NY, 1961.

[49] W. A. Rosenkrantz. Why Stock prices Have a log-normal Distribution. *Department of Mathematics and Statistics, University of Massachusetts at Amhers* (2003).

[50] D. E. Rumelhart et al. Learning representations by back-propagating errors. *Cognitive modeling* 5.3 (1988), 1.

[51] P. A. Samuelson and W. D. Nordhaus. Economics, chapter 18. *Protecting the Environment* (2004).

[52] J. A. Sirignano. Deep learning for limit order books. *Quantitative Finance* 19.4 (2019), 549–570.

[53] E. Smith et al. Statistical theory of the continuous double auction. *Quantitative finance* 3.6 (2003), 481–514.

[54] A. N. Tikhonov. On the stability of inverse problems. *Dokl. Akad. Nauk SSSR*. Vol. 39. 1943, 195–198.

[55] P. D. Wasserman and T. Schwartz. Neural networks. II. What are they and why is everybody so interested in them now?: *IEEE Expert* 3.1 (1988), 10–15.

[56] J. W. Wilder Jr. The relative strength index. *Journal of Technical Analysis of Stocks and Commodities* 4 (1986), 343–346.

[57] J. W. Wilder. *New Concepts in Technical Trading Systems*. Trend Research, 1978.

[58] L. Williams. The ultimate oscillator. *Technical Analysis of Stocks and Commodities* 3.4 (1985), 140–141.

[59] J. M. Wooldridge. *Introductory Econometrics: A modern Approach*. Nelson Education, 2015.

[60] L. Zhang, P. A. Mykland and Y. Aït-Sahalia. A tale of two time scales: Determining integrated volatility with noisy high-frequency data. *Journal of the American Statistical Association* 100.472 (2005), 1394–1411.

[61] L. Zhao. A model of limit-order book dynamics and a consistent estimation procedure. PhD thesis. Carnegie Mellon University, 2010.

PUBLICATION 1

Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods

A. Ntakaris, M. Magris, J. Kanniainen, M. Gabbouj and A. Iosifidis

Journal of Forecasting 37.8 (2018), 852–866

Received: 15 May 2018 | Accepted: 7 July 2018
DOI: 10.1002/for.2543

RESEARCH ARTICLE

WILEY

Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods

Adamantios Ntakaris[1] | Martin Magris[2] | Juho Kanniainen[2] | Moncef Gabbouj[1] | Alexandros Iosifidis[3]

[1]Laboratory of Signal Processing, Tampere University of Technology, Tampere, Finland

[2]Laboratory of Industrial and Information Management, Tampere University of Technology, Tampere, Finland

[3]Department of Engineering, Electrical and Computer Engineering, Aarhus University, Aarhus, Denmark

Correspondence
Adamantios Ntakaris, Laboratory of Signal Processing, Tampere University of Technology, Korkeakoulunkatu 1, Tampere, Finland.
Email: adamantios.ntakaris@tut.fi

Funding information
H2020 Marie Sklodowska-Curie Actions, Grant/Award Number: MSCA-ITN-ETN 675044

Abstract

Managing the prediction of metrics in high-frequency financial markets is a challenging task. An efficient way is by monitoring the dynamics of a limit order book to identify the information edge. This paper describes the first publicly available benchmark dataset of high-frequency limit order markets for mid-price prediction. We extracted normalized data representations of time series data for five stocks from the Nasdaq Nordic stock market for a time period of 10 consecutive days, leading to a dataset of ∼4,000,000 time series samples in total. A day-based anchored cross-validation experimental protocol is also provided that can be used as a benchmark for comparing the performance of state-of-the-art methodologies. Performance of baseline approaches are also provided to facilitate experimental comparisons. We expect that such a large-scale dataset can serve as a testbed for devising novel solutions of expert systems for high-frequency limit order book data analysis.

KEYWORDS

high-frequency trading, limit order book, mid-price, machine learning, ridge regression, single hidden feedforward neural network

1 | INTRODUCTION

Automated trading became a reality when the majority of exchanges adopted it globally. This environment is ideal for high-frequency traders. High-frequency trading (HFT) and a centralized matching engine, referred to as a limit order book (LOB), are the main drivers for generating big data (Seddon & Currie, 2017). In this paper, we describe a new order book dataset consisting of approximately 4 million events for 10 consecutive trading days for five stocks. The data are derived from the ITCH feed provided by Nasdaq OMX Nordic and consists of the time-ordered sequences of messages that track and record all the events occurring in the specific market. It provides a complete market-wide history of 10 trading days. Additionally, we define an experimental protocol to evaluate the performance of research methods in mid-price prediction.[1]

Datasets, like the one presented here, come with challenges, including the selection of appropriate data transformation, normalization, description, and classification. This type of massive dataset requires a very good understanding of the available information that can be extracted

[1]Mid-price is the average of the best bid and best ask prices.

for further processing. We follow the information edge, as has been recently presented by Kercheval and Zhang (2015). The authors provide a detailed description of representations that can be used for a mid-price movement prediction metric. In light of this data representation, they apply nonlinear classification based on support vector machines (SVM) in order to predict the movement of this metric. Such a supervised learning model exploits class labels[2] for short- and long-term prediction. However, they train their model based on a very small (when compared to the size of the data that can be available for such applications) dataset of 4,000 samples. This is due to the limitations of many nonlinear kernel-based classification models related to their time and space complexity with respect to the training data size. On the other hand, Sirignano (2016) uses large amounts of data for nonlinear classification based on a feedforward network. The author takes advantage of the local spatial structure[3] of the data for modeling the joint distribution of the LOB's state based on its current state.

Despite the major importance of publicly available datasets for advancing research in the HFT field, there are no detailed public available benchmark datasets for method evaluation purposes. In this paper, we describe the first publicly available dataset[4] for an LOB-based HFT that has been collected in the hope of facilitating future research in the field. Based on Kercheval and Zhang (2015), we provide time series representations of approximately 4,000,000 trading events and annotations for five classification problems. Baseline results of two widely used methods—that is, linear and nonlinear regression models, are also provided. In this way, we introduce this new problem for the expert systems community and provide a testbed for facilitating future research. We hope that attracting the interest of expert systems will lead to the rapid improvement of the performance achieved in the provided dataset, thus leading to much better state-of-the-art solutions to this important problem.

The dataset described in this paper can be useful for financial expert systems in two ways. First, it can be used to identify circumstances under which markets are stable, which is very important for liquidity providers (market makers) to make the spread. Consequently, such an intelligent system would be valuable as a framework that can increase liquidity provision. Secondly, analysis of the data can be used for model selection by speculative traders, who are trading based on their predictions on market movements. In future research, this paper can be employed to identify order book spoofing—that is, situations where markets are exposed to manipulation by limit orders. In this case, spoofers could aim to move markets in certain directions by limit orders that are canceled before they are filled. Therefore, this research is relevant not only for market makers and traders but also for supervisors and regulators.

Therefore, the present work makes the following contributions: (1) To the best of our knowledge this is the first publicly available LOB-ITCH dataset for machine learning experiments on the prediction of mid-price movements. (2) We provide baselines methods based on ridge regression and a new implementation of an RBF neural network based on *k*-means algorithm. (3) The paper provides information about the prediction of mid-price movements to market makers, traders, and regulators. This paper does not suggest any trading strategies and is reliant on purely machine learning metrics prediction. Overall, this work is an empirical exploration of the challenges that come with high-frequency trading and machine learning applications.

The data from Nasdanq Helsinki Stock Exchange offers important benefits. In the USA the limit orders for a given asset are spread between several exchanges, causing fragmentation of liquidity. The fragmentation poses a problem for empirical research, because, as Gould, Porter, Williams, McDonald, Fenn, and Howison (2013) point out, the "differences between different trading platforms' matching rules and transaction costs complicate comparisons between different limit order books for the same asset." These issues related to fragmentation are not present with data obtained from less fragmented Nasdaq Nordic markets. Moreover, Helsinki Exchange is a pure limit order market, where the market makers have a limited role.

The rest of the paper is organized as follows. We provide a comprehensive literature review of the field in Section 2. Dataset and experimental protocol descriptions are provided in Section 33. Quantitative and qualitative comparisons of the new dataset, along with related data sources, are provided in Section 4. In Section 5, we describe the engineering of our baselines. Section 6 presents our empirical results and Section 7 concludes.

2 | MACHINE LEARNING FOR HFT AND LOB

The complex nature of HFT and LOB spaces is suitable for interdisciplinary research. In this section, we provide a comprehensive review of recent methods exploiting

[2]Labels are extracted from annotations provided by experts and represent the direction of the mid-price. Three different states are defined—that is, upward, downward, and stationary movement.

[3]By local movement, the author means that the conditional movement of the future price (e.g., best ask price movement) depends, locally, on the current LOB state.

[4]The dataset can be downloaded from: https://etsin.avointiede.fi/dataset/urn-nbn-fi-csc-kata20170601153214969115https://etsin.avointiede.fi/dataset/urn-nbn-fi-csc-kata20170601153214969115.

machine learning approaches. Regression models, neural networks, and several other methods have been proposed to make inferences of the stock market. Existing literature ranges from metric prediction to optimal trading strategies identification. The research community has tried to tackle the challenges of prediction and data inference from different angles. Although mid-price prediction can be considered a traditional time series prediction problem, there are several challenges that justify HFT as a unique problem.

2.1 | Regression analysis

Regression models have been widely used for HFT and LOB prediction. Zheng, Moulines, and Abergel (2012) utilize logistic regression in order to predict the inter-trade price jump. Alvim, dos Santos, and Milidiu (2010) use support vector regression (SVR) and partial least squares (PLS) for trading volume forecasting for 10 Bovespa stocks. Pai and Lin (2005) use a hybrid model for stock price prediction. They combine an autoregressive integrated moving average (ARIMA) model and an SVM classifier in order to model nonlinearities of class structure in regression estimation models. Liu and Park (2015) develop a multivariate linear model to explain short-term stock price movement where a bid–ask spread is used for classification purposes. Detollenaere and D'hondt (2017) apply an adaptive least absolute shrinkage and selection operator (LASSO)[5] for variable selection, which best explains the transaction cost of the split order. They apply an adjusted ordinal logistic method for classifying ex ante transaction costs into groups. Cenesizoglu, Dionne, and Zhou (2014) work on a similar problem. They hold that the state of the limit order can be informative for the direction of future prices and try to prove their position by using an autoregressive model.

Panayi, Peters, Danielsson, and Zigrand (2016) use generalized linear models (GLM) and generalized additive models for location, shape, and scale (GAMLSS) models in order to relate the threshold exceedance duration (TED), which measures the length of time required for liquidity replenishment, to the state of the LOB. Yu (2006) tries to extract information from order information and order submission based on the ordered probit model.[6] The author shows, in the case of Shanghai's stock market, that an LOB's information is affected by the trader's strategy, with different impacts on the bid and ask sides. Amaya, Filbien, Okou, and Roch (2015) use panel regression[7] for order imbalances and liquidity costs in LOBs so as to identify resilience in the market. Their findings show that such order imbalances cause liquidity issues that last for up to 10 minutes. Malik and Lon Ng (2014) analyze the asymmetric intra-day patterns of LOBs. They apply regression with a power transformation on the notional volume weighted average price (NVWAP) curves in order to conclude that both sides of the market behave asymmetrically to market conditions.[8] In the same direction, Ranaldo (2004) examines the relationship between trading activity and the order flow dynamics in LOBs, where the empirical investigation is based on a probit model. Cao, Hansch, and Wang (2009) examine the depth of different levels of an order book by using an autoregressive (AR) model of order 5 (the AR(5) framework). They find that levels beyond the best bid and best ask prices provide moderate information regarding the true value of an asset. Finally, Creamer (2012) suggests that the LogitBoost algorithm is ideal for selecting the right combination of technical indicators.[9]

2.2 | Neural networks

HFT is mainly a scalping[10] strategy according to which the chaotic nature of the data creates the proper framework for the application of neural networks. Levendovszky and Kia (2012) propose a multilayer feedforward neural network for predicting the price of a EUR/USD pair, trained by using the backpropagation algorithm. Sirignano (2016) proposes a new method for training deep neural networks that try to model the joint distribution of the bid and ask depth, where a focal point is the spatial nature[11] of LOB levels. Bogoev and Karam (2016) propose the use of a single hidden-layer feedforward neural (SLFN) network for the detection of quote stuffing and momentum ignition. Dixon (2016) uses a recurrent neural network (RNN) for mid-price predictions of T-bond[12] and ES futures[13] based on ultra-high-frequency data. Rehman, Khan, and

[5]Adaptive weights are used for penalizing different coefficients in the l_1 penalty term.
[6]The method is the generalization of a linear regression model when the dependent variable is discrete.
[7]Panel regression models provide information on data characteristics individually, but also across both individuals over time.
[8]Market conditions of an industry sector have an impact on sellers and buyers who are related to it. Factors to consider include the number of competitors in the sector. For example, if there is a surplus, new companies may find it difficult to enter the market and remain in business.
[9]Technical indicators are mainly used for short-term price movement predictions. They are formulas based on historical data.
[10]Scalping is a type of trading strategy according to which the trader tries to make a profit for small changes in a stock.
[11]The spatial nature of this type of neural network and its gradient can be evaluated at far fewer grid points. This makes the model less computationally expensive. Furthermore, the suggested architecture can model the entire distribution in the R^d space.
[12]Treasury bond (T-bond) is a long-term fixed interest rate debt security issued by the federal government.
[13]E-mini S&P 500 (ES futures) are electronically traded futures contracts whose value is one-fifth the size of standard S&P futures.

Mahmud (2014) apply recurrent Cartesian genetic programming evolved artificial neural network (RCGPANN) for predicting five currency rates against the Australian dollar. Galeshchuk (2016) suggests that a multilayer perceptron (MLP) architecture, with three hidden layers, is suitable for exchange rate prediction. Majhi, Panda, and Sahoo (2009) use the functional link artificial neural network (FLANN) in order to predict price movements in the DJIA[14] and S&P 500[15] stock indices.

Deep belief networks are employed by Sharang and Rao (2015) to design a medium-frequency portfolio trading strategy. Hallgren and Koski (2016) use continuous-time Bayesian networks (CTBNs) for causality detection. They apply their model on tick-by-tick high-frequency foreign exchange (FX) EUR/USD data using a Skellam process.[16] Sandoval and Hernández (2015) create a profitable trading strategy by combining hierarchical hidden Markov models (HHMM), where they consider wavelet-based LOB information filtering. In their work, they also consider a two-layer feedforward neural network in order to classify the upcoming states. They nevertheless report limitations in the neural network in terms of the volume of the input data.

2.3 | Maximum margin and reinforcement learning

Palguna and Pollak (2016) use nonparametric methods on features derived from LOB, which are incorporated into order execution strategies for mid-price prediction. In the same direction, Kercheval and Zhang (2015) employ a multi-class SVM for mid-price and price spread crossing prediction. Han et al. (2015) base their research on Kercheval and Zhang by using multi-class SVM for mid-price movement prediction. More precisely, they compare multi-class SVM (exploring linear and RBF kernels) to decision trees using bagging for variance reduction.

Kim (2001) uses input/output hidden Markov models (IOHMMs) and reinforcement learning (RL) in order to identify the order flow distribution and market-making strategies, respectively. Yang et al. (2015) apply apprenticeship learning[17] methods, like linear inverse reinforcement learning (LIRL) and Gaussian process IRL (GPIRL), to recognize traders or algorithmic trades based on the observed limit orders. Chan and Shelton (2001) use RL for market-making strategies, where experiments based on a Monte Carlo simulation and a state–action–reward–state–action (SARSA) algorithm test the efficacy of their policy. In the same vein, Kearns and Nevmyvaka (2013) implement RL for trade execution optimization in lit and dark pools. Especially in the case of dark pools, they apply a censored exploration algorithm to the problem of smart order routing (SOR). Yang, Paddrik, Hayes, Todd, Kirilenko, Beling, and Scherer (2012) examine an IRL algorithm for the separation of HFT strategies from other algorithmic trading activities. They also apply the same algorithm to the identification of manipulative HFT strategies (i.e., spoofing). Felker, Mazalov, and Watt (2014) predict changes in the price of quotes from several exchanges. They apply feature-weighted Euclidean distance to the centroid of a training cluster. They calculate this type of distance to the centroid of a training cluster where feature selection is taken into consideration because several exchanges are included in their model.

2.4 | Additional methods for HFT and LOB

HFT and LOB research activity also covers topics like the optimal submission strategies of bid and ask orders, with a focus on the inventory risk that stems from an asset's value uncertainty, as in the work of Avellaneda and Stoikov (2008). Chang (2015) models the dynamics of LOB by using a Bayesian inference of the Markov chain model class, tested on high-frequency data. An and Chan (2017) suggest a new stochastic model that is based on independent compound Poisson processes of the order flow. Talebi, Hoang, and Gavrilova (2014) try to predict trends in the FX market by employing a multivariate Gaussian classifier (MGC) combined with Bayesian voting. Fletcher, Hussain, and Shawe-Taylor (2010) examine trading opportunities for the EUR/USD where the price movement is based on multiple kernel learning (MKL). More specifically, the authors utilize SimpleMKL and the more recent LPBoost-MKL methods for training a multi-class SVM. Christensen and Woodmansey (2013) develop a classification method based on the Gaussian kernel in order to identify iceberg[18] orders for GLOBEX.

Maglaras, Moallemi, and Zheng (2015) consider the LOB as a multi-class queueing system in order to solve the problem placement of limit and market order placements. Mankad, Michailidis, and Kirilenko (2013) apply a static plaid clustering technique to synthetic data in order to

[14] The Dow Jones Industrial Average (DJIA) is the price-weighted average of the 30 largest, publicly owned US companies.

[15] S&P 500 is the index that provides a summary of the overall market by tracking some of the 500 top stocks in US stock market.

[16] A Skellam process is defined as $S(t) = N^{(1)}(t) - N^{(2)}(t), \quad t \geq 0$, where $N^{(1)}(t)$ and $N^{(2)}(t)$ are two independent homogeneous Poisson processes.

[17] Motivation for apprenticeship learning is to use IRL techniques to learn the reward function and then use this function in order to define a Markov decision problem (MDP).

[18] Iceberg order is the conditional request made to the broker to sell or buy a larger quantity of the stock, but in smaller predefined quantities.

classify the different types of trades. Aramonte, Schindler, and Rosen (2013) show that the information asymmetry in a high-frequency environment is crucial.

Vella and Ng (2016) use higher-order fuzzy systems (i.e., an adaptive neuro-fuzzy inference system) by introducing T2 fuzzy sets, where the goal is to reduce microstructure noise in the HFT sphere. Abernethy and Kale (2013) apply market-maker strategies based on low-regr et algorithms for the stock market. Almgren and Lorenz (2006) explain price momentum by modeling Brownian motion with a drift whose distribution is updated based on Bayesian inference. Næs and Skjeltorp (2006) show that the order book slope measures the elasticity of supplied quantity as a function of asset prices related to volatility, trading activity, and an asset's dispersion beliefs.

3 | THE LOB DATASET

In this section, we describe in detail our dataset collected in order to facilitate future research in LOB-based HFT. We start by providing a detailed description of the data in Section 3.1. Data processing steps are followed in order to extract message books and LOBs, as described in Section 3.2.

3.1 | Data description

Extracting information from the ITCH flow, and without relying on third-party data providers, we analyze stocks from different industry sectors for 10 full days of ultra-high-frequency intra-day data. The data provide information regarding trades against hidden orders. Coherently, the nondisplayable hidden portions of the total volume of a so-called iceberg order are not accessible from the data. Our ITCH feed data is day specific and market wide, which means that we deal with one file per day with data over all the securities. Information (block A in Figure 1) regarding (i) messages for order submissions, (ii) trades, and (iii) cancellations is included. For each order, its type (buy/sell), price, quantity, and exact time stamp on a millisecond basis is available. In addition, (iv) administrative messages (i.e., trading halts or basic security data), (v) event controls (i.e., start and ending of trading days, states of market segments), and (vi) net order imbalance indicators are also included.

The next step is the development and implementation of a C++ converter to extract all the information relevant to a given security. We perform the same process for five stocks traded on the Nasdaq OMX Nordic at the Helsinki exchange from June 1, 2010 to June 14, 2010.[19] These data are stored in a Linux cluster. Information related to the five stocks is illustrated in Table 1. The selected stocks[20] are traded in one exchange (Helsinki) only. By choosing only one stock market exchange, the trader has the advantage of avoiding issues associated with fragmented markets. In the case of fragmented markets, the limit orders for

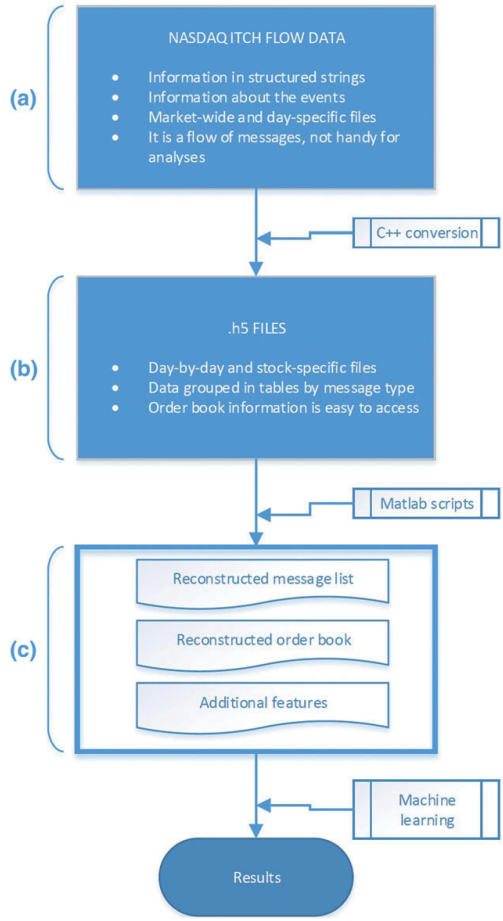


FIGURE 1 Data processing flow [Colour figure can be viewed at wileyonlinelibrary.com]

[19]There have been about 23,000 active order books, the vast majority of which are very illiquid, show sporadic activity, and correspond to little and noisy data.

[20]The choice is driven by the necessity of having a sufficient amount of data for training (this excludes illiquid stocks) while covering different industry sectors. These five selected stocks (see Table 1), which aggregate input message list and order book data for feature extraction, are about 4 GB; RTRKS was suspended from trading and delisted from the Helsinki exchange on November 20, 2014.

a given asset are spread between several exchanges, posing problems from empirical data analysis (O'Hara & Ye, 2011).

The Helsinki Stock Exchange, operated by Nasdaq Nordic, is a pure electronic limit order market. The ITCH feed keeps a record of all the events, including those that take place outside active trading hours. At the Helsinki exchange, the trading period goes from 10:00 to 18:25 (local time, UTC/GMT +2 hours). However, in the ITCH feed, we observe several records outside those trading hours. In particular, we consider the regulated auction period before 10:00, which is used to set the opening price of the day (the so-called pre-opening period) before trading begins. This is a structurally different mechanism following different rules with respect to the order book flow during trading hours. Similarly, another structural break in the order book's dynamics is due to the different regulations that are in force between 18:25 and 18:30 (the so-called post-opening period). As a result, we retain exclusively the events occurring between 10:30 and 18:00. More information related to the above-mentioned issues can be found in Siikanen, Kanniainen, and Luoma 2017 and (Siikanen, Kanniainen, & Valli, 2017). Here, the order book is expected to have comparable dynamics with no biases or exceptions caused by its proximity to the market opening and closing times.

3.2 | Limit order and message books

Message and LOBs are processed for each of the 10 days for the five stocks. More specifically, there are two types of messages that are particularly relevant here: (i) "add order messages," corresponding to order submissions; and (ii) "modify order messages," corresponding to updates on the status of existing orders through order cancellations and order executions. Example message[21] and limit order[22] books are illustrated in Tables 2 and Table 3, respectively.

LOB is a centralized trading method that is incorporated by the majority of exchanges globally. It aggregates the limit orders of both sides (i.e., the ask and bid sides) of the stock market (e.g., the Nordic stock market). LOB matches every new event type according to several characteristics. Event types and LOB characteristics describe the current state of this matching engine. Event types can be executions, order submissions, and order cancellations. Characteristics of LOB are the resolution parameters (Gould, Porter, Williams, McDonald, Fenn, & Howison, 2013), which are the tick size π (i.e., the smallest permissible price between different orders), and the lot size σ (i.e., the smallest amount of a stock that can be traded and is defined as $\{k\sigma | k = 1, 2, \dots\}$). Order inflow and resolution parameters will formulate the dynamics of the LOB, whose current state will be identified by the state variable of four elements $(s_t^b, q_t^b, s_t^a, q_t^a), t \geq 0$, where s_t^b (s_t^b) is the best bid (ask) price and q_t^b (q_t^a) is the size of the best bid (ask) level at time t.

In our data, timestamps are expressed in milliseconds based on 1 Jan 1970 format and shifted by three hours with respect to Eastern European Time (in the data, the trading day goes from 7:00 to 15:25). ITHC feed prices are recorded up to 4 decimal places and, in our data, the decimal point is removed by multiplying the price by 10,000, where currency is in euros for the Helsinki exchange. The tick size, defined as the smallest possible gap between the ask and bid prices, is 1 cent. Similarly, order quantities are constrained to integers greater than one.

3.3 | Data availability and distribution

In compliance with Nasdaq OMX agreements, the normalized feature dataset is made available to the research community.[23] The open-access version of our data has been normalized in order to prevent reconstruction of the original Nasdaq data.

3.4 | Experimental protocol

In order to make our dataset a benchmark that can be used for the evaluation of HTF methods based on LOB information, the data are accompanied by the following experimental protocol. We develop a day-based prediction framework following an anchored forward cross-validation format. More specifically, the training set is increased by 1 day in each fold and stops after $n - 1$ days (i.e., after 9 days in our case where $n = 10$). On each fold, the test set corresponds to 1 day of data, which moves in a rolling window format. The experimental setup is illustrated in Figure 2. Performance is measured by calculating the mean accuracy, recall, precision, and F1 score over all folds, as well as the corresponding standard deviation. We measure our results based on these metrics, which are defined as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad (1)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad (2)$$

[21] A sample from FI0009002422 on June 1, 2010.
[22] A sample from FI0009002422 on June 1, 2010.

[23] We thank Ms. Sonja Salminen at Nasdaq for her support and help.

TABLE 1 Stocks used in the analysis

ID	ISIN code	Company	Sector	Industry
KESBV	FI0009000202	Kesko Oyj	Consumer Defensive	Grocery Stores
OUT1V	FI0009002422	Outokumpu Oyj	Basic Materials	Steel
SAMPO	FI0009003305	Sampo Oyj	Financial Services	Insurance
RTRKS	FI0009003552	Rautaruukki Oyj	Basic Materials	Steel
WRT1V	FI0009000727	Wärtsilä Oyj	Industrials	Diversified Industrials

TABLE 2 Message list example

Timestamp	ID	Price	Quantity	Event	Side
1275386347944	6505727	126200	400	Cancellation	Ask
1275386347981	6505741	126500	300	Submission	Ask
1275386347981	6505741	126500	300	Cancellation	Ask
1275386348070	6511439	126100	17	Execution	Bid
1275386348070	6511439	126100	17	Submission	Bid
1275386348101	6511469	126600	300	Cancellation	Ask

TABLE 3 Order book example

			Level 1				Level 2				…
			Ask		Bid		Ask		Bid		
Timestamp	Mid-price	Spread	Price	Quantity	Price	Quantity	Price	Quantity	Price	Quantity	
1275386347944	126200	200	126300	300	126100	17	126400	4765	126000	2800	…
1275386347981	126200	200	126300	300	126100	17	126400	4765	126000	2800	…
1275386347981	126200	200	126300	300	126100	17	126400	4765	126000	2800	…
1275386348070	126050	100	126100	291	126000	2800	126200	300	125900	1120	…
1275386348070	126050	100	126100	291	126000	2800	126200	300	125900	1120	…
1275386348101	126050	100	126100	291	126000	2800	126200	300	125900	1120	…

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{3}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{4}$$

where TP and TF represent the true positives and true negatives, respectively, of the mid-price prediction label compared with the ground truth, where FP and FN represents the false positives and false negatives, respectively. From among the above metrics, we focus on the $F1$ score performance. The main reason that we focus on $F1$ score is based on its ability only to be affected in one direction of skew distributions, in the case of unbalanced classes like ours. On the contrary, accuracy cannot differentiate between the number of correct labels (i.e., related to mid-price movement direction prediction) of different classes where the other three metrics can separate the correct labels among different classes, with $F1$ being the harmonic mean of Precision and Recall.

We follow an event-based inflow, as used in Li, et al. (2016). This is due to the fact that events (i.e., orders, executions, and cancellations) do not follow a uniform inflow rate. Time intervals between two consecutive events can vary from milliseconds to several minutes of difference. Event-based data representation avoids issues related to such big differences in data flow. As a result, each of our representations is a vector that contains information for 10 consecutive events. Event-based data description leads to a dataset of approximately half a million representations (i.e., 394,337 representations). We represent these events using the 144-dimensional representation proposed recently by Kercheval and Zhang (2015), formed by three types of features: (a) the raw data of a 10-level limit order containing price and volume values for bid and ask orders; (b) features describing the state of the LOB, exploiting past information; and (c) features describing the information edge in the raw data by taking time into account. Derivations of time, stock price, and volume are calculated for short and long-term projections. More specifically, types in features u_7, u_8, and u_9 are: *trades, orders, cancellations, deletion, execution of a visible limit order*, and *execution of a hidden limit order*. Expressions used for calculating these features are provided in Table 4. One limitation of the adopted features

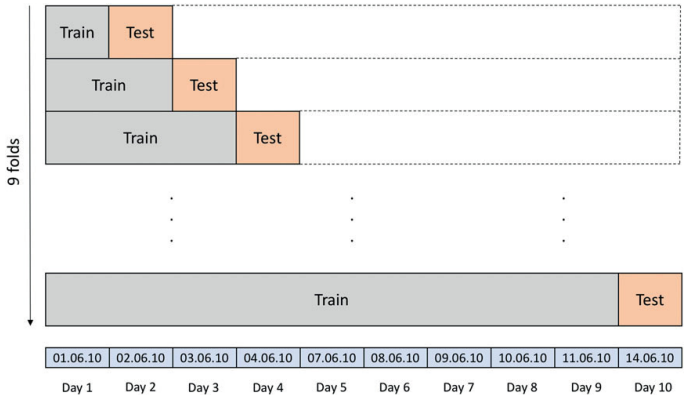


FIGURE 2 Experimental setup framework [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 4 Feature sets

Feature set	Description	Details
Basic	$u_1 = \{P_i^{\text{ask}}, V_i^{\text{ask}}, P_i^{\text{bid}}, V_i^{\text{bid}}\}_{i=1}^{n}$	$10(=n)$-level LOB data
Time-insensitive	$u_2 = \{(P_i^{\text{ask}} - P_i^{\text{bid}}), (P_i^{\text{ask}} + P_i^{\text{bid}})/2\}_{i=1}^{n}$	Spread & Mid-price
	$u_3 = \{P_n^{\text{ask}} - P_1^{\text{ask}}, P_1^{\text{bid}} - P_n^{\text{bid}}, \lvert P_{i+1}^{\text{ask}} - P_i^{\text{ask}}\rvert, \lvert P_{i+1}^{\text{bid}} - P_i^{\text{bid}}\rvert\}_{i+1}^{n}$	Price differences
	$u_4 = \left\{ \frac{1}{n}\sum_{i=1}^{n} P_i^{\text{ask}}, \frac{1}{n}\sum_{i=1}^{n} P_i^{\text{bid}}, \frac{1}{n}\sum_{i=1}^{n} V_i^{\text{ask}}, \frac{1}{n}\sum_{i=1}^{n} V_i^{\text{bid}} \right\}$	Price & Volume means
	$u_5 = \left\{ \sum_{i=1}^{n} (P_i^{\text{ask}} - P_i^{\text{bid}}), \sum_{i=1}^{n} (V_i^{\text{ask}} - V_i^{\text{bid}}) \right\}$	Accumulated differences
Time-sensitive	$u_6 = \{\mathrm{d}P_i^{\text{ask}}/\mathrm{d}t, \mathrm{d}P_i^{\text{bid}}/\mathrm{d}t, \mathrm{d}V_i^{\text{ask}}/\mathrm{d}t, \mathrm{d}V_i^{\text{bid}}/\mathrm{d}t\}_{i=1}^{n}$	Price & Volume derivation
	$u_7 = \{\lambda^1_{\Delta t}, \lambda^2_{\Delta t}, \lambda^3_{\Delta t}, \lambda^4_{\Delta t}, \lambda^5_{\Delta t}, \lambda^6_{\Delta t}\}$	Average intensity per type
	$u_8 = \left\{ \mathbf{1}_{\lambda^1_{\Delta_t} > \lambda^1_{\Delta_T}}, \mathbf{1}_{\lambda^2_{\Delta_t} > \lambda^2_{\Delta_T}}, \mathbf{1}_{\lambda^3_{\Delta_t} > \lambda^3_{\Delta_T}}, \mathbf{1}_{\lambda^4_{\Delta_t} > \lambda^4_{\Delta_T}}, \mathbf{1}_{\lambda^5_{\Delta_t} > \lambda^5_{\Delta_T}}, \mathbf{1}_{\lambda^6_{\Delta_t} > \lambda^6_{\Delta_T}} \right\}$	Relative intensity comparison
	$u_9 = \{\mathrm{d}\lambda^1/\mathrm{d}t, \mathrm{d}\lambda^2/\mathrm{d}t, \mathrm{d}\lambda^3/\mathrm{d}t, \mathrm{d}\lambda^4/\mathrm{d}t, \mathrm{d}\lambda^5/\mathrm{d}t, \mathrm{d}\lambda^6/\mathrm{d}t\}$	Limit activity acceleration

is the lack of information related to order flow (i.e., the sequence of order book messages). However, as can be seen in the Results Section 6, the baselines achieve relatively good performance and therefore we leave the introduction of extra features that can enhance performance to future research.

We provide three sets of data, each created by following a different data normalization strategy—that is, *z*-score, min–max, and decimal precision normalization—for every *i* data sample. *Z*-score, in particular, is the normalization process through which we subtract the mean from our input data for each feature separately and divide by the standard deviation of the given sample:

$$\mathbf{x}_i^{(\text{z-score})} = \frac{\mathbf{x}_i - \frac{1}{N}\sum_{j=1}^{N} \mathbf{x}_j}{\sqrt{\frac{1}{N}\sum_{j=1}^{N} (\mathbf{x}_j - \bar{\mathbf{x}})^2}}, \tag{5}$$

where $\bar{\mathbf{x}}$ denotes the mean vector, as appears in Equation 5. On the other hand, min–max scaling, as described by

$$\mathbf{x}_i^{(\text{MM})} = \frac{\mathbf{x}_i - \mathbf{x}_{\min}}{\mathbf{x}_{\max} - \mathbf{x}_{\min}}, \tag{6}$$

is the process of subtracting the minimum value from each feature and dividing it by the difference between the maximum and minimum value of that feature sample. The third scaling setup is the decimal precision approach. This normalization method is based on moving the decimal points of each of the feature values. Calculations follow the absolute value of each feature sample:

$$\mathbf{x}_i^{(\text{DP})} = \frac{\mathbf{x}_i}{10^k}, \tag{7}$$

where *k* is the integer that will give us the maximum value for $|\mathbf{x}_{\text{DP}}| < 1$.

Having defined the event representations, we use five different projection horizons for our labels. Each of these

TABLE 5 HFT dataset examples

	Dataset	Public available	Unit time	Period	Asset class / No. of stocks	Size	Annotations
1	Dukascopy	✓	ms	Up to date	Various	∼20,000 events/day	×
2	truefx	✓	ms	Up to date	15 FX pairs	∼300,000 events/day	×
3	Nasdaq	AuR	ms	2008-09	Equity / 120	—	×
4	Nasdaq	AuR	ms	10/07 & 06/08	Equity / 500	∼55,000 events/day	×
5	Nasdaq	×	ms	—	Equity / 5	2,000 data points	×
6	Euronext	AuR	—	—	Several products	—	×
7	Nasdaq	×	ns	01/14-08/15	Equity / 489	50 TB	×
8	Our–Nasdaq	✓	ms	01-14/06/10	Equity / 5	4 M samples	✓

horizons portrays a different future projection interval of the mid-price movement (i.e., upward, downward, and stationary mid-price movement). More specifically, we extract labels based on short-term and long-term, event-based, relative changes for the next 1, 2, 3, 5, and 10 events for our representations dataset.

Our labels describe the percentage change of the mid-price, which is calculated as follows:

$$l_i^{(j)} = \frac{\frac{1}{k}\sum_{j=i+1}^{i+k} m_j - m_i}{m_i}, \tag{8}$$

where m_j is the future mid-price ($k = 1, 2, 3, 5,$ or 10 next events in our representations) and m_i is the current mid-price. The extracted labels are based on a threshold for the percentage change of 0.002. For percentage changes equal to or greater than 0.002, we use label 1. For percentage change that varies from −0.00199 to 0.00199, we use label 2, and, for percentage change smaller or equal to −0.002, we use label 3.

4 | EXISTING DATASETS DESCRIBED IN THE LITERATURE

In this section, we list existing HFT datasets described in the literature and provide qualitative and quantitative comparisons to our dataset. The following works mainly focus on datasets that are related to machine learning methods.

There are mainly three sources of data from which a high-frequency trader can choose. The first option is the use of publicly available data (e.g., (1) Dukascopy and (2) truefx), where no prior agreement is required for data acquisition. The second option is publicly available data upon request for academic purposes, which can be found in (3) Brogaard, Hendershott, and Riordan (2014), (4) Hasbrouck and Saar (2013), (5) De Winne and D'hondt 2007, Detollenaere and D'hondt (2017), and Carrion (2013). Finally, the third and most common option is data through platforms requiring a subscription fee, like those in (6) Kercheval and Zhang (2015); Li et al. (2016), and (7) Sirignano (2016). Existing data sources and characteristics are listed in Table 5.

In particular, the datasets are at a millisecond resolution, except for number 6 in the table. Access to various asset classes including FX, commodities, indices, and stocks is also provided. To the best of our knowledge, there is no available literature based on this type of dataset for equities. Another source of free tick-by-tick historical data is the truefx.com site, but the site provides data only for the FX market for several pairs of currencies at a millisecond resolution. The data contain information regarding timestamps (in millisecond resolution) and bid and ask prices. Each of these .csv files contains approximately 200,000 events per day. This type of data is used in a mean-reverting jump-diffusion model, as presented in Suwanpetai (2016).

There is a second category of datasets available upon request (AuR), as seen in Hasbrouck and Saar (2013). In this paper, the authors use the Nasdaq OMX ITCH for two periods: October 2007 and June 2008. For that period, they run samples at 10-minute intervals for each day where they set a cutoff mechanism for available messages per period.[24] The main disadvantage of uniformly sampling HFT data is that the trader loses vital information. Events come randomly, with inactive periods varying from a few milliseconds to several minutes or hours. In our work, we overcome this challenge by considering the information based on event inflow, rather than equal time sampling. Another example of data that is available only for academic purposes is Brogaard et al. (2014). The dataset contains information regarding timestamps, price, and buy–sell side prices but no other details related to daily events or feature vectors. Hasbrouck and Saar provide a detailed description of their Nasdaq OMX ITCH data, which is not directly accessible for testing and comparison with their

[24]The authors provide a threshold, which is based on 250 events per 10-minute sample interval.

baselines. They use these data to applying low-latency strategies based on measures that capture links between submissions, cancellations, and executions. De Winne and D'hondt (2007) and Detollenaere and D'hondt (2017) use similar datasets from Euronext for LOB construction. They specify that their dataset is available upon request from the provider. What is more, the data provider supplies details regarding the LOB construction by the user. Our work fills that gap since our dataset provides the full LOB depth and it is ready for use and comparison with our baselines.

The last category of dataset has dissemination restrictions. An example is the paper by Kercheval and Zhang (2015), where the authors are trying to predict the mid-price movement by using machine learning (i.e., SVM). They train their model with a very small number of samples (i.e., 4,000 samples). The HFT activity can produce a huge volume of trading events daily, as our database does with 100,000 daily events for only one stock. Moreover, the datasets in Kercheval and Zhang and in Sirignano (2016) are not publicly available, which makes comparison with other methods impossible. In the same direction, we also add works such as Hasbrouck (2009), Kalay, Sade, and Wohl (2004), and Kalay, Wei, and Wohl (2002), which utilize TAQ and Tel Aviv stock exchange datasets (not for machine learning methods), and require subscription.

5 | BASELINES

In order to provide performance baselines for our new dataset of HFT with LOB data, we conducted experiments with two regression models using the data representations described in Section 3.4. Details on the models used are provided in Sections 5.1 and 5.2. The baseline performances are provided in Section 6.

5.1 | Ridge regression (RR)

Ridge regression defines a linear mapping, expressed by the matrix $\mathbf{W} \in \mathbb{R}^{D\times C}$, that optimally maps a set of vectors $\mathbf{x}_i \in \mathbb{R}^D,\ i = 1, \cdots, N$ to another set of vectors (noted as target vectors) $\mathbf{t}_i \in \mathbb{R}^C,\ i = 1, \cdots, N$, by optimizing the following criterion:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \sum_{i=1}^{N} \|\mathbf{W}^T\mathbf{x}_i - \mathbf{t}_i\|_2^2 + \lambda\|\mathbf{W}\|_F^2, \tag{9}$$

or using a matrix notation:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \|\mathbf{W}^T\mathbf{X} - \mathbf{T}\|_F^2 + \lambda\|\mathbf{W}\|_F^2. \tag{10}$$

In the above, $\mathbf{X} = [\mathbf{x}_i, \ldots, \mathbf{x}_N]$ and $\mathbf{T} = [\mathbf{t}_i, \ldots, \mathbf{t}_N]$ are matrices formed by the samples $\mathbf{x}_i$ and $\mathbf{t}_i$ as columns, respectively.

In our case, each sample $\mathbf{x}_i$ corresponds to an event, represented by a vector (with $D = 144$), as described in Section 3.4. For the three-class classification problems in our dataset, the elements of vectors $\mathbf{t}_i \in \mathbb{R}^C$ ($C = 3$ in our case) take values equal to $t_{ik} = 1$, if $\mathbf{x}_i$ belongs to class k, and if $t_{ik} = -1$ otherwise. The solution of Equation 10 is given by

$$\mathbf{W} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{T}^T, \tag{11}$$

or

$$\mathbf{W} = \left(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}\right)^{-1}\mathbf{X}\mathbf{T}^T, \tag{12}$$

where $\mathbf{I}$ is the identity matrix of appropriate dimensions. Here, we should note that, in our case, where the size of the data is large, $\mathbf{W}$ should be computed using Equation 12, since the calculation of Equation 11 is computationally very expensive.

After the calculation of $\mathbf{W}$, a new (test) sample $\mathbf{x} \in \mathbb{R}^D$ is mapped on its corresponding representation in space $\mathbb{R}^C$—that is, $\mathbf{o} = \mathbf{W}^T\mathbf{x}$—and is classified according to the maximum value of its projection:

$$l_{\mathbf{x}} = \arg\max_k o_k. \tag{13}$$

5.2 | SLFN network-based nonlinear regression

We also test the performance of a nonlinear regression model. Since the application of kernel-based regression is computationally too intensive for the size of our data, we use an SLFN (Figure 3) network-based regression model. Such a model is formed as follows.

For fast network training, we train our network based on the algorithm proposed in Huang, Zhou, Ding, and Zhang (2012), Zhang, Kwok, and Parvin (2009), and Iosifidis, Tefas, and Pitas (2017). This algorithm is formed by

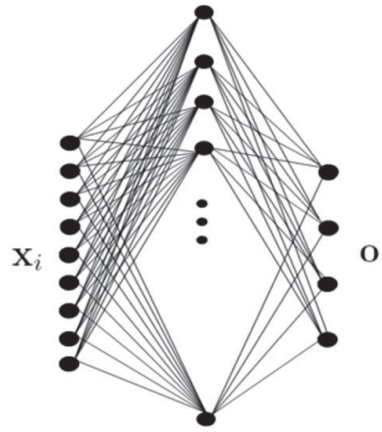


FIGURE 3 SLFN

two processing steps. In the first step, the network's hidden layer weights are determined either randomly (Huang, Zhou, Ding, & Zhang, 2012) or by applying clustering on the training data. We apply K-means clustering in order to determine K prototype vectors, which are subsequently used as the network's hidden layer weights.

Having determined the network's hidden layer weights $\mathbf{V} \in \mathbb{R}^{D\times K}$, the input data $\mathbf{x}_i, i = 1, \dots, N$ are nonlinearly mapped to vectors $\mathbf{h}_i \in \mathbb{R}^K$, expressing the data representations in the feature space determined by the network's hidden layer outputs $\mathbb{R}^K$. We use the radial basis function—that is, $\mathbf{h}_i = \varphi_{\text{RBF}}(\mathbf{x}_i)$—calculated in an element-wise manner, as follows:

$$h_{ik} = \exp\left(\frac{\|\mathbf{x}_i - \mathbf{v}_k\|_2^2}{2\sigma^2}\right), \quad k = 1, \cdots, K, \tag{14}$$

where σ is a hyperparameter denoting the spread of the RBF neuron and $\mathbf{v}_k$ corresponds to the kth column of $\mathbf{V}$.

The network's output weights $\mathbf{W} \in \mathbb{R}^{K\times C}$ are subsequently determined by solving for

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \|\mathbf{W}^T\mathbf{H} - \mathbf{T}\|_F^2 + \lambda\|\mathbf{W}\|_F^2, \tag{15}$$

where $\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_N]$ is a matrix formed by the network's hidden layer outputs for the training data and $\mathbf{T}$ is a matrix formed by the network's target vectors $\mathbf{t}_i, i = 1, \dots, N$ as defined in Section 5.1. The network's output weights are given by

$$\mathbf{W} = \left(\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I}\right)^{-1}\mathbf{H}\mathbf{T}^T. \tag{16}$$

After calculation of the network parameters $\mathbf{V}$ and $\mathbf{W}$, a new (test) sample $\mathbf{x} \in \mathbb{R}^D$ is mapped on its corresponding representations in spaces $\mathbb{R}^K$ and $\mathbb{R}^C$; that is, $\mathbf{h} = \phi_{\text{RBF}}(\mathbf{x})$ and $\mathbf{o} = \mathbf{W}^T\mathbf{h}$, respectively. It is classified according to the maximal network output:

$$l_{\mathbf{x}} = \arg\max_k o_k. \tag{17}$$

6 | RESULTS

In our first set of experiments, we have applied two supervised machine learning methods, as described in Sections 5.1 and 5.2, on a dataset that does not include the auction period. Results with the auction period will also be available. Since there is not a widely adopted experimental protocol for these datasets, we provide information for the five different label scenarios under the three normalization setups.

The tables in this section provide details regarding the results of experiments conducted on raw data and three different normalization setups. We present these results, for our baseline models, in order to give insight into the preprocessing step for a dataset like ours, to examine the strength of the predictability of the projected time horizon, and to understand the implications of the suggested methods. Data normalization can significantly improve the metric's performance in combination with the use of the right classifier. More specifically, we measure the predictability power of our models via the performance of the metrics of accuracy, precision, recall, and $F1$ score. For instance, Table 6 presents the results based on raw data (i.e., no data decoding), and in the case of the linear classifier RR and label 5 (i.e., the 5th mid-price event as predicted horizon), we achieve an $F1$ score of 40%, where as in Table 7 (i.e., the Z-score data decoding method), Table 8 (i.e., min–max data decoding method), and Table 9 (i.e., the decimal precision decoding method), we achieve 43%, 42%, and 40%, respectively. This shows

TABLE 6 Results based on unfiltered representations

Label	$\text{RR}_{\text{Accuracy}}$	$\text{RR}_{\text{Precision}}$	$\text{RR}_{\text{Recall}}$	RR_{F1}
1	0.637 ± 0.055	0.505 ± 0.145	0.337 ± 0.003	0.268 ± 0.014
2	0.555 ± 0.064	0.504 ± 0.131	0.376 ± 0.023	0.320 ± 0.050
3	0.489 ± 0.061	0.423 ± 0.109	0.397 ± 0.031	0.356 ± 0.070
5	0.429 ± 0.049	0.402 ± 0.113	0.425 ± 0.038	0.400 ± 0.093
10	0.453 ± 0.054	0.400 ± 0.105	0.400 ± 0.030	0.347 ± 0.066
Label	$\text{SLFN}_{\text{Accuracy}}$	$\text{SLFN}_{\text{Precision}}$	$\text{SLFN}_{\text{Recall}}$	SLFN_{F1}
1	0.636 ± 0.055	0.299 ± 0.075	0.335 ± 0.002	0.262 ± 0.015
2	0.536 ± 0.069	0.387 ± 0.132	0.345 ± 0.009	0.260 ± 0.035
3	0.473 ± 0.074	0.334 ± 0.080	0.357 ± 0.005	0.270 ± 0.021
5	0.381 ± 0.038	0.342 ± 0.058	0.370 ± 0.020	0.327 ± 0.043
10	0.401 ± 0.039	0.284 ± 0.102	0.356 ± 0.020	0.290 ± 0.070

TABLE 7 Results based on *Z*-score normalization

Label	$RR_{Accuracy}$	$RR_{Precision}$	RR_{Recall}	RR_{F1}
1	0.480 ± 0.040	0.418 ± 0.021	0.435 ± 0.029	0.410 ± 0.022
2	0.498 ± 0.052	0.444 ± 0.025	0.443 ± 0.031	0.440 ± 0.031
3	0.463 ± 0.045	0.438 ± 0.027	0.437 ± 0.033	0.433 ± 0.034
5	0.439 ± 0.042	0.436 ± 0.028	0.433 ± 0.028	0.427 ± 0.041
10	0.429 ± 0.046	0.429 ± 0.028	0.429 ± 0.043	0.416 ± 0.044
Label	$SLFN_{Accuracy}$	$SLFN_{Precision}$	$SLFN_{Recall}$	$SLFN_{F1}$
1	0.643 ± 0.056	0.512 ± 0.037	0.366 ± 0.019	0.327 ± 0.046
2	0.556 ± 0.066	0.550 ± 0.029	0.378 ± 0.011	0.327 ± 0.030
3	0.512 ± 0.069	0.497 ± 0.024	0.424 ± 0.047	0.389 ± 0.082
5	0.473 ± 0.036	0.468 ± 0.024	0.464 ± 0.028	0.459 ± 0.031
10	0.477 ± 0.048	0.453 ± 0.056	0.432 ± 0.025	0.410 ± 0.040

TABLE 8 Results Based on min–max normalization

Label	$RR_{Accuracy}$	$RR_{Precision}$	RR_{Recall}	RR_{F1}
1	0.637 ± 0.054	0.499 ± 0.118	0.339 ± 0.005	0.272 ± 0.015
2	0.561 ± 0.063	0.467 ± 0.117	0.400 ± 0.028	0.368 ± 0.060
3	0.492 ± 0.070	0.428 ± 0.111	0.400 ± 0.030	0.357 ± 0.072
5	0.437 ± 0.048	0.419 ± 0.078	0.429 ± 0.043	0.417 ± 0.063
10	0.452 ± 0.054	0.421 ± 0.110	0.399 ± 0.028	0.348 ± 0.066
Label	$SLFN_{Accuracy}$	$SLFN_{Precision}$	$SLFN_{Recall}$	$SLFN_{F1}$
1	0.640 ± 0.055	0.488 ± 0.104	0.348 ± 0.007	0.291 ± 0.022
2	0.558 ± 0.065	0.469 ± 0.066	0.399 ± 0.023	0.367 ± 0.050
3	0.499 ± 0.063	0.447 ± 0.068	0.410 ± 0.032	0.370 ± 0.063
5	0.453 ± 0.038	0.441 ± 0.041	0.444 ± 0.030	0.432 ± 0.050
10	0.450 ± 0.048	0.432 ± 0.070	0.406 ± 0.037	0.377 ± 0.062

TABLE 9 Results based on decimal precision normalization

Label	$RR_{Accuracy}$	$RR_{Precision}$	RR_{Recall}	RR_{F1}
1	0.638 ± 0.054	0.518 ± 0.132	0.341 ± 0.007	0.277 ± 0.018
2	0.551 ± 0.066	0.473 ± 0.118	0.372 ± 0.018	0.315 ± 0.045
3	0.490 ± 0.069	0.432 ± 0.113	0.386 ± 0.023	0.330 ± 0.059
5	0.435 ± 0.051	0.406 ± 0.115	0.430 ± 0.039	0.405 ± 0.095
10	0.451 ± 0.052	0.417 ± 0.108	0.399 ± 0.029	0.349 ± 0.067
Label	$SLFN_{Accuracy}$	$SLFN_{Precision}$	$SLFN_{Recall}$	$SLFN_{F1}$
1	0.641 ± 0.055	0.512 ± 0.027	0.351 ± 0.007	0.297 ± 0.024
2	0.565 ± 0.063	0.505 ± 0.020	0.410 ± 0.026	0.385 ± 0.054
3	0.504 ± 0.061	0.465 ± 0.032	0.421 ± 0.040	0.393 ± 0.073
5	0.457 ± 0.038	0.451 ± 0.029	0.449 ± 0.031	0.438 ± 0.046
10	0.461 ± 0.053	0.453 ± 0.036	0.420 ± 0.035	0.399 ± 0.053

that in the case of the linear classifier the suggested decoding methods did not offer any significant improvements, since the variability of the performance range is approximately 3%. On the other hand, our nonlinear classifier (i.e., SLFN) for the same projected time horizon (i.e., label 5) reacted more efficiently in the decoding process. SLFN achieves 33% for the *F*1 score for nonnormalized data, while the *Z*-score, min–max and decimal precision methods achieve 46%, 43%, and 43%, respectively. As a result, normalization improves the *F*1 score performance by almost 10%.

Normalization and model selection can also affect the predictability of mid-price movements over the projected time horizon. Very interesting results come to light if we try to compare the *F*1 performance over different time horizons. For instance, we can see that, regardless of the decoding method, the *F*1 score is always better for label 5 than 1, meaning that 'our models'

predictions are better further in the future. This result is significant, especially with unfiltered data and min–max and decimal precision normalizations, when $F1$ score is approximately 27%, in the case of the one-step prediction problem (label 1), and 43% in the case of the five-step problem (label 5).

Another aspect of the experimental results above stems from the pros and cons of linear and nonlinear classifiers. More specifically, the RR linear classifier performed better on the raw dataset and for the Z-score decoding method in terms of $F1$ when compared to the SLFN (i.e., nonlinear classifier). This is not the case for the last decoding methods (i.e., min–max and decimal precision), where our nonlinear classifier presents similar or better results than RR. An explanation for this $F1$ performance discrepancy is due to each of these methods' engineering has. The RR classifier tends to be very efficient in high-dimensional problems, and these types of problems are linearly separable, in most cases. Another reason that RR can perform better when compared to a nonlinear classifier is that RR can control the complexity by penalizing the bias, via cross-validation, using the ridge parameter. On the other hand, a nonlinear classifier is prone to overfitting, which means that in some cases it offers a better degree of freedom for class separation.

7 | CONCLUSION

This paper described a new benchmark dataset formed by the Nasdaq ITCH feed data for five stocks for 10 consecutive trading days. Data representations that were exploited by order flow features were made available. We formulated five classification tasks based on mid-price movement predictions for 1, 2, 3, 5, and 10 predicted horizons. Baseline performances of two regression models were also provided in order to facilitate future research in the field. Despite the data size, we achieved an average out-of-sample performance ($F1$) of approximately 46% for both methods. These very promising results show that machine learning can effectively predict mid-price movement.

Potential avenues of research that can benefit from exploiting the provided data include: (a) prediction of the stability of the market, which is very important for liquidity providers (market makers) to make the spread, as well as for traders to increase liquidity provision (when markets can be predicted to be stable); (b) prediction on market movements, which is important for expert systems used by speculative traders; (c) identification of order book spoofing—that is, situations where markets are manipulated by limit orders. Although there is no spoofing activity information available for the provided data, the exploitation of such a large corpus of data can be used in order to identify patterns in stock markets that can be further analyzed as normal or abnormal.

ACKNOWLEDGMENT

This work was supported by H2020 Project BigDataFinance MSCA-ITN-ETN 675044 (http://bigdatafinance.eu), Training for Big Data in Financial Research and Risk Management.

ORCID

Adamantios Ntakaris http://orcid.org/0000-0001-6949-5337

REFERENCES

Abernethy, J., & Kale, S. (2013). Adaptive market making via online learning, *Advances in Neural Information Processing Systems* (pp. 2058–2066). Cambridge, MA: MIT Press.

Almgren, R., & Lorenz, J. (2006). Bayesian adaptive trading with a daily cycle. *Journal of Trading*, *1*(4), 38–46.

Alvim, L. G., dos Santos, C. N., & Milidiu, R. L. (2010). Daily volume forecasting using high frequency predictors. In *Proceedings of the 10Th IASTED International Conference*, Acta Press, Calgary, Canada, Vol. 674, pp. 248.

Amaya, D., Filbien, J.-Y, Okou, C., & Roch, A. F. (2015). Distilling liquidity costs from limit order books. Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstractid=2660226.

An, Y., & Chan, N. H. (2017). Short-term stock price prediction based on limit order book dynamics. *Journal of Forecasting*, *36*(5), 541–556.

Aramonte, S., Schindler, J. W., & Rosen, S. (2013). Assessing and combining financial conditions indexes. Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstractid=2976840.

Avellaneda, M., & Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, *8*(3), 217–224.

Bogoev, D., & Karam, A. (2016). An Empirical Detection of High Frequency Trading Strategies. (*Working Paper*). Durham, UK: Durham University.

Brogaard, J., Hendershott, T., & Riordan, R. (2014). High-frequency trading and price discovery. *Review of Financial Studies*, *27*(8), 2267–2306.

Cao, C., Hansch, O., & Wang, X. (2009). The information content of an open limit-order book. *Journal of Futures Markets*, *29*(1), 16–41.

Carrion, A. (2013). Very fast money: High-frequency trading on the NASDAQ. *Journal of Financial Markets*, *16*(4), 680–711.

Cenesizoglu, T., Dionne, G., & Zhou, X. (2014). Effects of the limit order book on price dynamics. Retrieved from https://depot.erudit.org/bitstream/003996dd/1/CIRPEE14-26.pdf.

Chan, N. T., & Shelton, C. (2001). An electronic market-maker. Retrieved from https://dspace.mit.edu/bitstream/handle/1721.1/7220/AIM-2001-005.pdf?sequence=2.

Chang, Y. L. (2015). Inferring Markov chain for modeling order book dynamics in high frequency environment. *International Journal of Machine Learning and Computing*, *5*(3), 247–251.

Christensen, H. L., & Woodmansey, R. (2013). Prediction of hidden liquidity in the limit order book of globex futures. *Journal of Trading*, *8*(3), 68–95.

Creamer, G. (2012). Model calibration and automated trading agent for euro futures. *Quantitative Finance*, *12*(4), 531–545.

De Winne, R., & D'hondt, C. (2007). Hide-and-seek in the market: placing and detecting hidden orders. *Review of Finance*, *11*(4), 663–692.

Detollenaere, B., & D'hondt, C. (2017). Identifying expensive trades by monitoring the limit order book. *Journal of Forecasting*, *36*(3), 273–290.

Dixon, M. (2016). High frequency market making with machine learning. Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstractid=2868473.

Felker, T., Mazalov, V., & Watt, S. M. (2014). Distance-based high-frequency trading. *Procedia Computer Science*, *29*, 2055–2064.

Fletcher, T., Hussain, Z., & Shawe-Taylor, J. (2010). Multiple kernel learning on the limit order book. In *Proceedings of the First Workshop on Applications of Pattern Analysis*, Vol. *11*, pp. 167–174.

Galeshchuk, S. (2016). Neural networks performance in exchange rate prediction. *Neurocomputing*, *172*, 446–452.

Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., & Howison, S. D. (2013). Limit order books. *Quantitative Finance*, *13*(11), 1709–1742.

Hallgren, J., & Koski, T. (2016). Testing for causality in continuous time Bayesian network models of high-frequency data. arXiv preprint retrieved from https://arxiv.org/abs/1601.06651.

Han, J., Hong, J., Sutardja, N., & Wong, S. F. (2015). Machine Learning Techniques for Price Change Forecast Using the Limit Order Book Data. (*Working Paper*). Berkeley, CA: University of California, Berkeley.

Hasbrouck, J. (2009). Trading costs and returns for US equities: Estimating effective costs from daily data. *Journal of Finance*, *64*(3), 1445–1477.

Hasbrouck, J., & Saar, G. (2013). Low-latency trading. *Journal of Financial Markets*, *16*(4), 646–679.

Huang, G.-B., Zhou, H., Ding, X., & Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, *42*(2), 513–529.

Iosifidis, A., Tefas, A., & Pitas, I. (2017). Approximate kernel extreme learning machine for large scale data classification. *Neurocomputing*, *219*, 210–220.

Kalay, A., Sade, O., & Wohl, A. (2004). Measuring stock illiquidity: An investigation of the demand and supply schedules at the TASE. *Journal of Financial Economics*, *74*(3), 461–486.

Kalay, A., Wei, L., & Wohl, A. (2002). Continuous trading or call auctions: Revealed preferences of investors at the Tel Aviv stock exchange. *Journal of Finance*, *57*(1), 523–542.

Kearns, M., & Nevmyvaka, Y. (2013). Machine Learning for Market Microstructure and High Frequency Trading. In D. Easley, M. López De Prado, & M. O'Hara (Eds.), *High Frequency Trading: New Realities for Traders, Markets and Regulators*. London, UK: Risk Books.

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

Kim, A. J. (2001). Input/Output Hidden Markov Models for Modeling Stock Order Flows. (*Technical Report No. 1370*). Cambridge, MA: MITAI Laboratory.

Levendovszky, J., & Kia, F. (2012). Prediction based-high frequency trading on financial time series. *Periodica Polytechnica: Electrical Engineering and Computer Science*, *56*(1), 29–34.

Li, X., Xie, H., Wang, R., Cai, Y., Cao, J., Wang, F., Min, H., & Deng, F. (2016). Empirical analysis: Stock market prediction via extreme learning machine. *Neural Computing and Applications*, *27*(1), 67–78.

Liu, J., & Park, S. (2015). Behind stock price movement: Supply and demand in market microstructure and market influence. *Journal of Trading*, *10*(3), 13–23.

Maglaras, C., Moallemi, C. C., & Zheng, H. (2015). Optimal execution in a limit order book and an associated microstructure market impact model. Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstractid=2610808.

Majhi, R., Panda, G., & Sahoo, G. (2009). Development and performance evaluation of FLANN based model for forecasting of stock markets. *Expert Systems with Applications*, *36*(3), 6800–6808.

Malik, A., & Lon Ng, W. (2014). Intraday liquidity patterns in limit order books. *Studies in Economics and Finance*, *31*(1), 46–71.

Mankad, S., Michailidis, G., & Kirilenko, A. (2013). Discovering the ecosystem of an electronic financial market with a dynamic machine-learning method. *Algorithmic Finance*, *2*(2), 151–165.

Næs, R., & Skjeltorp, J. A. (2006). Order book characteristics and the volume–volatility relation: Empirical evidence from a limit order market. *Journal of Financial Markets*, *9*(4), 408–432.

O'Hara, M., & Ye, M. (2011). Is market fragmentation harming market quality? *Journal of Financial Economics*, *100*(3), 459–474.

Pai, P.-F., & Lin, C.-S (2005). A hybrid Arima and support vector machines model in stock price forecasting. *Omega*, *33*(6), 497–505.

Palguna, D., & Pollak, I. (2016). Mid-price prediction in a limit order book. *IEEE Journal of Selected Topics in Signal Processing*, *10*(6), 1083–1092.

Panayi, E., Peters, G. W., Danielsson, J., & Zigrand, J.-P. (2016). Designating market maker behaviour in limit order book markets. *Econometrics and Statistics*, *5*, 20–44.

Ranaldo, A. (2004). Order aggressiveness in limit order book markets. *Journal of Financial Markets*, *7*(1), 53–74.

Rehman, M., Khan, G. M., & Mahmud, S. A. (2014). Foreign currency exchange rates prediction using CGP and recurrent neural network. *IERI Procedia*, *10*, 239–244.

Sandoval, J., & Hernández, G. (2015). Computational visual analysis of the order book dynamics for creating high-frequency foreign exchange trading strategies. *Procedia Computer Science*, *51*, 1593–1602.

Seddon, J. J., & Currie, W. L. (2017). A model for unpacking big data analytics in high-frequency trading. *Journal of Business Research*, *70*, 300–307.

Sharang, A., & Rao, C. (2015). Using machine learning for medium frequency derivative portfolio trading. arXiv preprint retrieved from https://arxiv.org/abs/1512.06228

Siikanen, M., Kanniainen, J., & Luoma, A. (2017). What drives the sensitivity of limit order books to company announcement arrivals? *Economics Letters*, *159*, 65–68.

Siikanen, M., Kanniainen, J., & Valli, J. (2017). Limit order books and liquidity around scheduled and non-scheduled announcements: Empirical evidence from NASDAQ Nordic. *Finance Research Letters*, *21*, 264–271.

Sirignano, J. (2016). Deep learning for limit order books. Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2710331.

Suwanpetai, P. (2016). Estimation of exchange rate models after news announcement. In *AP16Thai Conference 2016: Sixth Asia–Pacific Conference on Global Business, Economics, Finance and Social Sciences*.

Talebi, H., Hoang, W., & Gavrilova, M. L. (2014). Multi-scale foreign exchange rates ensemble for classification of trends in FOREX market. *Procedia Computer Science*, *29*, 2065–2075.

Vella, V., & Ng, W. L. (2016). Improving risk-adjusted performance in high frequency trading using interval type-2 fuzzy logic. *Expert Systems with Applications*, *55*, 70–86.

Yang, S., Paddrik, M., Hayes, R., Todd, A., Kirilenko, A., Beling, P., & Scherer, W. (2012). Behavior Based Learning in Identifying High Frequency Trading Strategies. In *2012 IEEE Conference on Computational Intelligence for Financial Engineering and Economics (CIFEr)*, IEEE, Piscataway, NJ, pp. 1–8.

Yang, S. Y., Qiao, Q., Beling, P. A., Scherer, W. T., & Kirilenko, A. A. (2015). Gaussian process-based algorithmic trading strategy identification. *Quantitative Finance*, *15*(10), 1683–1703.

Yu, Y. (2006). The Limit Order Book Information and the Order Submission Strategy: a Model Explanation. In *2006 International Conference on Service Systems and Service Management*, IEEE, Piscataway, NJ, Vol. *1*, pp. 687–691.

Zhang, K., Kwok, J. T., & Parvin, B. (2009). Prototype Vector Machine for Large Scale Semi-Supervised Learning. In *Proceedings of the 26Th Annual International Conference on Machine Learning*, ACM, New York, NY, pp. 1233–1240.

Zheng, B., Moulines, E., & Abergel, F. (2012). Price jump prediction in limit order book. Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2026454.

Adamantios Ntakaris is an ESR within the Marie Curie BigDataFinance project in the Dept. of Signal Processing at Tampere University of Technology. He received a B.Sc. in Mathematics in 2009 from the Aristotle University of Thessaloniki and an M.Sc. in Financial Modelling and Optimization in 2014 from the University of Edinburgh. In 2014 Adamantios completed an industrial placement at Standard Life Investments in Edinburgh. Before commencing his PhD, he worked as an Effective Interest Rate Analyst at CitiGroup investment bank in Edinburgh, and as a Maths Olympiad Coach in Thessaloniki.

Martin Magris is an Early Stage Researcher within the Marie Curie BigDataFinance training network in the Laboratory of Industrial and Information Management at Tampere University of Technology (Finland) since April 2016. He received a B.Sc. in Statistics and Mathematics in 2013 and a M.Sc in Statistical and Actuarial Sciences in 2015 from Universitá degli studi di Trieste, Italy. As a part of his master studies, Martin visited Aarhus university for seven months in 2014. In the years 2015-2016, before commencing his PhD, Martin worked as actuarial analyst for a non-life insurance company, specifically in the car-insurance pricing and in the development, profit-testing and pricing of multiple-peril non-life insurance products.

Juho Kanniainen is a Professor of Financial Engineering at the Tampere University of Technology, Finland. His research agenda is focused on quantitative finance with emphasis on big data problems. Dr. Kanniainen has published in many journals in Finance and Engineering, including Review of Finance, Journal of Banking and Finance, and Digital Signal Processing. He has been coordinating two international EU projects, BigDataFinance (www.bigdatafinance.eu) and HPCFinance (www.hpcfinance.eu).

Moncef Gabbouj is a Professor of Signal Processing at the Department of Signal Processing, Tampere University of Technology, Tampere, Finland. He was Academy of Finland Professor during 2011-2015. He held several visiting professorships at different universities. Dr. Gabbouj is currently the TUT-Site Director of the NSF IUCRC funded Center for Visual and Decision Informatics. His research interests include Big Data analytics, multimedia content-based analysis, indexing and retrieval, artificial intelligence, machine learning, pattern recognition, nonlinear signal and image processing and analysis, voice conversion, and video processing and coding.

Alexandros Iosifidis is currently an Assistant Professor of Machine Learning and Computer Vision in the Department of Engineering, at Aarhus University, Denmark. He has held Postdoctoral Researcher positions in Tampere University of Technology, Finland and Aristotle University of Thessaloniki, Greece. He has participated in many R&D projects financed by EU, Greek, Finnish, and Danish funding agencies and companies. He has co-authored more than 120 papers in international journals and conferences proposing novel Machine Learning techniques and their application in a variety of problems.

How to cite this article: Ntakaris A, Magris M, Kanniainen J, Gabbouj M, Iosifidis A. Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting*. 2018;37:852–866. https://doi.org/10.1002/for.2543

PUBLICATION 2

Mid-Price Prediction Based on Machine Learning Methods with Technical and Quantitative Indicators

A. Ntakaris, J. Kanniainen, M. Gabbouj and A. Iosifidis

PLOS ONE (under review)

Mid-price Prediction Based on Machine Learning Methods with Technical and Quantitative Indicators

Adamantios Ntakaris[a,*], Juho Kanniainen[a], Moncef Gabbouj[a], Alexandros Iosifidis[b]

[a] *Faculty of Information Technology and Communication Sciences, Tampere University, FI-33720, Tampere, Finland*
[b] *Department of Engineering, Electrical and Computer Engineering, Aarhus University, 8000, Aarhus, Denmark*

Abstract

Stock price prediction is a challenging task, but machine learning methods have recently been used successfully for this purpose. In this paper, we extract over 270 hand-crafted features (factors) inspired by technical and quantitative analysis and tested their validity on short-term mid-price movement prediction. We focus on a wrapper feature selection method using entropy, least-mean squares, and linear discriminant analysis. We also build a new quantitative feature based on adaptive logistic regression for online learning, which is constantly selected first among the majority of the proposed feature selection methods. This study examines the best combination of features using high frequency limit order book data from Nasdaq Nordic. Our results suggest that sorting methods and classifiers can be used in such a way that one can reach the best performance with a combination of only very few advanced hand-crafted features.

Keywords: high-frequency trading, mid-price, machine learning, technical analysis, quantitative analysis

1. Introduction

The problem under consideration in this paper is the prediction of a stock's mid-price movement during high-frequency financial trading. At a given time instance, the mid-price of a stock is defined as the average of the best ask and bid prices. We consider the mid-price as vital information for market makers who continuously balance inventories as well as for traders who need to be able to predict market movements in the correct direction to make money. Moreover, the mid-price facilitates the process of monitoring the markets' stability (i.e. spoofing identification).

Over the past few years, several methods, such as those described in [18], [33], [68], [81], [87], [89], and [90], have been proposed for analyzing stock market data. All these methods follow the standard classification pipeline formed by two processing steps. Given a time instance during the trading process, the state of the market is described based on a (usually short) time window preceding the current instance. A set of hand-crafted features is selected to describe the dynamics of the market, leading to a vector representation. Based on such a representation, a classifier is then employed to predict the state of the market at a time instance within a prediction horizon, as illustrated in Fig.1.

The majority of the studies as we see Section 2 utilize a very limited amount of features without providing any motivation why they selected them, while here we cover the majority of: i) the technical indicators, ii) state-of-the-art limit order book (LOB) features, and iii) quantitative indicators. A work which also utilizes these sets of features can be found in [65]. Additionally, we propose a new advance quantitative feature that is selected first among several feature selection mechanisms for the task of mid-price movement prediction. The use of different hand-crafted features leads to encoding different properties of the financial time-series, and excluding some of these features can result in failing to exploit

*Corresponding author
Email address: adamantios.ntakaris@tuni.fi (Adamantios Ntakaris)

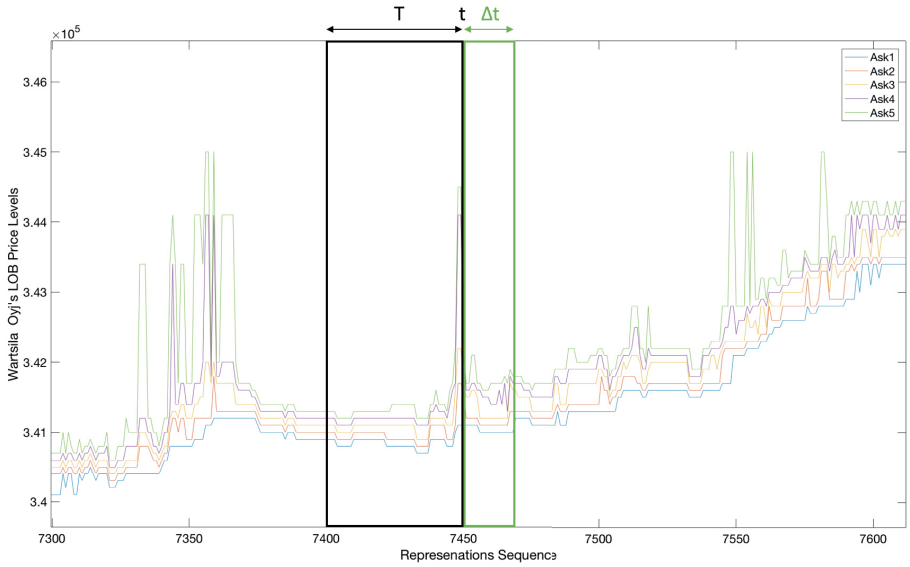


Figure 1: The concept of mid-price prediction can be described as follows: at a given time instance t, the state of the stock is encoded in a vector-based representation calculated using a multi-dimensional time series information from a short-term time window of length T. Given this representation, the direction of the mid-price is predicted at a horizon of Δt.

the relevant information. The definition of a good set of features is directly connected to the performance of the subsequent analysis, since any discarded information at this stage cannot be recovered later by the classifier.

One of the most widely followed approaches used to address this problem is using feature selection methods (e.g., [15], [57]) which can be performed in a wrapper fashion using various types of criteria for feature ranking. While the use of transformation-based dimensionality reduction techniques such as principal component analysis or linear discriminant analysis can lead to a similar processing pipeline, in this paper, we are interested in defining the set of features that convey most of the information in the data. The use of feature selection using unsupervised criteria, and in particular, the maximum entropy criterion, has been used in [5] and [52]. The motivation behind this approach is the fact that as the entropy of a feature increases (when it is calculated in a set of data), the data variance and, thus, the information it encodes, also increases. However, the combination of many high-entropy features in a vector-based representation does not necessarily lead to good classification performance. This is because different dimensions of the adopted data representation need to encode different information.

In this paper, we provide an extensive analysis of various hand-crafted features (273 in total) for mid-price movement prediction. We base our analysis on a wrapped-based feature selection method (i.e., [48]) that exploits unsupervised and supervised criteria for feature ranking. More specifically, we use maximum entropy (i.e., [5], [52]), maximum class discrimination based on linear discriminant analysis (LDA) [84], and regression-based classification [16]. These different realizations of the feature selection method are applied to a wide pool of hand-crafted features. The list of hand-crafted features used in our study is selected to cover both basic and advanced features from two different trading approaches, which means those focusing on technical and quantitative analyses. Technical analysis is based on the fact that price prediction can be achieved by monitoring price and volume charts, while quantitative analysis focuses on statistical models and parameter estimation.

For the technical indicators, we calculate basic and advanced features accompanied by digital filters,

while for the quantitative indicators, we primarily focus on time series analysis and online machine learning. We provide the full feature list and description and use it as input in twelve feature selection models (each corresponding to a different criterion and classifier combination) for our classification task. We not only present the best subset combination of these two types of features but also make a clear comparison of the two trading styles of feature tanks in terms of F1 performance (i.e. F1 score is a test to measure performance and is calculated as the harmonic mean of precision and recall). To the best of our knowledge this is the first study to define which types of information needs to be used for high-frequency time series description and classification.

The main contribution of our work lies on three pillars. The first pillar refers to the utilization of the majority of the technical indicator for the first time in the literature in the high-frequency trading universe. The second pillar refers to development of new quantitative feature, named adaptive logistic regression feature, which selected first among several feature selection metrics. The third pillar, finally, refers to a fair and extent evaluation of these three feature sets (i.e., technical, quantitative, and LOB indicators) via the conversion of entropy, LDA, and LMS as feature selection criteria. This evaluation utilizes LMS, LDA, and radial basis function network (RBFN) as classifiers for the task of mid-price movememt prediction task. Our findings suggest that the best performance is reached by utilizing only very few, advanced, features derived from both quantitative and technical hand-crafted feature sets.

The remainder of the paper is organized as follows. We provide a comprehensive literature review in Section 2. The problem statement and data description are provided in Section 3. The list of hand-crafted features follows in Section 4. In Section 5, we describe the various realizations of the wrapper method adopted in our analysis, while Section 6 provides empirical results, and Section 7 concludes the paper. A detailed description of all features used in our experiments, as well as all ranking lists for each method, are provided in the Appendix section.

2. Related Literature

The rise of algorithmic trading, a type of trading requiring the use of computers under specific rules that can rapidly perform accurate calculations, suggests signal and statistical analyses. Several tools are based on these two types of analysis that a machine learning (ML) trader can utilize to select the best trade. However, which indicator or indicators (i.e. features) should be considered for a ML trader to secure a profitable move? Do historical and present prices contain all the relevant information? Finding answers to these questions is challenging due to the use of technical and quantitative analysis. The former category suggests that there is hidden information and patterns that can be extracted from historical data, whereas the latter suggests that statistical models and probabilities can provide relevant information to an ML trader.

Technical analysis (i.e., [61]) has traditionally received less academic scrutiny than quantitative analysis. Nevertheless, several studies employ technical indicators as the main mechanism for signal analysis and price prediction. In the sphere of HFT, authors in [78] utilize seven trading rule families as a measure of the impact of trading speed, while in [44] a fuzzy momentum analysis based on technical indicators for high speed trading is presented. Grammatical evolution is used in the E-mini S&P 500 index futures market along with technical indicators for entry and exit trading exploration in [32]. In [53] authors provide an extensive investigation of charting analysis of nonparametric kernel regression for Nasdaq stocks via an automated strategy. A decision support system based on artificial neural networks (ANN) where six basic technical indicators are used as input features for signal generation is utilized in [18]. An adaptive neuro fuzzy inference system (ANFIS) is used in [43] for the FOREX market where technical indicators are utilized to benchmark ANFIS performance. Technical indicators are the basis for works in [10], [50], [74], and [86] for passive trading strategies (i.e. buy-and-hold) and stop-loss/stop-gain strategies. A list of ten technical indicators are utilized in [69] as input features for several ML algorithms (i.e. ANN, SVM, random forest, and Naive Bayes) to predict stock trends. The interested reader can also find the implementation of ML methods with technical indicators in [19], [20], [47], [66], [92], and [97]. Technical indicators are also used by [2] for equity premium prediction in the US market, where they have been proved to be efficient in the out-of-sample period (1966 - 2014). For

the German bond market, authors in [4] extend the judgemental bootstrapping domain for the technical analysts' case. The authors prove that technical analysts can be as profitable as the statistical models of experts, by using only a subset of technical indicators.

However, there is also quantitative analysis, which involves ML traders using complex mathematics and statistics as indicators when making trading decisions. Quantitative finance is a broad field that varies from topics like portfolio optimization (e.g., [3], [12], [38], [55], [56], [70]) and asset pricing (e.g., [29], [31], [42], [51], [58], [75], [79]) risk management (e.g., [21], [23], [36], [82]), and time series analysis (e.g., [7], [9], [39], [85]). In this work, we focus on time series analysis and use ideas from financial quantitative time series analysis that have been adjusted to ML. For example, authors in [80] use SVMs and decision trees via correlation analysis for stock market prediction. Another aspect of quantitative analysis is building trading strategies such as mean-reversion (i.e., [71]). A simplistic example of this trading strategy is when a ML trader calculates Bollinger bands to spot trading signals and test a hypothesis. Furthermore, a financial time series is used for entry and exit signal exploration generated by Bollinger bands as described in [54]. A time series analysis should also be tested for cointegration as suggested in [26]. An additional aspect of quantitative analysis is the calculation of order book imbalance for order imbalance strategies. This idea is used as a feature in a deep neural network in [81].

In the present work, we focus on extracted hand-crafted features based on technical and quantitative analysis. We show that a combination of features derived from these groups is able to improve forecasting ability. A combined method is employed by [30] for asset returns predicatibility based on technical indicators and time series models. To the best of our knowledge this is the first attempt at a comparison between these trading schools using several feature selection methods in a wrapper fashion in the HFT literature.

3. Problem Statement

HF-type trading requires the constant analysis of market dynamics. One way to formulate these dynamics is constructing a limit order book (LOB), as illustrated in Table 2. LOB is the cumulative order flow representing valid limit orders that are not executed nor cancelled, which are listed in the so-called message list, as illustrated in Table 1. LOBs are multi-dimensional signals described by stochastic processes, and their dynamics are described as càdlàg functions(i.e., [33]). Functions are formulated for a specific limit order (i.e. an order with specific characteristics in terms of price and volume at a specific time t), as: $order = (t, Price_t, Volume_t)$ that becomes active at time t holds that: $order \in \mathcal{L}(t), order \notin lim_{order' \uparrow order_x} \mathcal{L}(order')$.

Timestamp	Id	Price	Quantity	Event	Side
1275377039033	1372349	341100	300	Submission	Bid
1275377039033	1372349	341100	300	Cancellation	Bid
1275377039037	1370659	343700	100	Submission	Ask
1275377039037	1370659	343700	100	Cancellation	Ask
1275377039037	1372352	341700	150	Submission	Bid
1275377039037	1372352	341700	150	Cancellation	Bid

Table 1: Message list example. A sample from Wartsila Oyj on 01 June 2010.

Depending on how the LOB is constructed, we treat the new information according to event arrivals. The objective of our work is to predict the direction (i.e. up, down, and stationary condition) of the mid-price (i.e. $(p_a + p_b)/2$, where p_a is the ask price and p_b is the bid price at the first level of LOB). The goal is to utilize informative features based on the order flow (i.e. message list or message book [MB]) and LOB, which will help an ML trader improve the accuracy of mid-price movement prediction.

4. Feature Pool

LOB and MB are the sources that we utilize for feature extraction. We provide the complete list of the features that have been explored in the literature for technical and quantitative trading in Table 3. The

	Level 1				...
	Ask		Bid		
Timestamp	Price	Quantity	Price	Quantity	
1127537703903	343600	100	342300	485	...
1275377039033	343600	100	342300	485	...
1275377039037	343600	200	342300	485	...
1275377039037	343600	200	342300	485	...
1275377039037	343600	200	342300	485	...
1275377039037	343600	200	342300	485	...
1275377039037	343600	200	342300	485	...
1275377039037	343600	200	342300	485	...

Table 2: Order book example. A sample from Wartsila Oyj on 01 June 2010.

motivation for choosing the suggested list of features is based on an examination of all the basic and advanced features from technical analysis and comparisons with advanced statistical models, such as adaptive logistic regression for online learning. The present research has identified a gap in the existing literature concerning the performance of technical indicators and comparisons with quantitative models. The present work sets the groundwork for every future work in this direction since it provides insight into the features that are likely to achieve a high rank on the ordering list in terms of predictability power. To this end, we divide our feature set into three main groups. The first group of features is extracted according to [46] and [64]. This group of features aims to capture the dynamics of LOB. This is possible if we consider the actual raw LOB data and relative intensities of different look-back periods of the trade types (i.e. order placement, execution, and cancellation). The second group of features is based on technical analysis. The suggested list describes most of the existing technical indicators (basic and advanced). Technical indicators might help traders spot hidden trends and patterns in their time series. The third group is based on quantitative analysis, which is mainly based on statistical models; it can provide statistics that are hidden in the data. This can be verified by the ranking process, where the advanced online feature (i.e. adaptive logistic regression) is placed first in most of the feature selection metrics (i.e. four out of five feature lists). The suggested features are fully described in Appendix A apart from the description of the newly introduced adapative logistic regresion feature which follows.

4.1. Adaptive Logistic Regression

We build a logistic regression model that we use as a feature in our experimental protocol. Motivation for this model is [63] and [81] where the focal point is the local behavior of LOB levels. We extend this idea by doing online learning with an adaptive learning rate. More specifically, we use the Hessian matrix as our adaptive rate. We also report the ratio of the logistic coefficients based on the relationship of the LOB levels close to the best LOB level and the ones which are deeper in LOB. Since $0 \leqslant h_\theta(V) \leqslant 1$ and V are the stock volumes for the first best six levels of the LOB, we formulate the model as follows:

$$h_\theta(V) = \frac{1}{1 + e^{-\theta^T V}} \tag{1}$$

be the logistic function (i.e. Hypothesis function) and $\theta^T V = \theta_0 + \sum_{j=1}^{n} \theta_j V_j$. Parameter estimation is considered by calculating the parameters likelihood:

$$L(\theta) = \prod_{i=1}^{m} (h_\theta(V^{(i)}))^{y^{(i)}} (1 - h_\theta(V^{(i)}))^{1-y^{(i)}} \tag{2}$$

for m training samples and the cost function, based on this probabilistic approach, is as follows:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \big[- y^{(i)} log(h_\theta(V^{(i)})) - (1 - y^{(i)}) log(1 - h_\theta(V^{(i)})) \big]. \tag{3}$$

Feature Sets	Description
First group: Basic	n levels of LOB Data
Time-Insensitive	Spread & Mid-Price
	Price Differences
	Price & Volume Means
	Accumulated Differences
Time-Sensitive	Price & Volume Derivation
	Average Intensity per Type
	Relative Intensity Comparison
	Limit Activity Acceleration
Second group: Technical Analysis	
	Accumulation Distribution Line
	Awesome Oscillator
	Accelerator Oscillator
	Average Directional Index
	Average Directional Movement Index Rating
	Displaced Moving Average based on Williams Alligator Indicator
	Absolute Price Oscillator
	Aroon Indicator
	Aroon Oscillator
	Average True Range
	Bollinger Bands
	Ichimoku Clouds
	Chande Momentum Oscillator
	Chaikin Oscillator
	Chandelier Exit
	Center of Gravity Oscillator
	Donchian Channels
	Double Exponential Moving Average
	Detrended Price Oscillator
	Heikin-Ashi
	Highest High and Lowest Low
	Hull MA
	Internal Bar Strength
	Keltner Channels
	Moving Average Convergence/Divergence Oscillator
	Median Price
	Momentum
	Variable Moving Average
	Normalized Average True Range
	Percentage Price Oscillator
	Rate of Change
	Relative Strength Index
	Parabolic Stop and Reverse
	Standard Deviation
	Stochastic Relative Strength Index
	T3-Triple Exponential Moving Average
	Triple Exponential Moving Average
	Triangular Moving Average
	True Strength Index
	Ultimate Oscillator
	Weighted Close
	Williams %R
	Zero-Lag Exponential Moving Average
	Fractals
	Linear Regression Line
	Digital Filtering: Rational Transfer Function
	Digital Filtering: Savitzky-Golay Filter
	Digital Filtering: Zero-Phase Filter
	Remove Offset and Detrend
	Beta-like Calculation
Third group: Quantitative Analysis	
	Autocorrelation
	Partial Correlation
	Cointegration based on Engle-Granger test
	Order Book Imbalance
	Adaptive Logistic Regression

Table 3: Feature list for the three groups (description of the majority of the hand-crafted is in Appendix A where the description of the newly introduced feature (in bold) can be found in Section 4.1).

The next step is the process of choosing θs for optimizing (i.e. minimizing) J(θ). To do so, we will use Newton's update method:

$$\theta^{(s+1)} = \theta^{(s)} - H^{-1}\nabla_\theta J, \tag{4}$$

where the gradient is: $\nabla_\theta J = \frac{1}{m}\sum_{1}^{m}(h_\theta(V^{(i)}) - y^{(i)})V^{(i)}$ and the Hessian matrix is: $H = \frac{1}{m}\sum_{i=1}^{m}\left[h_\theta(V^{(i)})\big(1 - h_\theta(V^{(i)})\big)V^{(i)}(V^{(i)})^T\right]$ with $V^{(i)}(V^{(i)})^T \in \mathbb{R}^{(n+1)\times(n+1)}$ and $y^{(i)}$ are the labels which are calculated as the differences of the best level's ask (and bid) prices. The suggested labels describe a binary classification problem since we consider two states, one for change in the best ask price and another on for no change in the best ask price.

We perform the above calculation in an online manner. The online process considers the 9^{th} element of every 10 MB block multiplied by the θ coefficient first-order tensor to obtain the probabilistic behavior (we filter the obtained first-order tensor through the hypothesis function) of the 10^{th} event of the 10 MB block. The output is the feature representation expressed as scalar (i.e. probability) of the bid and ask price separately.

5. Wrapper Method of Feature Selection

Feature selection is an area which focuses on applications with multidimensional datasets. An ML trader performs feature selection for three primary reasons: to reduce computational complexity, to improve performance, and to gain a better understanding of the underlying process. Feature selection, as a pre-processing method, can enhance classification power by adding features that contain information relevant to the task at hand. There are two metaheuristic feature selection methods: the wrapper method and the filter (i.e. transformation-based) method. We choose to perform classification based on the wrapper method since it considers the relationship among the features while the filter methods do not.

Our wrapper approach consists of five different feature subset selection criteria based on two linear and one non-linear methods for evaluation (see Algorithm 1 for a general description). More specifically, we convert entropy, least-mean-square (LMS), and linear discriminant analysis (LDA) as feature selection criteria. For the last two cases (i.e., LMS and LDA) we provide two different selection criteria as follows: i) for LMS the first metric follows the $\mathcal{L}_2$-norm and the second metric is a statistical bias measure, and ii) for LDA the first metric is based on the ratio of the *within*-class scatter matrix while the second metric is derived according to the *between*-class scatter matrix. For classification evaluation we utilize LMS, LDA and a radial basis function network (i.e., [11]) as this utilized by [40] and [64]). Our choice to apply these linear and non-linear classifiers is informed by the amount of data in our dataset (details will be provided in the following section). We measure classification performance according to accuracy, precision, recall, and the F1 scores for every possible combination of the hand-crafted features by utilizing LMS, LDA, and RBFN. F1 score is defined as the harmonic average of the recall and precision, where recall is defined as $TP/TP + FN$ and precision is defined as $TP/(TP + FP)$ for TP, FN, and FP be the true positives, false negatives, and false positives, respectively.

5.1. Feature Sorting

We convert sample entropy, LMS, and LDA (for the latter two methods we use two different criteria for feature evaluation) into feature selection methods. A visual representation of the feature sorting method can be seen in Fig. 2.

5.1.1. Feature Sorting with Entropy

We employ entropy [73], a measure of signal complexity where the signal is the time series of the multidimensional two-mode tensor with dimensions $\mathbb{R}^{p\times n}$, and where p is the number of features and n number of samples, as a measure of feature relevance. We calculate the bits of every feature in the

Algorithm 1 Wrapper-Based Feature Selection

procedure $l_{opt} = Feature\_Select(X, labels, criterion)$
  $l = [1 : D]$
  $l_{opt} = [\ ]$
  $X_{opt} = [\ ],\ [D, N] = size(X)$
  for $d = 1 : D$ **do**
    $crit\_list = [\ ]$
    for $i = 1 : D - d + 1$ **do**
      $curr\_X = [X_{opt}; X(i, :)]$
      $crit\_list[i] = crit(curr\_X)$
    end for
    $[best\_d, best\_crit] = opt(crit\_list)$
    $l_{opt}[d] = best\_d$
    $l[best\_d] = [\ ]$
    $X_{opt} = [X_{opt}; X(best\_d, :)]$
    $X(best\_d, :) = [\ ]$
  end for
end procedure

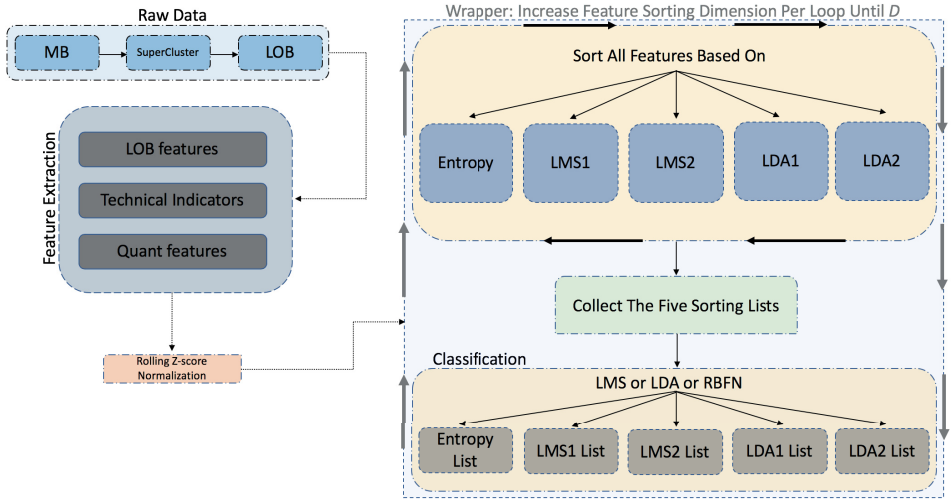


Figure 2: The process of feature sorting and classification is based on the wrapper method. From left to right: 1) Raw data is converted to LOB data via supeclustering, 2) the feature extraction process follows (i.e., three feature sets are extracted), 3) then every feature set is normalized based on z-score in a rolling window basis (see Section 6 for details), 4) Wrapper method: there are two main blocks in this process - the first block refers to the sorting (i.e., five different sorting criteria based on entropy, LMS1, LMS2, LDA1, and LDA2) of the feature sets independently (i.e., LOB features, technical indicators, and quantitative indicators are sorted separately) and all together (i.e., the three feature sets are merged and sorted all together) and the second block refers to the incremental classification (i.e., we increase the dimension of every sorting list during classification by one feature in every loop and average the final f1 scores per sorting list) based on three classifiers (i.e., LMS, LDA, and RBFN).

feature set in an iterative manner and report the order. We measure the entropy as follows: $H(X) =$

$-\sum_{i=1}^{p} p(x_i)\ log\ p(x_i)$, where $p(x_i)$ is the probability of the frequency per feature for the given data samples.

5.1.2. Feature Sorting with Least-Mean-Square

We perform feature selection based on the least-mean square classification rate (LMS1) and $\mathcal{L}_2$-norm (LMS2). LMS is a fitting method which aims to produce an approximation that minimizes the sum of squared differences between given data and predicted values. We use this approach to evaluate the relevance of our hand-crafted features. A hand-crafted feature evaluation is performed sequentially via LMS. More specifically, each of the features is evaluated based on the classification rate, the $\mathcal{L}_2$-norm of the predicted labels, and the ground truth. The evaluation process is performed as follows: $\mathbf{HW} = \mathbf{T}$, where $\mathbf{H} \in \mathbb{R}^{p_i \times n}$ is the input data with feature dimension p_i where it is calculated incrementally for the number of training samples n, $\mathbf{W} \in \mathbb{R}^{p_i \times \#cl}$ are the weighted coefficients for the number of features p_i of the number(#) of classes (i.e. up, down, and stationary labelling), and $\mathbf{T} \in \mathbb{R}^{\#cl \times n}$ represents the target labels of the training set. The weight coefficient matrix $\mathbf{W}$ is estimated via the following formula: $\mathbf{W} = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{H}^{\dagger}$ is the Moore-Penrose pseudoinverse matrix.

5.1.3. Feature Sorting with Linear Discriminant Analysis

Linear discriminant analysis (LDA) can be used for classification and dimensionality reduction. However, instead of performing these two tasks, we convert LDA into a feature selection algorithm. We measure feature selection performance based on two metrics. One is the classification rate (LDA1) and the other is based on the error term (LDA2), which we define as the ratio of the *within*-class scatter matrix and the *between*-class scatter matrix. The main objective of LDA is finding the projection matrix $\mathbf{W} \in \mathbb{R}^{m \times \#cl-1}$, where m is the sample dimension, and $\#cl$ is the number of classes, such that $Y = W^T X$ maximizes the class separation. For the given sample set $X = X_1 \cup X_2 \cup ... \cup X_{\#cl}$, where $X_k = \{x_1^k, ..., x_{\ell_k}^k\}_{k=1,...,\#cl}$ is the class-specific data subsample, we try to find $\mathbf{W}$ that maximizes Fisher's ratio $J(\mathbf{W}) = \dfrac{trace\left(\mathbf{W}^T S_B \mathbf{W}\right)}{trace\left(\mathbf{W}^T S_W \mathbf{W}\right)}$, where $S_B = \sum_{i=1}^{\#cl} N_i(\mu_i - \mu)(\mu_i - \mu)^T$ and $S_W = \sum_{i=1}^{C} \sum_{x \in X_i}^{d} (x - \mu_i)(x - \mu_i)^T$ are the *between*-class and *within*-class scatter matrices, respectively, with $\mu_i = \frac{1}{\ell_k} \sum_{k \in \#cl} X_k$ and $\mu = \frac{1}{m} \sum_{k \in \#cl} \ell_k X_k$. In a similar fashion, we perform calculations for the projected samples $\mathbf{y}$ with $\widetilde{\mu_i} = \frac{1}{\ell_k} \sum_{k \in \#cl} Y_k$ and $\widetilde{\mu} = \frac{1}{m} \sum_{k \in \#cl} \ell_k Y_k$, while the scatter matrices (i.e. *within* and *between* scatter matrices, respectively) are $\widetilde{S_W} = \sum_{i=1}^{\#cl} \sum_{k \in \#cl} (y - \widetilde{\mu_i})(y - \widetilde{\mu_i})^T$ and $\widetilde{S_B} = \sum_{i=1}^{\#cl} \ell_k (\widetilde{\mu_i} - \widetilde{\mu})(\widetilde{\mu_i} - \widetilde{\mu})^T$. The above calculations constitute the basis for the two metrics that we use to evaluate the hand-crafted features incrementally. The two evaluation metrics are based on the classification rate and the ratio of the *within*-class and *between*-class scatter matrices of the projected space Y.

5.2. Classification for Feature Selection

We perfrom classification evaluation based on three classifiers: LMS, LDA, and RBFN. Theory for the first two discussed in Sections 5.1.2 and 5.1.3 while RBFN classifiers is descibed in the following section.

5.2.1. RBFN Classifier

We utilize a SLFN (Fig. 3) as this suggested by [37]. The description and the implememntation can be also found in [64]. This type of model is formed as in Fig. 3.

In order to train fast this model we follow [98], and [41]. We utilize K-means clustering for K prototype vectors identification, which then are used as the network's hidden layer weights. Having identified the network's hidden layer weights $\mathbf{V} \in \mathbb{R}^{D \times K}$, the input data $\mathbf{x}_i$, $i = 1, \ldots, N$ are mapped to vectors $\mathbf{h}_i \in \mathbb{R}^K$ in a non-linear way, expressing the data representations in the feature space determined

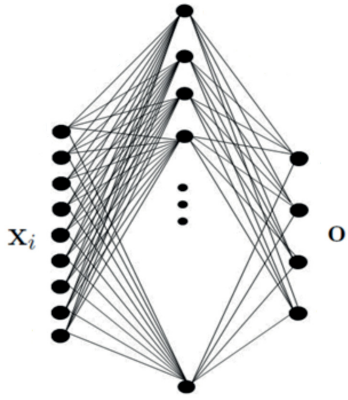


Figure 3: SLFN

by the network's hidden layer outputs $\mathbb{R}^K$. Then radial basis function is utilized, i.e. $\mathbf{h}_i = \phi_{RBF}(\mathbf{x}_i)$, calculated in an element-wise manner, as follows:

$$h_{ik} = \exp\left(\frac{\|\mathbf{x}_i - \mathbf{v}_k\|_2^2}{2\sigma^2}\right), \ k = 1, \ldots, K, \tag{5}$$

where σ is a hyper-parameter denoting the spread of the RBF neuron and $\mathbf{v}_k$ corresponds to the k-th column of $\mathbf{V}$.

The network's output weights $\mathbf{W} \in \mathbb{R}^{K \times C}$ are determined by solving the following equation:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \ \|\mathbf{W}^T\mathbf{H} - \mathbf{T}\|_F^2 + \lambda\|\mathbf{W}\|_F^2, \tag{6}$$

where $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_N]$ is a matrix formed by the network's hidden layer outputs for the training data and $\mathbf{T}$ is a matrix formed by the network's target vectors $\mathbf{t}_i$, $i = 1, \ldots, N$. The network's output weights are given by:

$$\mathbf{W} = \left(\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I}\right)^{-1}\mathbf{H}\mathbf{T}^T. \tag{7}$$

Then a new (test) sample $\mathbf{x} \in \mathbb{R}^D$ is mapped on its corresponding representations in spaces $\mathbb{R}^K$ and $\mathbb{R}^C$, i.e. $\mathbf{h} = \phi_{RBF}(\mathbf{x})$ and $\mathbf{o} = \mathbf{W}^T\mathbf{h}$, respectively. fFinally, the classification task is based on the maximal network output, i.e.:

$$l_{\mathbf{x}} = \arg\max_k \ o_k. \tag{8}$$

6. Results

In this section, we provide details regarding the conducted experiments. The experiments are based on the idea of a mid-price prediction state (i.e. up, down, and stationary) for ITCH feed data in millisecond resolution. For the experimental protocol, we followed the setup in [64], which is based on the anchored cross-validation format. According to this format, we use the first day as training and second day as testing for the first fold, whereas the second fold consists of the previous training and testing periods as a training set, and the next day is always used as a test set. Each of the training and testing sets contains the hand-crafted feature representations for all the five stocks from the FI-2010 dataset. Hence, we obtain a mode-three tensor of dimensions $273 \times 458,125$. The first dimension is the number of features, whereas the second one is the number of sample events. At this point, we must specify that the process

of hand-crafted feature extraction in Section 5 is conducted in the full length of the given information based on MB with 4,581,250 events. The motivation for taking separate blocks of messages of ten events is the time-invariant nature of the data. To keep the ML trader fully informed regarding MB blocks, we use features that convey this information by calculating, among others, averages, regression, risk, and online learning feedback.

The results we present here are the mid-price predictions for the next 10^{th}, 20^{th}, and 30^{th} events (i.e. translated into MB events) or else one, two, and three next events after the current state translated into a feature representations setup. The prediction performance of these events is measured by the accuracy, precision, recall and F1 score, whereas we emphasize the F1 score. We focus on the F1 score because it can only be affected in one direction by skewed distributions for unbalanced classes, as observed in our data. Performance metrics are calculated against the mid-price labelling calculation of ground truth extraction. More specifically, we extract labels based on the percentage change of the smoothed mid-price with a span window of 9, for our supervised learning methods, by calculating it as follows: $L_1 = \frac{MP_{next} - MP_{curr}}{MP_{curr}}$, where MP_{curr} is the current mid-price, and MP_{next} is the next mid-price. We threshold the percentage change identification by a fixed number $\gamma = 0.002$ and perform a rolling z-score normalization on our dataset in order to avoid look-ahead bias[1]. The rolling window z-score normalization is based on the anchored cross validation setup which means that the normalization of the training set is totally unaffected from any fututre information.

We report our results in Table 4 - Table 8 of the best feature selection list for the five sorting lists (i.e. based on entropy, LMS1, LMS2, LDA1, and LDA2) according to the supervised linear and non-linear classifiers of LMS, LDA, and RBFN. For the last classifier (i.e. RBFN), we use a multilayer perceptron based on the extreme learning machine model with a twist in the initialization process of weights calculation based on the k-means algorithm. The full description of this method can be found in [64]. We provide results based on the whole feature pool (see Table 4), the first feature pool according to [46] and [64] (see Table 5), based only on technical indicators (see Table 6) and quantitative indicators (see Table 7). More specifically, for the first feature pool, we have 135 features, while for the second pool we have 83 features, and for the last pool we have 55 features; in total, we have 273 features. The number of best features that used in the above methods is different in every case and can be monitored in Fig.2 and Fig.3. We should point out that we tested all the possible combinations for the five sorting methods and the three classifiers (i.e., 15 different cases) but we report results that had some variations. For instance, in Table 6 we report results for entropy as sorting method and LMS together with RBFN as classifiers but not with LDA (as classifier) since the last method reports similar results.

[1]Look ahead bias refers to the process that future information is injected to the training set.

Sorting	Classifier	T	*Accuracy*	*Precision*	*Recall*	*F1*
Entropy	LMS	10	0.529 ± 0.059	0.447 ± 0.007	0.477 ± 0.013	0.440 ± 0.018
LMS1	LMS	10	0.540 ± 0.059	0.437 ± 0.007	0.456 ± 0.013	0.430 ± 0.018
LMS2	LMS	10	0.538 ± 0.052	0.447 ± 0.005	0.478 ± 0.013	0.444 ± 0.011
LDA1	LDA	10	0.616 ± 0.048	0.408 ± 0.019	0.398 ± 0.011	0.397 ± 0.015
LDA2	LDA	10	0.543 ± 0.057	0.430 ± 0.010	0.455 ± 0.017	0.429 ± 0.018
LDA1	LMS	10	0.604 ± 0.068	0.468 ± 0.035	0.431 ± 0.042	0.408 ± 0.035
LDA2	LMS	10	0.522 ± 0.026	0.441 ± 0.020	0.473 ± 0.007	0.435 ± 0.007
Entropy	RBFN	10	0.474 ± 0.046	0.420 ± 0.031	0.445 ± 0.039	0.400 ± 0.039
LMS1	RBFN	10	0.600 ± 0.045	0.436 ± 0.019	0.425 ± 0.021	0.417 ± 0.019
LMS2	RBFN	10	0.537 ± 0.016	0.442 ± 0.011	0.470 ± 0.016	0.439 ± 0.012
LDA1	RBFN	10	0.585 ± 0.061	0.443 ± 0.018	0.438 ± 0.037	0.419 ± 0.026
LDA2	RBFN	10	0.528 ± 0.029	0.438 ± 0.020	0.467 ± 0.010	0.434 ± 0.017
Entropy	LMS	20	0.503 ± 0.049	0.469 ± 0.008	0.482 ± 0.014	0.462 ± 0.017
LMS1	LMS	20	0.503 ± 0.049	0.470 ± 0.008	0.482 ± 0.014	0.462 ± 0.017
LMS2	LMS	20	0.503 ± 0.049	0.469 ± 0.008	0.481 ± 0.014	0.462 ± 0.018
LDA1	LDA	20	0.478 ± 0.060	0.400 ± 0.038	0.404 ± 0.041	0.393 ± 0.018
LDA2	LDA	20	0.505 ± 0.046	0.452 ± 0.009	0.461 ± 0.012	0.450 ± 0.012
LDA1	LMS	20	0.530 ± 0.032	0.457 ± 0.024	0.426 ± 0.048	0.401 ± 0.048
LDA2	LMS	20	0.499 ± 0.019	0.462 ± 0.019	0.476 ± 0.007	0.457 ± 0.015
Entropy	RBFN	20	0.464 ± 0.038	0.436 ± 0.033	0.448 ± 0.035	0.425 ± 0.036
LMS1	RBFN	20	0.519 ± 0.023	0.430 ± 0.016	0.417 ± 0.022	0.412 ± 0.027
LMS2	RBFN	20	0.508 ± 0.010	0.456 ± 0.015	0.466 ± 0.018	0.454 ± 0.017
LDA1	RBFN	20	0.523 ± 0.025	0.441 ± 0.024	0.429 ± 0.041	0.416 ± 0.046
LDA2	RBFN	20	0.502 ± 0.018	0.454 ± 0.019	0.465 ± 0.009	0.452 ± 0.015
Entropy	LMS	30	0.503 ± 0.042	0.475 ± 0.013	0.484 ± 0.014	0.470 ± 0.019
LMS1	LMS	30	0.503 ± 0.042	0.475 ± 0.013	0.484 ± 0.014	0.470 ± 0.019
LMS2	LMS	30	0.503 ± 0.043	0.474 ± 0.012	0.482 ± 0.014	0.461 ± 0.019
LDA1	LDA	30	0.464 ± 0.048	0.414 ± 0.025	0.420 ± 0.027	0.403 ± 0.018
LDA2	LDA	30	0.500 ± 0.043	0.457 ± 0.012	0.464 ± 0.013	0.455 ± 0.014
LDA1	LMS	30	0.489 ± 0.018	0.451 ± 0.030	0.429 ± 0.051	0.405 ± 0.072
LDA2	LMS	30	0.496 ± 0.016	0.476 ± 0.018	0.479 ± 0.009	0.472 ± 0.015
Entropy	RBFN	30	0.464 ± 0.035	0.446 ± 0.035	0.449 ± 0.034	0.440 ± 0.037
LMS1	RBFN	30	0.471 ± 0.018	0.425 ± 0.018	0.414 ± 0.020	0.409 ± 0.026
LMS2	RBFN	30	0.494 ± 0.014	0.464 ± 0.021	0.466 ± 0.021	0.461 ± 0.024
LDA1	RBFN	30	0.481 ± 0.022	0.438 ± 0.034	0.428 ± 0.045	0.415 ± 0.057
LDA2	RBFN	30	0.493 ± 0.017	0.465 ± 0.018	0.467 ± 0.010	0.463 ± 0.016

Table 4: F1-macro (i.e. F1-macro = $\frac{1}{C}\sum_{k\in C} F1_k$, with C as the number of classes for the 9-fold experimental protocol) results, based on the total feature pool, for the five sorting lists classified per LMS, LDA, and RBFN for the next $T = 10^{th}$, 20^{th}, and 30^{th} events, respectively, as the predicted horizon. The number of best features used in the above methods is different in every case (as seen in Fig. 3).

Sorting	Classifier	T	*Accuracy*	*Precision*	*Recall*	*F1*
Entropy	LMS	10	0.420 ± 0.025	0.379 ± 0.011	0.397 ± 0.011	0.355 ± 0.013
LMS1	LMS	10	0.574 ± 0.055	0.402 ± 0.013	0.396 ± 0.018	0.384 ± 0.020
LMS2	LMS	10	0.519 ± 0.015	0.400 ± 0.009	0.413 ± 0.009	0.396 ± 0.010
LDA1	LDA	10	0.561 ± 0.090	0.389 ± 0.018	0.382 ± 0.016	0.363 ± 0.030
LDA2	LDA	10	0.507 ± 0.041	0.373 ± 0.014	0.384 ± 0.017	0.362 ± 0.019
Entropy	LMS	20	0.386 ± 0.018	0.386 ± 0.015	0.397 ± 0.015	0.363 ± 0.016
LMS1	LMS	20	0.527 ± 0.027	0.411 ± 0.013	0.389 ± 0.015	0.375 ± 0.029
LMS2	LMS	20	0.462 ± 0.013	0.405 ± 0.013	0.410 ± 0.009	0.400 ± 0.012
LDA1	LDA	20	0.529 ± 0.031	0.406 ± 0.017	0.381 ± 0.011	0.360 ± 0.024
LDA2	LDA	20	0.461 ± 0.036	0.378 ± 0.016	0.380 ± 0.016	0.368 ± 0.021
Entropy	LMS	30	0.391 ± 0.016	0.395 ± 0.018	0.401 ± 0.015	0.380 ± 0.017
LMS1	LMS	30	0.459 ± 0.025	0.405 ± 0.017	0.388 ± 0.020	0.366 ± 0.040
LMS2	LMS	30	0.432 ± 0.009	0.407 ± 0.015	0.409 ± 0.013	0.401 ± 0.016
LDA1	LDA	30	0.447 ± 0.041	0.391 ± 0.018	0.377 ± 0.017	0.352 ± 0.037
LDA2	LDA	30	0.418 ± 0.028	0.373 ± 0.016	0.375 ± 0.015	0.361 ± 0.018

Table 5: F1-macro (i.e. F1-macro = $\frac{1}{C}\sum_{k\in C} F1_k$, with C as the number of classes for the 9-fold experimental protocol) results, based only on the hand-crafted features from [64], for the five sorting lists classified based on LMS, LDA, and RBFN for the next $T = 10^{th}$, 20^{th}, and 30^{th} events, respectively, as the predicted horizon. The number of best features used in the above methods is different in every case (as seen in Fig. 3).

Sorting	Classifier	T	*Accuracy*	*Precision*	*Recall*	*F1*
Entropy	LMS	10	0.456 ± 0.038	0.372 ± 0.021	0.380 ± 0.014	0.353 ± 0.020
LMS1	LMS	10	0.497 ± 0.066	0.371 ± 0.017	0.377 ± 0.021	0.354 ± 0.024
LMS2	LMS	10	0.460 ± 0.016	0.383 ± 0.012	0.394 ± 0.009	0.365 ± 0.010
LDA1	LDA	10	0.517 ± 0.064	0.367 ± 0.015	0.371 ± 0.020	0.344 ± 0.026
LDA2	LDA	10	0.475 ± 0.023	0.371 ± 0.010	0.382 ± 0.009	0.351 ± 0.015
Entropy	LMS	20	0.430 ± 0.029	0.384 ± 0.025	0.387 ± 0.017	0.371 ± 0.023
LMS1	LMS	20	0.480 ± 0.033	0.384 ± 0.023	0.381 ± 0.021	0.364 ± 0.037
LMS2	LMS	20	0.452 ± 0.011	0.400 ± 0.018	0.402 ± 0.011	0.391 ± 0.015
LDA1	LDA	20	0.483 ± 0.034	0.379 ± 0.022	0.377 ± 0.020	0.355 ± 0.038
LDA2	LDA	20	0.453 ± 0.014	0.382 ± 0.015	0.387 ± 0.009	0.369 ± 0.016
Entropy	LMS	30	0.423 ± 0.030	0.394 ± 0.028	0.394 ± 0.020	0.385 ± 0.027
LMS1	LMS	30	0.450 ± 0.018	0.395 ± 0.028	0.393 ± 0.027	0.379 ± 0.050
LMS2	LMS	30	0.446 ± 0.013	0.409 ± 0.020	0.408 ± 0.013	0.403 ± 0.019
LDA1	LDA	30	0.430 ± 0.041	0.384 ± 0.027	0.382 ± 0.027	0.353 ± 0.053
LDA2	LDA	30	0.433 ± 0.021	0.397 ± 0.017	0.396 ± 0.016	0.386 ± 0.026

Table 6: F1-macro (i.e. F1-macro = $\frac{1}{C}\sum_{k\in C} F1_k$, with C as the number of classes for the 9-fold experimental protocol) results, based only on technical features, for the five sorting lists classified based on LMS,LDA, and RBFN for the next $T = 10^{th}$, 20^{th}, and 30^{th} events, respectively, as the predicted horizon. The number of best features used in the above methods is different in every case (as seen in Fig. 3).

Sorting	Classifier	T	*Accuracy*	*Precision*	*Recall*	*F1*
Entropy	LMS	10	0.393 ± 0.109	0.399 ± 0.047	0.419 ± 0.047	0.340 ± 0.064
LMS1	LMS	10	0.665 ± 0.033	0.468 ± 0.043	0.388 ± 0.016	0.366 ± 0.016
LMS2	LMS	10	0.571 ± 0.071	0.470 ± 0.053	0.418 ± 0.032	0.384 ± 0.020
LDA1	LDA	10	0.611 ± 0.088	0.422 ± 0.039	0.390 ± 0.020	0.370 ± 0.024
LDA2	LDA	10	0.380 ± 0.101	0.401 ± 0.024	0.428 ± 0.027	0.339 ± 0.063
Entropy	LMS	20	0.400 ± 0.074	0.408 ± 0.048	0.422 ± 0.048	0.372 ± 0.061
LMS1	LMS	20	0.553 ± 0.029	0.429 ± 0.037	0.373 ± 0.016	0.335 ± 0.025
LMS2	LMS	20	0.483 ± 0.017	0.447 ± 0.022	0.457 ± 0.025	0.435 ± 0.029
LDA1	LDA	20	0.513 ± 0.072	0.402 ± 0.032	0.379 ± 0.026	0.347 ± 0.040
LDA2	LDA	20	0.424 ± 0.073	0.431 ± 0.026	0.444 ± 0.023	0.340 ± 0.053
Entropy	LMS	30	0.410 ± 0.062	0.416 ± 0.051	0.423 ± 0.048	0.391 ± 0.060
LMS1	LMS	30	0.478 ± 0.022	0.407 ± 0.038	0.370 ± 0.019	0.320 ± 0.036
LMS2	LMS	30	0.481 ± 0.012	0.457 ± 0.026	0.460 ± 0.024	0.449 ± 0.034
LDA1	LDA	30	0.464 ± 0.037	0.394 ± 0.030	0.378 ± 0.027	0.338 ± 0.049
LDA2	LDA	30	0.425 ± 0.063	0.437 ± 0.029	0.443 ± 0.022	0.406 ± 0.055

Table 7: F1-macro (i.e. F1-macro = $\frac{1}{C}\sum_{k\in C} F1_k$, with C as the number of classes for the 9-fold experimental protocol) results, based only on quantitative features, for the five sorting lists classified based on LMS,LDA, and RBFN for the next $T = 10^{th}$, 20^{th}, and 30^{th} events, respectively, as the predicted horizon. The number of best features used in the above methods is different in every case (as seen in Fig. 3).

Sorting	Classifier	5	50	100	200	273
Entropy	LMS	0.319 ± 0.008	0.363 ± 0.027	0.414 ± 0.020	0.425 ± 0.025	0.440 ± 0.018
LMS1	LMS	0.377 ± 0.008	0.374 ± 0.036	0.393 ± 0.047	0.419 ± 0.036	0.440 ± 0.018
LMS2	LMS	0.402 ± 0.015	0.443 ± 0.014	0.441 ± 0.018	0.440 ± 0.018	0.440 ± 0.018
LDA1	LMS	0.373 ± 0.013	0.380 ± 0.017	0.395 ± 0.016	0.315 ± 0.018	0.289 ± 0.025
LDA2	LMS	0.412 ± 0.011	0.420 ± 0.017	0.420 ± 0.019	0.289 ± 0.013	0.309 ± 0.027
LDA1	LMS	0.370 ± 0.011	0.372 ± 0.032	0.387 ± 0.041	0.440 ± 0.017	0.440 ± 0.018
LDA2	LMS	0.421 ± 0.010	0.435 ± 0.011	0.435 ± 0.014	0.440 ± 0.017	0.441 ± 0.018
Entropy	RBFN	0.316 ± 0.010	0.363 ± 0.020	0.413 ± 0.016	0.430 ± 0.016	0.441 ± 0.016
LMS1	RBFN	0.387 ± 0.018	0.402 ± 0.022	0.421 ± 0.017	0.429 ± 0.017	0.441 ± 0.016
LMS2	RBFN	0.403 ± 0.015	0.444 ± 0.012	0.439 ± 0.013	0.439 ± 0.019	0.442 ± 0.016
LDA1	RBFN	0.371 ± 0.011	0.387 ± 0.021	0.411 ± 0.013	0.440 ± 0.016	0.441 ± 0.016
LDA2	RBFN	0.416 ± 0.011	0.434 ± 0.015	0.436 ± 0.015	0.436 ± 0.016	0.442 ± 0.016

Table 8: F1 results based on different numbers of best features for the five criteria of the wrapper-based feature selection methods.

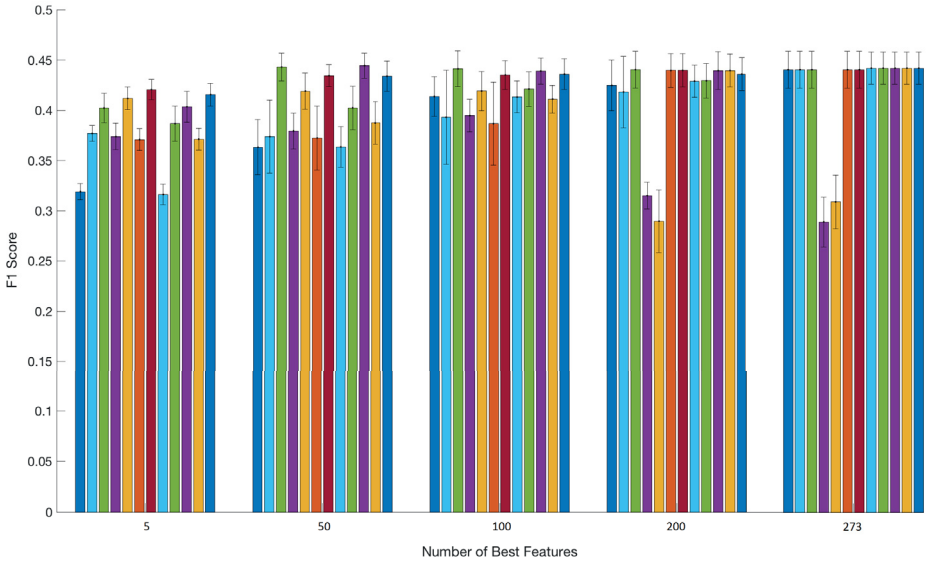


Figure 4: Bar plots with variance present the average (i.e. average F1 performance for the 9-fold protocol for all the features) F1 score of the 12 different models for the cases of 5, 50, 100, 200, and 273 number of best features. The order of the models from the left to the right column is (1) feature list sorted based on entropy and classified based on LMS, (2) feature list sorted based on LMS1 and classified based on LMS, (3) feature list sorted based on LMS2 and classified based on LMS, (4) feature list sorted based on LDA1 and classified based on LDA, (5) feature list sorted based on LDA2 and classified based on LDA, (6) feature list sorted based on LDA1 and classified based on LMS, (7) feature list sorted based on LDA2 and classified based on LMS, (8) feature list sorted based on LDA2 and classified based on LMS, (9) feature list sorted based on entropy and classified based on RBFN, (10) feature list sorted based on LMS2 and classified based on RBFN, (11) feature list sorted based on LDA1 and classified based on RBFN, and (12) feature list sorted based on LDA2 and classified based on RBFN.

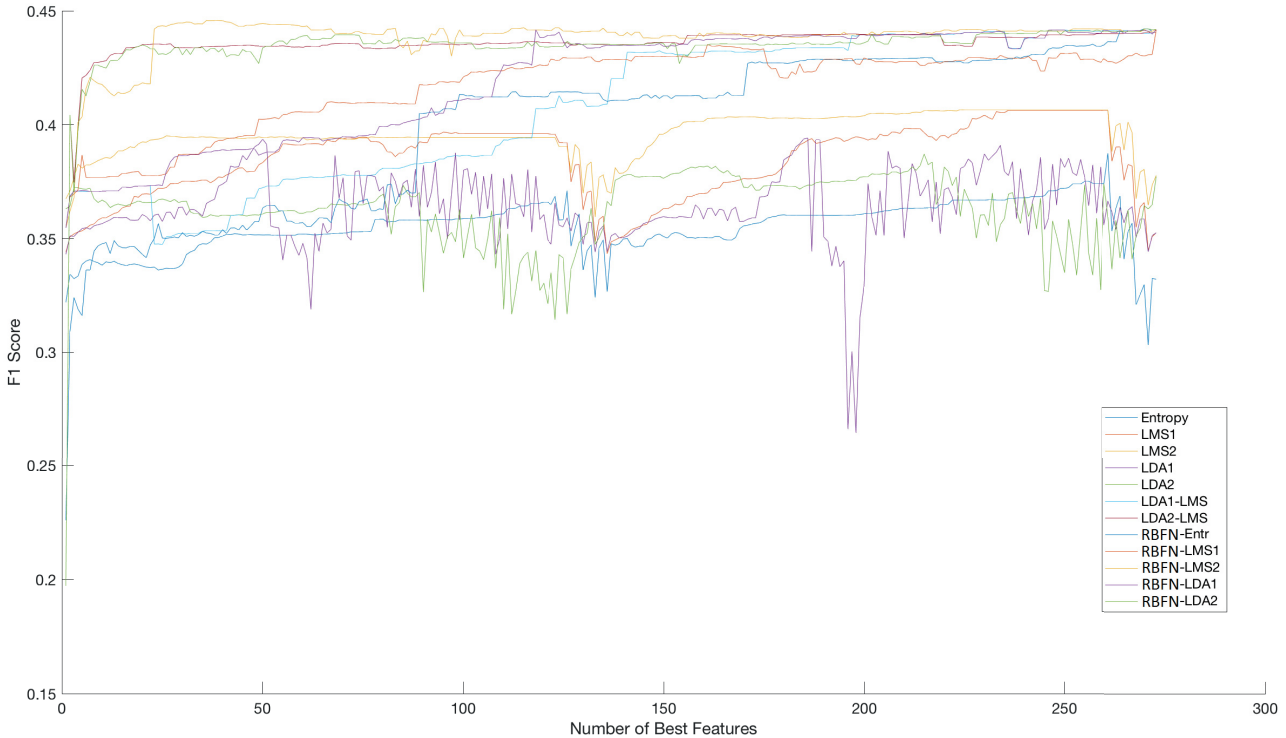


Figure 5: F1 performance per number of best features sequence for 10 events as the projected horizon, where lines represent the 5 different sorting methods as classified based on LMS, LDA, and RBFN.

There is a dual interpretation of the suggested feature lists and wrapper method results. Regarding the feature lists, we have five different feature sorting methods starting from entropy, to LMS1 and LMS2 and continue to LDA1 and LDA2. More specifically, results based on the entropy sorting method reveal that the first 20 places are covered by features almost entirely from technical indicators (i.e. 19 out of 20 places and only one from the first basic group), while the first 100 best places are covered by 36 quant features, 48 technical features, and 16 from the first basic group.

For the LMS case, we present two sorting lists where we use two different criteria for the final feature selection. In the LMS1 case, the first best 20 places covered by features are derived mainly from quantitative analysis (11 out of 20), 7 from the first basic group, and only 2 from the technical group. The first place is covered by a very advanced feature based on the logistic regression model for online learning. For the same method, the first 100 best places covered by 25 quant features, 18 features from technical analysis, and the remaining 57 from the first basic group. In the LMS2 case, the first 20 best places are covered by 7 features from the quant pool, 9 from the technical pool, and only 4 from the first basic group. LMS2 also selects the advanced feature based on the logistic regression model for online learning first.

The last method that we use as the basis for the feature selection process is based on LDA. In a similar fashion, we use two different criteria as a measure for the selection process. In the LDA1 case, the first 20 best places are covered by 10 quant features, 3 technical indicators, and 7 from the first basic group. The first 100 best positions are covered by 19 quant features, 20 technical features, and the remaining 61 places from the first basic group. Again, the first place is covered by the advanced feature based on the logistic regression model for online learning. The last feature selection model, LDA2, selects 6 features from the quant pool, 6 from the technical pool, and 8 from the first basic group. LDA2 selects for the first best 100 places 24 quant features, 27 technical features, and 49 from the first basic group. To gain better insight into the feature list, we present the names of the best 10 features for each of the 5 sorting methods in Table 9.

The second interpretation of our findings is the performance of the 12 different classifiers (based on LMS, LDA, and RBFN) that we used to measure, in terms of F1 score, the predictability of the mid-price movement. Fig.3 provides a quick overview of the F1 score performance in terms of best feature numbers and classifiers. We can divide these twelve models (pairs based on the sorting and classification method) into three groups according to their response in terms of information flow. The first group, where LMS2-LMS, LDA2-LMS, LMS2-RBFN, and LDA2-RBFN belong, reach their plateau very early in the incremental process of adding less informative features. These models where able to reach their maximum (or close to their maximum) F1 score performance with approximately 5 best features, which means that the dimensionality of the input matrix to the classification model is quite small. The second group of models, Entropy-LMS, LMS1-LMS, LDA1-LMS, Entropy-RBFN, LMS1-RBFN, and LDA1-RBFN, had a slower reaction in the process of reaching their best F1 score performance. The last group of models, LDA1-LDA and LDA2-LDA, reached their best performance very early in the process (which is not higher performance compared to the other models) with only five features. Interestingly, it is right after this point that their predictability power starts to decrease.

The experiments conducted show that this quantitative analysis can provide significant trading information, but results are improved with respect to features also taken from the technical pool. Features on the top of the lists come from the logistic regression model. This is the very first time this model is presented as a feature in the HFT sphere. This shows that more sophisticated features from the pool of quantitative analysis will provide the ML trader with vital information regarding metrics prediction. We would like to point out that the main idea of the present work is to utilize the majority of the technical indicators [2] and provide a fair evaluation against other state-of-the-art features and advanced quantitative hand-crafted features like the adaptive logistic regression feature and see which ones are more informative. Classification performance can be easily improved by utilizing more advanced classifiers like convolutional neural networks and recurrent neural networks (e.g., [22]), but it is outside

[2] In the literature it is very common to see experiments based on a limited number of technical indicators.

Feature Sets	Description
Entropy	
1	Autocorrelation
2	Donchian Channels
3	Highest High
4	Center of Gravity Oscillator
5	Heikin-Ashi
6	Linear Regr. - Regression Coeffic.
7	Linear Regr. - Correlation Coeffic.
8	T3
9	TEMA
10	TRIMA
LMS1	
1	Logistic Regr. - Local Spatial Ratio
2	Best LOB Level - Bid Side Volume
3	Second Best LOB Level - Ask Volume
4	Price and Volume Derivation
5	Best LOB Level - Ask Side
6	Linear Regr. - Corr. Coeffic.
7	Logistic Regr. - Logistic Coeffic.
8	Logistic Regr. - Extended Spatial Ratio
9	Autocorrelation for Log Returns
10	Partial Autocorrelation
LMS2	
1	Logistic Regression - Spatial Ratio
2	Cointegration - Boolean Vector
3	Cointegration - Test Statistics
4	Price and Volume Means
5	Average Type Intensity
6	Average Type Intensity
7	Spread & Mid-Price
8	Alligator Jaw
9	Directional Index
10	Fractals
LDA1	
1	Logistic Regression - Spatial Ratio
2	Second Best LOB Level - Ask Volume
3	Price & Volume derivation
4	Spread & Mid-Price
5	Partial Autocorrelation for Log Returns
6	Linear Regression Line - Squared Corr. Coeffic.
7	Order Book Imbalance
8	Linear Regression - Corr. Coeffic.
9	Linear Regression - Regr. Coeffic.
10	Third Best LOB Level - Ask Volume
LDA2	
1	Logistic Regression - Probability Estimation
2	Logistic Regression - Spatial Ratio
3	Bollinger Bands
4	Alligator Teeth
5	Cointegration - Test Statistics
6	Best LOB Level - Bid Side Volume
7	Cointegration - p Values
8	Price & Volume Means
9	Price & Volume derivation
10	Price differences

Table 9: List for the first 10 best features for the 5 sorting methods

of the scope of our evaluation. Our work open avenues for other application as well. For instance, the same type of analysis is suitable for exchange rates and bitcoin time series. Last, we intend to test our experimental protocol on a longer trading period.

7. Conclusion

In this paper, we extracted hand-crafted features inspired by technical and quantitative analysis and tested their validity on the mid-price movement prediction task. We used entropy, least-mean-squares (LMS), and linear discriminant analysis (LDA) criteria to guide feature selection methods combined with linear and non-linear classifiers based on LMS, LDA, and RBFN. This work is the first attempt of this extent to develop a framework in information edge discovery via informative hand-crafted features. Therefore, we provided the description of three sets of hand-crafted features adjusted to the HFT universe by considering each 10-message book block as a separate trading unit (i.e. trading days). We evaluated our theoretical framework on five ITCH feed data stocks from the Nordic stock market. The dataset contained more than 4.5 million events and was incorporated into the hand-crafted features. The results suggest that sorting methods and classifiers can be combined in such a way that market makers and traders can reach, with only very few informative features, the best performance of their algorithm. Furthermore, the very advanced quantitative feature based on logistic regression for online learning is placed first among most the five sorting methods. This is a strong indication for future research on developing more advanced features combined with more sophisticated feature selection methods.

Appendix

A. Feature Pool

A.1. First Group of Features

This set of features is based on [46] and [64] and is divided into three groups: basic, time-insensitive, and time-sensitive. These are fundamental features since they reflect the raw data directly without any statistical analysis or interpolation. We calculated them as follows:

A.1.1. Basic

- $u_1 = \{P_i^{ask}, V_i^{ask}, P_i^{bid}, V_i^{bid}\}_{i=1}^{n}$

which represents the raw data of the 10 levels of our LOB.

A.1.2. Time-Insensitive

- $u_2 = \{(P_i^{ask} - P_i^{bid}), (P_i^{ask} + P_i^{bid})/2\}_{i=1}^{n}$
- $u_3 = \{P_n^{ask} - P_1^{ask}, P_1^{bid} - P_n^{bid}, |P_{i+1}^{ask} - P_i^{ask}|, |P_{i+1}^{bid} - P_i^{bid}|\}_{i+1}^{n}$
- $u_4 = \left\{ \frac{1}{n} \sum_{i=1}^{n} P_i^{ask}, \frac{1}{n} \sum_{i=1}^{n} P_i^{bid}, \frac{1}{n} \sum_{i=1}^{n} V_i^{ask}, \frac{1}{n} \sum_{i=1}^{n} V_i^{bid} \right\}$
- $u_5 = \left\{ \sum_{i=1}^{n} (P_i^{ask} - P_i^{bid}), \sum_{i=1}^{n} (V_i^{ask} - V_i^{bid}) \right\}$

where u_3 represents the spread and the mid-price, u_4 the price differences, and u_5 the price and the volume means, respectively.

A.1.3. Time-Sensitive

- $u_6 = \left\{ dP_i^{ask}/dt, dP_i^{bid}/dt, dV_i^{ask}/dt, dV_i^{bid}/dt \right\}_{i=1}^{n}$
- $u_7 = \left\{ \lambda^1_{\Delta t}, \lambda^2_{\Delta t}, \lambda^3_{\Delta t}, \lambda^4_{\Delta t}, \lambda^5_{\Delta t}, \lambda^6_{\Delta t} \right\}$
- $u_8 = \Big\{ \mathbf{1}_{\lambda^1_{\Delta_t} > \lambda^1_{\Delta_T}}, \mathbf{1}_{\lambda^2_{\Delta_t} > \lambda^2_{\Delta_T}}, \mathbf{1}_{\lambda^3_{\Delta_t} > \lambda^3_{\Delta_T}}, \mathbf{1}_{\lambda^4_{\Delta_t} > \lambda^4_{\Delta_T}}, \mathbf{1}_{\lambda^5_{\Delta_t} > \lambda^5_{\Delta_T}}, \mathbf{1}_{\lambda^6_{\Delta_t} > \lambda^6_{\Delta_T}} \Big\}$
- $u_9 = \{ d\lambda^1/dt, d\lambda^2/dt, d\lambda^3/dt, d\lambda^4/dt, d\lambda^5/dt, d\lambda^6/dt \}$

where u_6 represents the price and volume derivation, u_7 the average type intensity, u_8 the relative comparison intensity, and u_9 the limit activity acceleration, respectively.

A.2. Technical Analysis

Technical analysis is based mainly on the idea that historical data provides all the relevant information for trading prediction. The prediction, based on technical analysis, takes place according to open-close-high and low prices in day-to-day trading. We adjust this idea to the HFT ML problem for every 10-MB block of events. More specifically, we consider every 10-MB block as a 'trading' day (i.e. with t as the current 10-MB block and t-1 the previous 10-MB block), and we extract features according to this formation as follows:

A.2.1. Accumulation Distribution Line

Accumulation Distribution Line (ADL) [17] is a volume-based indicator for measuring supply and demand and is a three-step process:

- $MoneyFlowMultiplier = [(C_t - L_t) - (H_t - C_t)]/(H_t - L_t)$
- $MoneyFlowVolume_t = MoneyFlowMultiplier$ x $BlockPeriodVolume$
- $ADL = ADL_{t-1} + MoneyFlowVolume_t$

with C_t, L_t, and H_t being the closing, lowest, and highest current 10-MB block prices, respectively, and $BlockPeriodVolume$, ADL_{t-1}, and $MoneyFlowVolume_t$ are the total amounts of 10-MB block volume density, the previous ADL price, and the current $MoneyFlowVolume_t$, respectively.

A.2.2. Awesome Oscillator

An awesome oscillator (AO) [95] is used to capture market momentum. Here, we adjust the trading rules according to the previous block horizon investigation to 5 and 34 previous 10-MB blocks as follows:

- AO $= SMA_5((H_t + L_t)/2) - SMA_{34}((H_t + L_t)/2)$

where SMA_5 and SMA_{34} are the simple moving averages of the previous 5 and 34 previous blocks, respectively.

A.2.3. Accelerator Oscillator

An accelerator oscillator [95] is another market momentum indicator derived from AO. It is calculated as follows:

- $AC = AO - SMA_5(AO)$

A.2.4. Average Directional Index

An average directional index (ADX) indicator [94] has been developed to identify the strength of a current trend. The ADX is calculated as follows:

- $TR = max(H_t - L_t, |H_t - CL_{t-1}|, |L_t - CL_{t-1}|)$
- $+DM = H_t - H_{t-1}$
- $-DM = L_t - L_{t-1}$
- $TR_{14} = TR_{t-1} - (TR_{t-1}/14) + TR$
- $+DM_{14} = (+DL_{t-14}) - ((+DL_{t-14})/14) + (+DM)$
- $-DM_{14} = (-DL_{t-14}) - ((-DL_{t-14})/14) + (-DM)$
- $+DI_{14} = 100 \times ((+D_{14})/(+TR_{14}))$
- $-DI_{14} = 100 \times ((-D_{14})/(-TR_{14}))$
- $DI_{diff_{14}} = |(+D_{14}) - (-D_{14})|$
- $DI_{sum_{14}} = |(+D_{14}) + (-D_{14})|$
- $DX = 100 \times ((DI_{diff_{14}})/(DI_{sum_{14}}))$
- $ADX = (ADX_{t-1} \times 13) + DX)/14$

where TR = true range, H_t = the current 10-block's highest MB price, L_t = the current 10-block's lowest MB price, CL_t = the previous 10-block's closing MB price, $+DM$ = positive Directional Movement (DM), $-DM$ = negative DM, TR_{14} =TR based on the previous 14-blocks, TR_{t-1} = the previous TR price, $+DM_{14}$ = DM based on the previous 14 $+DM$ blocks, $-DM_{14}$ = DM based on the previous 14 $-DM$ blocks, $+DM_{t-14}$ = +DM of the previous 14 $+DM$ blocks, $DI_{diff_{14}}$ = is the directional indicator (DI) of the difference between $+DM_{14}$ and $-DM_{14}$, $DI_{sum_{14}}$ = DI of the sum between $+DM_{14}$ and $-DM_{14}$, DX = directional movement index and ADX_{t-1} = the previous average directional index.

A.2.5. Average Directional Movement Index Rating

An average directional movement index rating (ADXR) evaluates the momentum change of ADX, and it is calculated as the average of the current and previous price of ADX:

- $ADXR = (ADX + ADX_{t-1})/2$

A.2.6. Displaced Moving Average Based on Williams Alligator Indicator

A displaced moving average [34] is the basis for building a trading signal named Alligator. In practice, this is a combination of three moving averages (MA). We adjust this idea as follows:

- $Alligator_{Jaw} = SMA_{13}((H_t + L_t)/2)$
- $Alligator_{Teeth} = SMA_{8}((H_t + L_t)/2)$
- $Alligator_{Lips} = SMA_{5}((H_t + L_t)/2)$

where $SMA_{13}((H_t + L_t)/2)$, $SMA_{8}((H_t + L_t)/2)$, and $SMA_{5}((H_t + L_t)/2)$ are the simple moving averages based on the previous 13, 8, and 5 average highest and lowest block prices, respectively.

A.2.7. Absolute Price Oscillator

An absolute price oscillator (APO) belongs to the family of price oscillators. It is a comparison between fast and slow exponential moving averages and is calculated as follows:

- $M_t = (H_t + L_t)/2$
- APO $= EMA_5(M_t) - EMA_{13}(M_t)$

where $EMA_5(M_t)$ and $EMA_{13}(M_t)$ are the exponential moving averages of range 5 and 13 periods, respectively, for the average of high and low prices of the current 10-MB block.

A.2.8. Aroon Indicator

An Aroon indicator [14] is used as a measure of trend identification of an underlying asset. More specifically, the indicator has two main bodies: the uptrend and downtrend calculation. We calculate the Aroon indicator based on the previous twenty 10-MB blocks for the highest-high and lowest-low prices, respectively, as follows:

- $Arron_{Up} = (20 - H_{high_{20}}/20) \times 100$
- $Arron_{Down} = (20 - L_{low_{20}}/20) \times 100$

where $H_{high_{20}}$ and $L_{low_{20}}$ are the highest-high and lowest-low 20 previous 10-MB block prices, respectively.

A.2.9. Aroon Oscillator

An Aroon oscillator is the difference between $Aroon_{Up}$ and $Aroon_{Down}$ indicators, which makes their comparison easier:

- Arron Oscillator = $Aroon_{Up} - Aroon_{Down}$

A.2.10. Average True Range

Average true range (ATR) [93] is a technical indicator which measures the degree of variability in the market and is calculated as follows:

- ATR = $(ATR_{t-1} \times (N-1) + TR)/N$

Here we use N =14, where N is the number of the previous 10-TR values, and ATR_{t-1} is the previous ATR 10-MB block price.

A.2.11. Bollinger Bands

Bollinger bands [8] are volatility bands which focus on the price edges of the created envelope (middle, upper, and lower band) and can be calculated as follows:

- $BB_{middle} = SMA_{20}(CL)$
- $BB_{upper} = SMA_{20}(CL) + BB_{std_{20}} \times 2$
- $BB_{lower} = SMA_{20}(CL) - BB_{std_{20}} \times 2$

where BB_{middle}, BB_{upper}, and BB_{lower} represent the middle, upper, and lower Bollinger bands, $SMA_{20}(CL)$ represents the simple moving average of the previous twenty 10-block closing prices, and $BB_{std_{20}}$ represents the standard deviation of the last twenty 10-MB blocks.

A.2.12. Ichimoku Clouds

Ichimoku clouds [60] are 'one glance equilibrium charts,' which means that the trader can easily identify a good trading signal and is possible since this type of indicator contains dense information (i.e. momentum and trend direction). Five modules are used in an indicator's calculation:

- Conversion Line $(Tenkan - sen) = (H_9 + L_9)/2$
- Base Line $(Kijun - sen) = H_{26} + L_{26}$
- Leading Span A $(Senkou\ Span\ A)$ = (Conversion Line + Base line)/2
- Leading Span B $(Senkou\ Span\ B) = (H_{52} + L_{52})/2$
- Lagging Span $(Chikou\ Span) = CL_{26}$

where H, L, and CL denotes the highest, lowest, and closing prices of the 10-MB raw data where subscripts 9, 26, and 52 denotes the historical horizon of our trading rules.

A.2.13. Chande Momentum Oscillator

A Chande momentum oscillator (CMO) [14] belongs to the family of technical momentum oscillators and can monitor overbought and oversold situations. There are two modules in the calculation process:

- $S_u = \sum_{i=1}^{19} CL_i \times \mathbb{1}_{CL_t > CL_{t-19}}$
- $S_d = \sum_{i=1}^{19} CL_i \times \mathbb{1}_{CL_t < CL_{t-19}}$
- CMO = $100 \times (S_u - S_d)/(S_u + S_d)$

where CL_i is the 10-block's closing price with $i = 1$, and CL_t and CL_{t-19} are the current block's closing price and the 19 previous blocks closing prices, respectively.

A.2.14. Chaikin Oscillator

The main purpose of a Chaikin oscillator [62] is to measure the momentum of the accumulation distribution line as follows:

- $MFM = (CL_t - L_t) - (H_t - CL_t)]/(H_t - L_t)$
- $MFV = MFM \times \sum_{j=1}^{10} V_j$
- $ADL = ADL_{t-1} + MFM$
- Chaikin Oscillator = $EMA_3(ADL)$ - $EMA_{10}(ADL)$

where MFM and MFV stand for *Money Flow Multiplier* and *Money Flow Volume*, respectively, V is the volume of each of the trading events in the 10-block MB, and $EMA_3(ADL)$ and $EMA_{10}(ADL)$ are the exponential moving average for the past 3 and 10 10-MB blocks, respectively.

A.2.15. Chandelier Exit

A Chandelier exit [25] is part of the trailing stop strategies based on the volatility measured by the ATR indicator. It is separated based on the number of ATRs that are below the 22-period high (long) or above the 22-period low (short) and is calculated as follows:

- $Chandelier_{Long} = H_{22} - ATR_{22} \times 3$
- $Chandelier_{Short} = L_{22} + ATR_{22} \times 3$

where H_{22} and L_{22} denote the highest and lowest prices for a period of 22 10-MB blocks, and ATR_{22} are the ATR values for the 22 previous 10-MB blocks.

A.2.16. Center of Gravity Oscillator

A center of gravity oscillator (COG) [24] is a comparison of current prices against older prices within a specific time window and is calculated as follows:

- $M_t = (H_t + L_t)/2$
- COG = $-(M_t + r \times M_{t-1})/(M_t + M_{t-1})$

where M_t is the current mid-price of the highest and lowest prices of each of the 10-MB blocks, and r is a weight that increases according to the number of the previous M_{t-1} prices.

A.2.17. Donchian Channels

A Donchian channel (DC) [72] is an indicator which bands the signal and notifies the ML trader of a price breakout. There are three modules in the calculation process:

- $DC_{upper} = H_{high_{20}}$
- $DC_{lower} = L_{low_{20}}$
- $DC_{middle} = (H_{high_{20}} + L_{low_{20}})/2$

where $H_{high_{20}}$ and $L_{low_{20}}$ are the highest high and lowest low prices of the previous twenty 10-MB blocks.

A.2.18. Double Exponential Moving Average

A double exponential moving average (DEMA) [59] provides a smoothed average and offers a diminished amount of delays as follows:

- $M_t = (H_t + L_t)/2$
- $DEMA = 2 \times EMA_{20}(M_t) - EMA_{20}(EMA_{20}(M_t))$

where EMA_{20} is the exponential moving average of span 20 of the closing prices under the 10-MB block format.

A.2.19. Detrended Price Oscillator

A detrended price oscillator (DPO) is an indicator used for short-term and long-term signal identification. A DPO eliminates cycles which are longer than the MA horizon. On day-to-day trading, the closing prices are considered for the calculation, but here, we use the highest 10-MB block price as follows:

- $DPO = (H_{high_{10}}/(10 + 2)) - SMA_{10}(CL)$.

A.2.20. Heikin-Ashi

Heikin-Ashi [91] is a candlestick method and is described as a visual technique that eliminates irregularities:

- $Heikin_{Close} = (O_t + H_t + L_t + CL_t)/4$
- $Heikin_{Open} = (O_{t-1} + CL_{t-1})/2$
- $Heikin_{High} = max(H_t, O_{t-1}, CL_{t-1})$
- $Heikin_{Low} = min(L_t, O_{t-1}, CL_{t-1})$

where O_{t-1} and CL_{t-1} are the open and close prices of the previous 10-MB block.

A.2.21. Highest High and Lowest Low

Highest high and lowest low creates an envelope of the trading signal for the last twenty 10-MB blocks:

- $Highest_{High} = H_{high_{20}}$
- $Lowest_{Low} = L_{low_{20}}$

A.2.22. Hull MA

A Hull moving average is a weighted moving average that reduces the smoothing lag effect by using the square root of the block period. It is calculated as follows:

- $Hull_{MA} = WMA_{\sqrt{10}}(AHL)(2 \times WMA_5(AHL) - WMA_{10}(AHL))$

where $WMA_5(AHL)$ and $WMA_{10}(AHL)$ denote the weighted moving average of the average high and low 10-MB block for periods 5 and 10, respectively.

A.2.23. Internal Bar Strength

Internal bar strength (IBS) [67] is based on the position of the days closing price in relation to the days range where we adjust this idea to the 10-MB block setup as follows:

- $IBS = (CL_t - L_t)/(H_t - L_t)$.

A.2.24. Keltner Channels

Keltner channels [45] are based on Bollinger bands. The main difference, for this volatility-based indicator, is that it uses ATR instead of standard deviation, as follows:

- $Middle_{Channel} = EMA_{20_{AHL}}$
- $Upper_{Channel} = Middle_{Channel} + 2 \times ATR_{10}$
- $Lower_{Channel} = Middle_{Channel} - 2 \times ATR_{10}$.

A.2.25. Moving Average Convergence/Divergence Oscillator (MACD)

A moving average convergence/divergence oscillator [1] is a measure of the convergence and divergence of two moving averages and is calculated as follows:

- $MACD = EMA_{12}(AHL) - EMA_{26}(AHL)$

where AHL is the average of high and low prices for 12 and 26 previous 10-MB blocks, respectively, with $EMA_{12}(AHL)$ and $EMA_{26}(AHL)$ as the exponential moving average of AHL of span 12 and 26, respectively.

A.2.26. Median Price

Median price is an indicator which simplifies the price overview. We calculate this indicator based on the 10-MB block highest and lowest average prices:

- $Median_t = (H_t + L_t)/2$

A.2.27. Momentum

A momentum (MOM) indicator measures the rate of change of the selected time series. In our case, we calculate it based on closing prices:

- $MOM = CL_t - CL_{t-1}$.

A.2.28. Variable Moving Average

A variable moving average (VMA) [13] is a dynamic indicator which acts as a variable-length moving average with volatility-adaptation capabilities. We calculate VMA based on the efficiency ratio (ER) as follows:

- $Direction = |CL_t - CL_{t-3}|$
- $Volatility = 3 \times \sum_{ii=1}^{3} |CL_{ii} - CL_{ii+1}|$
- $ER = Direction/Volatility$
- $VMA = \sum_{jj=1}^{3} \alpha \times ER_{jj} \times CL_{jj}$

where $\alpha = 2/(N+1)$, for N = 3 previous 10-MB blocks, is the adaptive parameter.

A.2.29. Normalized Average True Range

A normalized average true range (NATR) normalizes the average true range as follows:

- $NATR = (ATR/CL_t) \times 100$.

A.2.30. Percentage Price Oscillator

A percentage price oscillator (PPO) displays the convergence and divergence of two moving averages and focuses on the percentage change of the larger moving average, as follows:

- $MACD = EMA_{12}(AHL) - EMA_{26}(AHL)$
- $PPO = (MACD/EMA_{26}(AHL)) \times 100$.

A.2.31. Rate of Change

Rate of change (ROC) measures the ascent or descent speed of the time series change:

- $ROC = (CL_t - CL_{t-12}/CL_{t-12}) \times 100$.

A.2.32. Relative Strength Index

A relative strength index (RSI) [94] is a measure of the velocity and magnitude of directional time series movements and is calculated as follows:

- $CL_d = CL_t - CL_{t-1}$
- $AG_{14} = \sum_{l=1}^{14} CL_{d_l} \mathbb{1}_{CL_{d_t} > CL_{d_{t-1}}}$
- $AL_{14} = \sum_{l=1}^{14} CL_{d_l} \mathbb{1}_{CL_{d_t} < CL_{d_{t-1}}}$
- $Relative_{Strength} = AG_{14}/AL_{14}$
- $RSI = 100 - 100/(1 + Relative_{Strength})$

where AG_{14} and AL_{14} denotes the average gain and loss of the last fourteen 10-MB blocks, respectively.

A.2.33. Parabolic Stop and Reverse

Parabolic SAR (PSAR) [93] is a trend following indicator which protects profits. There are two main modules for its calculation, the Rising SAR and the Falling SAR, and they are calculated as follows:

- Rising SAR
  - $AF = Incremental\ increase\ of\ a\ predefined\ step$
  - $EP = H_{High_5}$
  - $SAR = SAR_{t-1} + AF_{t-1}(EP_{t-1} - SAR_{t-1})$
- Falling SAR
  - $AF = Incremental\ increase\ of\ a\ predefined\ step$
  - $EP = L_{Low_5}$
  - $SAR = SAR_{t-1} - AF_{t-1}(EP_{t-1} - SAR_{t-1})$

where AF is the acceleration factor, and EP is the extreme point

A.2.34. Standard Deviation

Standard deviation is a measure of volatility. We calculate this indicator based on the closing prices of every 10-MB block, as follows:

- $Deviation = CL_t - SMA_{10}(CL)$
- $SASD = \sqrt{SMA_{10}(SVD)}$

where $SMA_{10}(CL)$ is the simple moving average of the last 10 closing 10-MB prices, $SASD$ is the squared deviation of the SMA of the standard deviation (SVD) of the last 10 closing values of our 10-MB blocks.

A.2.35. Stochastic Relative Strength Index

A stochastic relative strength index (Stoch RSI) [14] is a range-bound momentum oscillator which provides information for the RSI based on the closing prices in terms of high and low stock prices:

- $Stoch_{RSI} = (RSI_{curr} - RSI_{L_{Low_{10}}})/(RSI_{H_{High_{10}}} - RSI_{L_{Low_{10}}})$

where $RSI_{L_{Low_{10}}}$ and $RSI_{H_{High_{10}}}$ are the lowest low and highest high of the last ten RSI values.

A.2.36. T3-Triple Exponential Moving Average

A triple exponential moving average [88] is a moving average indicator where the main motivation for its development is to reduce lag in the time series response. For this reason, we use the closing prices for our calculation and perform a reversal explanation calculation as follows:

- $T3 = c_1 \times EMA_6 + c_2 \times EMA_5 + c_3 \times EMA_4 + c_4 \times EMA_3$

with:

- $c_1 = -\alpha^3$
- $c_2 = 3 \times \alpha^2 + 3 \times \alpha^3$
- $c_3 = 6 \times \alpha^2 3 \times \alpha 3 \times \alpha^3$
- $c_4 = 1 + 3 \times \alpha + \alpha^3 + 3 \times \alpha^2$
- $EMA_1 = EMA_{10}(CL)$
- $EMA_2 = EMA_{10}(EMA_1)$
- $EMA_3 = EMA_{10}(EMA_2)$
- $EMA_4 = EMA_{10}(EMA_3)$
- $EMA_5 = EMA_{10}(EMA_4)$
- $EMA_6 = EMA_{10}(EMA_5)$

where α is the volume factor, and $EMA_{10}(CL)$ is the exponential moving average of the 10 previous 10-MB closing prices.

A.2.37. Triple Exponential Moving Average

A triple exponential moving average (TEMA) [59] is an attempt to reduce the lag associated with MA by adding weight to the most recent prices:

- $TEMA = (3 \times EMA_{10}(CL)) - (3 \times EMA_{10}(EMA_{10}(CL)) + EMA_{10}(EMA_{10}(EMA_{10}(CL)))$

with EMA being, in every case, the exponential moving average of the previous 10 prices (i.e. previous EMA and closing prices).

A.2.38. Triangular Moving Average

A triangular moving average (TRIMA) is the average of the time series with emphasis placed on the middle region:

- $TRIMA = SMA_{10}(SMA_{10}(SMA_{10}(CL)))$

Where, for its calculation, we use the closing prices of the last 10 10-MB blocks.

A.2.39. Triple Exponential Average

A triple exponential average (TRIX) is a momentum oscillator which measures the rate of change of the triple smoothed moving average as follows:

- $EMA_{First} = EMA_{10}(CL)$
- $EMA_{Double} = EMA_{10}(EMA_{First})$
- $EMA_{Triple} = EMA_{10}(EMA_{Double})$
- $TRIX = 1\text{-}period\ Rate\ of\ Change.$

A.2.40. True Strength Index

A true strength index (TSI) [6] is an indicator which specifies the overbought and oversold levels with market return anticipation. We calculate TSI as follows:

- $PC = CL_k - CL_{k-1}$, where $k = 2, ..., T$
- $APC = |CL_k - CL_{k-1}|$, where $k = 2, ..., T$
- $EMA_1 = EMA_{25}(PC)$
- $EMA_2 = EMA_{13}(EMA_1)$
- $EMA_3 = EMA_{25}(APC)$
- $EMA_4 = EMA_{13}(EMA_3)$
- $TSI = 100 \times EMA_2/EMA_4$

where PC represents the closing price differences for the whole time series lookback period.

A.2.41. Ultimate Oscillator

An ultimate oscillator (UO) [96] is a momentum oscillator indicator with a multiple timeframe perspective. There are three main modules as presented in the following calculations:

- Average of seven 10-MB blocks
  - $BP = CL_t - (CL_{t-1}\mathbb{1}_{CL_{t-1}<L_t} + L_t\mathbb{1}_{CL_{t-1}>L_t})$
  - $TR_1 = CL_{t-1}\mathbb{1}_{CL_{t-1}>H_t} + H_{curr}\mathbb{1}_{CL_{t-1}<H_t}$
  - $TR_2 = CL_{t-1}\mathbb{1}_{CL_{t-1}<L_t} + L_t\mathbb{1}_{CL_{t-1}>L_t}$
  - $TR = TR_1 + TR_2$
  - $Average_7 = \sum_{l=1}^{7} BP_l / \sum_{l=1}^{7} TR_l$
- Average of fourteen 10-MB blocks
  - $BP = CL_t - (CL_{t-1}\mathbb{1}_{CL_{t-1}<L_t} + L_t\mathbb{1}_{CL_{t-1}>L_t})$
  - $TR_3 = CL_{t-1}\mathbb{1}_{CL_{t-1}>H_t} + H_t\mathbb{1}_{CL_{t-1}<H_t}$
  - $TR_4 = CL_{t-1}\mathbb{1}_{CL_{t-1}<L_t} + L_t\mathbb{1}_{CL_{t-1}>L_t}$
  - $TR = TR_3 + TR_4$

  - $Average_{14} = \sum_{l=1}^{14} BP_l / \sum_{l=1}^{14} TR_l$

- Average of twenty-eight 10-MB blocks
  - $BP = CL_t - (CL_{t-1}\mathbb{1}_{CL_{t-1}<L_t} + L_t\mathbb{1}_{CL_{t-1}>L_t})$
  - $TR_5 = CL_{t-1}\mathbb{1}_{CL_{t-1}>H_t} + H_t\mathbb{1}_{CL_{t-1}<H_t}$
  - $TR_6 = CL_{t-1}\mathbb{1}_{CL_{t-1}<L_t} + L_t\mathbb{1}_{CL_{t-1}>L_t}$
  - $TR = TR_5 + TR_6$
  - $Average_{28} = \sum_{l=1}^{28} BP_l / \sum_{l=1}^{28} TR_l$
- $UO = 100 \times \big[(4 \times Average_7) + (2 \times Average_{14}) + Average_{28}\big]/(4+2+1)$

where BP represents buying pressure.

A.2.42. Weighted Close

Weighted close (WCL) is the average of the four universal types of prices which are included in each of our 10-MB blocks:

- $WCL = (H_t + L_t + 2 \times CL_t)/4$.

A.2.43. Williams %R

Williams %R [96] is a momentum technical indicator which informs the ML trader whether the market is trading close to the high or low trading range. It is calculated as follows:

- $\%R = -100 \times (H_{High_{14}} - CL_t)/(H_{High_{14}} - L_{Low_{14}})$

where -100 corrects the inversion.

A.2.44. Zero-Lag Exponential Moving Average

Zero-lag exponential moving average (ZLEMA) belongs to the EMA family of indicators where the main purpose is to reduce or remove the impulse lag by introducing an error term. It is calculated as follows:

- $error = CL - CL_{lag}$
- $Input = CL + error$
- $ZLEMA = EMA_{10}(Input)$

where $lag = (N-1)/2$ with N = 1 in our case.

A.2.45. Fractals

A fractal [34] is an indicator used to detect top and bottom trends by focusing on five consecutive blocks, which, in our case, are five 10-MB blocks used for two different scenarios:

- *Buy Fractals*

  A buy fractal is a sequence of five consecutive 10-MB blocks where the highest high is preceded by two lower highs and is followed by two lower highs.

- *Sell Fractals*

  The opposite framework is a sell fractal. 10-MB blocks can overlap in the quest of these two types of fractals.

Here, we calculate fractals separately for the open, close, lowest, and highest 10-MB block prices.

A.2.46. Linear Regression Line

Linear regression line (LRL) is a basic statistical method that provides information for a future projection wherein trading is used to capture overextended price trends. We perform LRL for each 10-MB block without any prior stationarity assumptions. The basic calculations are as follows:

- $PV = c_1 + c_2 \times MB_{prices}$
- $c_2 = r \times (std_{PV}/std_{MB_{prices}})$
- $r = \frac{\left(\sum_{i=1}^{10}(MB_{prices}(i)-\overline{MB_{prices}})(PV(i)-\overline{PV})\right)}{\left(\sqrt{\sum_{i=1}^{10}(MB_{prices}(i)-\overline{MB_{prices}})^2}(\sum_{i=1}^{10}(PV(i)-\overline{PV})^2\right)}$
- $c_1 = \overline{PV} - c_2 \times \overline{MB_{prices}}$

where PV are the predicted values, r is the correlation coefficient, and $\overline{MB_{prices}}$ and $\overline{PV}$ are the mean of 10-MB block prices and predicted values, respectively.

A.2.47. Digital Filtering: Rational Transfer Function

A rational transfer function [49] is a representation of a linear time-invariant (LTI) filter, with the assumption that the input signal depends on the time-frequency domain, which describes the input-output relationship of a signal. In the Z-tranform domain, we have the following rational transfer function:

- $O(z) = \frac{b(1)+b(2)z^{-1}+...+b(n_b+1)+z^{-n_b}}{1+\alpha(2)z^{-1}+...+\alpha(n_a+1)z^{-n_\alpha}}I(z)$,

where:

- $I(z)$ and $O(z)$ are the input (i.e. 10-MB block closing prices) and output respectively,
- b are the numerator coefficients,
- α are the denominator coefficients,
- n_a is the feedback order,
- n_b is the feedforward order,
- z is the complex variable,
- the lookback period for the calculations is ten 10-MB blocks.

A.2.48. Digital Filtering: Savitzky-Golay Filter

A Savitzky-Golay (S-G) digital filter [76], [77] is a discrete convolution with a specific impulse response. We describe how the ML trader can obtain the S-G signal based on higher degree polynomials:

- Least-Square Filter
  - Objective: minimize error $\mathcal{E}_N = \sum_{i=l}^{m} w_i (y_i - \sum_{r=0}^{n} p_r x_i^r)^2$
  - Partial derivative of the polynomial coefficients:
    $\frac{\partial Q}{\partial p_k} = 0 \Rightarrow \sum_{i=l}^{m} w_i \sum_{r=0}^{n} p_r x_i^{r+k} = \sum_{i=l}^{m} w_i y_i x_i^k$
  - Finite time series allow order summation change:
    $\sum_{r=0}^{n} p_r \sum_{i=l}^{m} w_i x_i^{r+k} = \sum_{i=l}^{m} w_i y_i x_i^k$
  - As a result, the desired linear equations are the following:

$$\begin{bmatrix} \sum_{i=l}^{m} w_i x_i^0 & \sum_{i=l}^{m} w_i x_i^1 & \dots & \sum_{i=l}^{m} w_i x_i^n \\ \sum_{i=l}^{m} w_i x_i^1 & \sum_{i=l}^{m} w_i x_i^2 & \dots & \sum_{i=l}^{m} w_i x_i^{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=l}^{m} w_i x_i^n & \sum_{i=l}^{m} w_i x_i^{n+1} & \dots & \sum_{i=l}^{m} w_i x_i^{2n} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} =$$

$$\begin{bmatrix} \sum_{i=l}^{m} w_i y_i x_i^0 \\ \sum_{i=l}^{m} w_i y_i x_i^1 \\ \vdots \\ \sum_{i=l}^{m} w_i y_i x_i^n \end{bmatrix}$$

    equivalent to the notation $\mathbf{AP} = \mathbf{B}$ where matrix $\mathbf{A}^{-1} \in \mathbb{R}^{(n+1)\times(n+1)}$ under the condition that the polynomial degree is $n \leqslant m - l$.

- S-G Filter
  - Local convolution coefficients calculation

$$\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} \sum_{i=l}^{m} w_i x_i^0 & \dots & \sum_{i=l}^{m} w_i x_i^n \\ \sum_{i=l}^{m} w_i x_i^1 & \dots & \sum_{i=l}^{m} w_i x_i^{n+1} \\ \vdots & \ddots & \vdots \\ \sum_{i=l}^{m} w_i x_i^n & \dots & \sum_{i=l}^{m} w_i x_i^{2n} \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=l}^{m} w_i y_i x_i^0 \\ \sum_{i=l}^{m} w_i y_i x_i^1 \\ \vdots \\ \sum_{i=l}^{m} w_i y_i x_i^n \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,n} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,0} & c_{n,1} & \cdots & c_{n,n} \end{bmatrix} \begin{bmatrix} \sum_{i=l}^{m} w_i y_i x_i^0 \\ \sum_{i=l}^{m} w_i y_i x_i^1 \\ \vdots \\ \sum_{i=l}^{m} w_i y_i x_i^n \end{bmatrix}$$

– Response at the local point of 0 degree is:

$$y[0] = c_{0,0} * \sum_{i=l}^{m} w_i y_i x_i^0 + c_{0,1} * \sum_{i=l}^{m} w_i y_i x_i^1 + ... + c_{0,n} * \sum_{i=l}^{m} w_i y_i x_i^n$$

A.2.49. Digital Filtering: Zero-Phase Filter

A zero-phase filter [83] is a bidirectional filtering technique. With zero phase slope and even impulse response $h(n)$, the filter provides an output signal, which is a zero phase recursive signal. This method is suitable for our experimental protocol since we use training and testing sets rather than online learning architecture as we will do in 4.2.4. The calculation process is as follows:

- Real Impulse response: $h(n)$, $n \in \mathbb{Z}$
- Discrete-time Fourier Transformation:

$$H_{\omega T}(h) = \sum_{n=1}^{\infty} h(n) cos(\omega n T) \text{ - } j \sum_{n=1}^{\infty} h(n) sin(\omega n T)$$

- Based on Euler formula and h-even: $H(e^{j\omega T}) = \sum_{n=1}^{\infty} h(n) cos(\omega n T)$

A.2.50. Remove Offset and Detrend

We present three detrend methods for short-term cycle isolation and calculate them as follows:

- Remove Offset
  - $Offset = CL_t - (\sum_{l=1}^{n} CL_l)/n$
    where n denotes the 10-MB lookback period
- Detrend - Least Squares Fitting Line
  - $\mathbf{R}^2 = \sum_{i=1}^{n} [y_i - g(x_i)]^2$
  - $\frac{\partial(\mathbf{R}^2)}{\partial \alpha} = 0$
  - $\frac{\partial(\mathbf{R}^2)}{\partial b} = 0$
  - $$\begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} n & \sum_{i=l}^{n} x_i \\ \sum_{i=l}^{n} x_i & \sum_{i=l}^{n} x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=l}^{n} y_i \\ \sum_{i=l}^{n} x_i y_i \end{bmatrix}$$
    where α and b are the regression coefficients of g, and x represents the 10-MB closing prices.

A.2.51. Beta-like Calculation

Beta [31] is a volatility indicator which considers market risk. We adjust the notion of beta calculation to our experimental protocol where we index based on the average of the closing prices (Av_{CL}) with Av_t as the current MB block price and Av_{t-1} as the previous MB block's closing price. Our calculations are as follow:

- $Index_{CL} = CL_t/CL_{t-1}$
- $Index_{Av_{CL}} = Av_t/Av_{t-1}$
- $Dev_{CL} = Index_{CL} - SMA_{10}(Index_{CL})$
- $Dev_{Av_{CL}} = Index_{Av_{CL}} - SMA_{10}(Index_{Av_{CL}})$
- $Beta = cov_{10}(Dev_{CL}, Dev_{Av_{CL}})/var_{10}(Dev_{Av_{CL}})$

where $cov_{10}(Dev_{CL}, Dev_{Av_{CL}})$ represents the covariance between the current closing price and the average of the previous ten 10-MB closing prices, and $var_{10}(Dev_{Av_{CL}})$ is the variance of the sum of the ten previous $Index_{Av_{CL}}$.

A.3. Quantitative Analysis

Quantitative analysis captures trading activity mainly via statistical modelling. We focus on time series analysis, and more specifically, we examine features such as autocorrelation and partial autocorrelation, among others (e.g., statistical tests), while in the end of the section, we build an ML feature extraction method based on an online learning setup and test the validity of our hypothesis.

A.3.1. Autocorrelation and Partial Correlation

Autocorrelation and partial correlation [9], [28] are key features in the development of time series analysis. We treat our time series (i.e. stock prices and log returns per 10-MB blocks) as stationary stochastic processes since we estimate their local behavior based on 10-MB blocks:

- Autocorrelation
  - $ac_k = \frac{E[(z_t-\mu)(z_{t+k}-\mu)]}{\sqrt{E[(z_t-\mu)^2]E[(z_{t+k}-\mu)^2]}}$

  where z_t and z_{t+k} are the time series of lag k, $\mu = E[z_t] = \int_{-\infty}^{\infty} zp(z)dz$ and $\sigma_z^2 = E[(z_t-\mu)^2] = \int_{-\infty}^{\infty}(z-\mu)^2 p(z)dz$ are the constant mean and constant variance respectively.
- Partial Correlation
  - For the general case of an autoregressive model $AR(p)$, we have:
    $x_{i+1} = \phi_1 x_i + \phi_2 x_{i-1} + ... + \phi_p x_{i-p+1} + \xi_{i+1}$ of lag 1 up to p follows:
    - $< x_i x_{i+1} > = \sum_{j=1}^{p} (\phi_j < x_i x_{i-j+1} >)$
    - $< x_{i-1} x_{i+1} > = \sum_{j=1}^{p} (\phi_j < x_{i-1} x_{i-j+1} >)$
    - $< x_{i-k+1} x_{i+1} > = \sum_{j=1}^{p} (\phi_j < x_{i-k+1} x_{i-j+1} >)$

* $< x_{i-p+1}x_{i+1} > = \sum_{j=1}^{p} (\phi_j < x_{i-p+1}x_{i-j+1} >)$
  by diving with $N-1$ and autocovariance of zero separated periods (where the autocovariance function is even), all the lag periods above will be:

* $r_1 = \sum_{j=1}^{p} \phi_j r_{j-1}$
* $r_2 = \sum_{j=1}^{p} \phi_j r_{j-2}$
* $r_k = \sum_{j=1}^{p} \phi_j r_{j-k}$
* $r_p = \sum_{j=1}^{p} \phi_j r_{j-p}$

where 2.1.5, 2.1.6, 2.1.7 and 2.1.8 can be described by the matrix operations $\boldsymbol{R}\boldsymbol{\Phi} = \boldsymbol{r}$, $\boldsymbol{R} \in \mathbb{R}^{p \times p}$, $\boldsymbol{\Phi} \in \mathbb{R}^{p \times 1}$ and $\boldsymbol{r} \in \mathbb{R}^{p \times 1}$. The symmetric and full rank $\boldsymbol{\Phi}$ are as follows: $\hat{\boldsymbol{\Phi}} = \boldsymbol{R}^{-1}\boldsymbol{r}$.

– *Yule-Walker* Equations calculation:

* Lag interval $1 \leqslant i \leqslant p$
* $\hat{\boldsymbol{\Phi}} = \left(\boldsymbol{R}^{(i)}\right)^{-1} \boldsymbol{r}^{(i)} = \begin{bmatrix} \hat{\phi_1} \\ \hat{\phi_2} \\ \vdots \\ \hat{\phi_i} \end{bmatrix}$

A.3.2. Cointegration

We investigate time-series equilibrium [35], [27] by testing the cointegrated hypothesis. Utilizing the cointegration test will help ML traders avoid the problem of spurious regression. We employ the Engle-Granger (EG) test for the multivariable case of LOB ask (A_t) and bid (B_t) times series. We formulate the EG test for the ask and bid LOB prices as follows:

- A_t and $B_t \sim I(d)$, where $I(d)$ represents the order of integration
- Cointegration equation based on the error term: $u_t = A_t - \alpha B_t$
- EG Hypothesis: $u(t) \sim I(d), d \neq 0$
- Perform ordinary leat squares (OLS) for the estimation of $\hat{\alpha}$ and unit root test for: $\hat{u} = A_t - \hat{\alpha} B_t$

A.3.3. Order Book Imbalance

We calculate the order book imbalance [81] based on the volume depth of our LOB as follows:

- $VI = \frac{V_l^b - V_l^\alpha}{V_l^b + V_l^\alpha}$

where V_l^α and V_l^b are the volume sizes for the ask and bid LOB sides at level l.

B. Feature Sorting Lists

A detailed feature name list is available upon request.

1. Features sorting list based on Entropy:

   {217;157;165;154;164;207;209;190;191;192;193;174;182;183;146;161;171;172;173;156;155;
   184;194;137;177;176;138;195;136;218;213;181;147;245;243;247;188;241;242;244;240;246;
   255;248;249;189;210;265;211;226;236;225;235;221;231;223;233;222;232;214;169;220;
   230;228;238;224;234;83;84;227;237;139;135;134;142;162;140;185;129;86;128;186;212;
   141;250;261;16;20;262;153;14;81;12;208;260;4;18;10;259;24;196;2;163;150;187;82;28;
   197;22;8;26;6;32;30;36;34;40;148;175;160;149;38;151;60;59;58;57;56;55;54;53;52;51;
   198;215;216;158;167;159;168;152;166;100;102;21;25;29;13;17;9;5;33;1;23;19;15;27;11;31;
   37;7;3;35;85;39;104;252;96;98;106;269;108;92;112;110;88;253;116;124;120;145;94;50;
   90;180;256;114;49;170;118;123;122;48;126;268;119;115;80;47;111;143;144;46;107;70;45;
   125;103;121;44;117;43;113;99;42;109;254;270;271;105;41;95;101;91;272;97;273;93;179;
   69;178;87;68;79;89;127;67;78;201;199;66;133;77;65;76;205;203;61;200;202;71;75;62;
   72;64;206;204;257;63;132;131;74;73;130;251;258;267;266;219;229;239;263;264}

2. Features sorting list based on LMS1:

   {269;4;6;88;2;211;266;267;249;250;251;92;253;254;91;252;255;193;122;270;174;183;
   268;108;103;32;18;147;216;100;118;263;264;111;41;90;112;20;273;24;127;116;120;109;
   110;126;225;235;164;107;98;102;124;165;89;94;133;114;119;104;96;95;87;115;188;61;
   93;125;101;97;105;113;208;209;99;180;121;117;246;106;123;189;248;228;238;8;210;
   186;130;185;10;14;242;157;136;16;218;240;244;65;66;64;69;153;28;73;22;170;143;
   142;184;178;30;154;76;79;67;75;63;74;78;256;247;245;146;219;229;239;68;187;62;
   70;176;201;179;194;72;197;131;132;77;217;42;43;44;45;46;47;48;49;50;71;85;207;40;
   258;226;236;80;260;262;134;135;36;204;144;145;38;26;12;84;199;195;182;215;156;
   171;158;151;167;148;161;168;191;152;159;160;149;150;141;198;169;166;1;212;213;181;
   3;5;7;9;11;13;15;17;19;21;23;25;27;29;31;33;35;37;39;51;52;53;54;55;56;57;58;59;60;
   81;82;86;155;163;196;175;214;272;172;173;140;190;192;139;200;162;227;237;222;232;
   220;230;241;243;224;234;271;206;34;83;177;205;203;221;223;231;233;202;138;137;
   261;128;129;257;259;265}

3. Features sorting list based on LMS2:

   {269;259;262;83;129;130;49;137;177;205;203;257;223;202;200;199;243;273;176;206;
   256;204;265;132;10;2;14;84;170;78;240;226;182;157;61;80;242;217;41;70;50;207;165;
   150;164;93;87;62;43;89;66;215;18;154;251;111;222;8;261;201;258;270;271;65;96;151;
   216;272;210;186;124;120;153;94;187;92;211;117;109;101;162;166;29;213;184;185;198;
   195;127;146;191;192;193;196;174;171;159;149;161;139;125;113;106;102;266;118;104;
   218;36;38;156;190;250;63;85;133;12;121;90;34;40;175;91;248;241;227;245;152;128;
   189;178;214;136;142;158;224;225;112;99;115;264;212;169;141;163;220;221;188;197;
   194;209;208;168;22;105;114;110;268;16;23;181;140;119;123;100;126;122;260;244;
   246;134;135;131;56;103;173;167;6;228;47;97;255;107;180;71;155;4;254;253;179;
   82;138;32;28;143;252;116;30;144;147;88;108;73;95;98;249;20;51;160;247;55;59;
   5;148;42;7;76;31;54;3;145;77;46;19;231;48;17;15;81;232;21;45;52;230;236;37;1;24;
   58;69;13;53;35;67;172;33;183;79;86;26;267;75;219;25;234;9;44;39;11;229;237;57;
   235;239;60;27;68;64;74;233;238;72;263}

4. Features sorting list based on LDA1:

{269;6;88;41;255;211;266;250;249;10;252;251;253;268;8;108;114;174;193;254;100;
263;264;110;186;273;216;90;99;122;185;92;183;267;16;225;235;14;103;119;112;107;
95;104;147;111;91;115;270;127;109;116;120;18;89;94;118;126;98;180;106;208;209;124;
96;188;113;125;121;153;123;105;117;93;97;101;248;242;61;133;189;87;102;210;145;66;
65;64;69;136;184;142;73;76;157;75;74;78;67;63;79;170;178;77;219;229;239;262;182;
130;245;70;22;194;244;24;12;265;84;247;167;173;146;60;207;59;17;33;196;158;165;
4;218;25;149;203;3;36;53;37;86;21;30;155;58;164;48;246;161;223;26;85;226;205;43;
144;80;47;15;135;179;152;27;160;39;38;81;241;50;236;40;220;7;204;260;83;143;258;
168;166;51;141;162;23;57;19;131;9;42;132;82;56;49;62;154;128;5;228;259;55;181;191;
163;156;187;272;213;224;52;46;35;1;54;234;169;150;240;227;45;238;31;201;192;190;
199;261;172;44;134;2;140;129;20;72;214;215;195;68;151;271;198;237;171;11;29;137;221;
222;32;13;217;148;230;232;231;233;197;28;159;206;139;212;256;176;177;243;71;200;
34;138;175;202;257}

5. Features sorting list based on LDA2:

{265;269;257;138;259;4;262;83;108;68;129;205;204;120;6;48;248;141;179;203;212;
139;184;43;144;118;79;177;18;52;8;193;132;110;70;191;100;103;146;241;143;206;252;
247;273;63;211;207;22;10;142;244;16;258;122;221;219;87;217;47;93;140;40;180;202;
34;256;2;115;96;218;134;102;99;270;111;253;66;189;88;90;94;36;199;12;75;254;243;
72;137;45;272;64;251;77;222;155;255;210;104;209;174;267;105;194;50;14;126;109;32;
170;200;125;98;127;89;227;44;201;119;28;245;61;65;268;192;216;112;20;186;42;250;
187;107;121;116;84;185;128;30;237;156;124;160;195;133;147;41;223;215;123;113;135;
173;148;271;214;169;131;232;39;149;35;178;71;190;31;198;106;157;188;38;260;168;
153;228;55;5;69;246;114;67;15;266;76;152;33;183;37;27;238;46;242;17;166;101;54;
23;117;58;56;11;167;261;9;91;162;29;7;97;163;151;233;57;78;24;95;86;225;164;220;
154;181;249;171;230;229;130;172;60;26;182;51;1;3;136;159;25;59;208;145;85;53;80;
224;92;240;13;81;231;175;264;197;150;74;158;234;213;196;176;235;19;21;263;165;82;
226;236;73;161;239;62;49}

Acknowledgment

The research leading to these results has received funding from the H2020 Project BigDataFinance MSCA-ITN-ETN 675044 (http://bigdatafinance.eu), Training for Big Data in Financial Research and Risk Management.

The authors wish to acknowledge CSC-IT Center for Science, Finland, for generous computational resources.

References

[1] Aspray, T. (1989). Individual stocks and macd. *Technical Analysis of Stocks & Commodities*, 7(2):56–61.

[2] Baetje, F. & Menkhoff, L. (2016). Equity premium prediction: Are economic and technical indicators unstable? *International Journal of Forecasting*, 32(4):1193–1207.

[3] Bank, P. & Baum, D. (2004). Hedging and portfolio optimization in financial markets with a large trader. *Mathematical Finance*, 14(1):1–18. Available from: http://dx.doi.org/10.1111/j.0960-1627.2004.00179.x.

[4] Batchelor, R. & Kwan, T. Y. (2007). Judgemental bootstrapping of technical traders in the bond market. *International Journal of Forecasting*, 23(3):427–445.

[5] Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *Trans. Neur. Netw.*, 5(4):537–550. Available from: http://dx.doi.org/10.1109/72.298224.

[6] Blau, W. (1991). Double smoothed-stochastics. *Technical Analysis of Stocks and Commodities*, 9.

[7] Bollerslev, T. (1987). A conditionally heteroskedastic time series model for speculative prices and rates of return. *The Review of Economics and Statistics*, 69(3):542–547. Available from: http://www.jstor.org/stable/1925546.

[8] Bollinger, J. (2001). *Bollinger on Bollinger bands*. McGraw Hill Professional.

[9] Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

[10] Brasileiro, R. C., Souza, V. L. F., Fernandes, B. J. T., & Oliveira, A. L. I. (2013). Automatic method for stock trading combining technical analysis and the artificial bee colony algorithm. In *IEEE Congress on Evolutionary Computation*, pages 1810–1817.

[11] Broomhead, D. S. & Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom).

[12] Chan, L. K. C., Karceski, J., & Lakonishok, J. (1999). On portfolio optimization: Forecasting covariances and choosing the risk model. *The Review of Financial Studies*, 12(5):937–974. Available from: +http://dx.doi.org/10.1093/rfs/12.5.937.

[13] Chande, T. S. (1992). Adapting moving averages to market volatility. *Stock & Commodities*, 10:3.

[14] Chande, T. S. & Kroll, S. (1994). The new technical trader. *New York*.

[15] Chandrashekar, G. & Sahin, F. (2014). A survey on feature selection methods. *Comput. Electr. Eng.*, 40(1):16–28. Available from: http://dx.doi.org/10.1016/j.compeleceng.2013.11.024.

[16] Chen, C.-H. (2011). Feature selectionfor unlabeled data. In Tan, Y., Shi, Y., Chai, Y., & Wang, G., editors, *Advances in Swarm Intelligence*, pages 269–274, Berlin, Heidelberg. Springer Berlin Heidelberg.

[17] Chua, S. (2006). *Sammy Chua's Day Trade Your Way to Financial Freedom*. John Wiley & Sons.

[18] Dash, R. & Dash, P. K. (2016). A hybrid stock trading framework integrating technical analysis with machine learning techniques. *The Journal of Finance and Data Science*, 2(1):42 – 57. Available from: http://www.sciencedirect.com/science/article/pii/S2405918815300179.

[19] de Oliveira, F. A., Nobre, C. N., & Zarate, L. E. (2013). Applying artificial neural networks to prediction of stock price and improvement of the directional prediction index - case study of petr4, petrobras, brazil. *Expert Systems with Applications*, 40(18):7596 – 7606. Available from: http://www.sciencedirect.com/science/article/pii/S0957417413004703.

[20] Dempster, M. A. H., Payne, T. W., Romahi, Y., & Thompson, G. W. P. (2001). Computational learning techniques for intraday fx trading using popular technical indicators. *IEEE Transactions on Neural Networks*, 12(4):744–754.

[21] Diebold, F. X., Hahn, J., & Tay, A. S. (1999). Multivariate density forecast evaluation and calibration in financial risk management: High-frequency returns on foreign exchange. *The Review of Economics and Statistics*, 81(4):661–673. Available from: https://doi.org/10.1162/003465399558526.

[22] Dixon, M. (2018). Sequence classification of the limit order book using recurrent neural networks. *Journal of computational science*, 24:277–286.

[23] Dolde, W. (1993). The trajectory of corporate financial risk management. *Journal of Applied Corporate Finance*, 6(3):33–41. Available from: http://dx.doi.org/10.1111/j.1745-6622.1993.tb00232.x.

[24] Ehlers, J. F. (2001). *Rocket science for traders: digital signal processing applications*, volume 112, 2001. John Wiley & Sons.

[25] Elder, A. (2002). *Come into my trading room: A complete guide to trading*, volume 163, 2002. John Wiley & Sons.

[26] Engle, R. F. & Granger, C. W. (1987a). Co-integration and error correction: representation, estimation, and testing. *Econometrica: journal of the Econometric Society*, pages 251–276.

[27] Engle, R. F. & Granger, C. W. J. (1987b). Co-integration and error correction: Representation, estimation, and testing. *Econometrica*, 55(2):251–276. Available from: http://www.jstor.org/stable/1913236.

[28] Eshel, G. (2003). The yule walker equations for the ar coefficients. *Internet resource*, 2:68–73.

[29] Fama, E. F. (1968). Risk, return and equilibrium: some clarifying comments. *The Journal of Finance*, 23(1):29–40.

[30] Fang, Y. & Xu, D. (2003). The predictability of asset returns: an approach combining technical analysis and time series forecasts. *International Journal of Forecasting*, 19(3):369–385.

[31] French, C. W. (2003). The treynor capital asset pricing model. *Journal of Investment Management*, 1(2):60–72.

[32] Gabrielsson, P., Johansson, U., & Konig, R. (2014). Co-evolving online high-frequency trading strategies using grammatical evolution. In *IEEE Conference on Computational Intelligence for Financial Engineering Economics*, pages 473–480.

[33] Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., & Howison, S. D. (2013). Limit order books. *Quantitative Finance*, 13(11):1709–1742.

[34] Gregory-Williams, J. & Williams, B. M. (2012). *Trading chaos: maximize profits with proven technical techniques*, volume 172, 2012. John Wiley & Sons.

[35] Hamilton, J. D. (1994). *Time series analysis*, volume 2, 1994. Princeton university press Princeton.

[36] Hampton, J. J. (1982). *Modern Financial Theory: Perfect and Imperfect Markets*. Reston Publishing Company.

[37] Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501.

[38] Inuiguchi, M. & Tanino, T. (2000). Portfolio selection under independent possibilistic information. *Fuzzy Sets and Systems*, 115(1):83 – 92. Available from: http://www.sciencedirect.com/science/article/pii/S0165011499000263.

[39] Iosifidis, A., Tefas, A., & Pitas, I. (2012). Multidimensional sequence classification based on fuzzy distances and discriminant analysis. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2564–2575.

[40] Iosifidis, A., Tefas, A., & Pitas, I. (2015). On the kernel extreme learning machine classifier. *Pattern Recognition Letters*, 54:11–17.

[41] Iosifidis, A., Tefas, A., & Pitas, I. (2017). Approximate kernel extreme learning machine for large scale data classification. *Neurocomputing*, 219:210–220.

[42] Jensen, M. C., Black, F., & Scholes, M. S. (1972). The capital asset pricing model: Some empirical tests. Available from: https://ssrn.com/abstract=908569.

[43] Kablan, A. (2009). Adaptive neuro-fuzzy inference system for financial trading using intraday seasonality observation model. *World Academy of Science, Engineering and Technology*, 58:479–488.

[44] Kablan, A. & Ng, W. (2010). High frequency trading using fuzzy momentum analysis. In *Proceedings of the World Congress on Engineering*, volume 1.

[45] Keltner, C. W. (1960). *How to make money in commodities*. Keltner Statistical Service.

[46] Kercheval, A. N. & Zhang, Y. (2015). Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, 15(8):1315–1329. Available from: http://dx.doi.org/10.1080/14697688.2015.1032546.

[47] Khaidem, L., Saha, S., & Dey, S. R. (2016). Predicting the direction of stock market prices using random forest. *CoRR*, abs/1605.00003. Available from: http://arxiv.org/abs/1605.00003.

[48] Kohavi, R. & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1):273 – 324. Available from: http://www.sciencedirect.com/science/article/pii/S000437029700043X.

[49] Kumaresan, R. (1990). Identification of rational transfer function from frequency response sample. *IEEE Transactions on Aerospace and Electronic Systems*, 26(6):925–934.

[50] Kwon, Y. K. & Moon, B. R. (2007). A hybrid neurogenetic approach for stock forecasting. *IEEE Transactions on Neural Networks*, 18(3):851–864.

[51] Lintner, J. (1965). The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. *The review of economics and statistics*, pages 13–37.

[52] Liu, H., Sun, J., Liu, L., & Zhang, H. (2009). Feature selection with dynamic mutual information. *Pattern Recognition*, 42(7):1330 – 1339. Available from: http://www.sciencedirect.com/science/article/pii/S0031320308004615.

[53] Lo, A. W., Mamaysky, H., & Wang, J. (2000). Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *The Journal of Finance*, 55(4):1705–1765. Available from: http:https://doi.org/10.1111/0022-1082.00265.

[54] Lubnau, T. & Todorova, N. (2015). Trading on mean-reversion in energy futures markets. *Energy Economics*, 51(Supplement C):312 – 319. Available from: http://www.sciencedirect.com/science/article/pii/S014098831500208X.

[55] Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1):77–91. Available from: http://www.jstor.org/stable/2975974.

[56] Markowitz, H. M. (1968). *Portfolio selection: efficient diversification of investments*, volume 16. Yale university press.

[57] Miao, J. & Niu, L. (2016). A survey on feature selection. In *4th International Conference on Information Technology and Quantitative Management*, pages 919 – 926, 2016.

[58] Mossin, J. (1966). Equilibrium in a capital asset market. *Econometrica: Journal of the econometric society*, pages 768–783.

[59] Mulloy, P. G. (1994). Smoothing data with faster moving averages. *Stocks & Commodities*, 12(1):11–19.

[60] Muranaka, K. (2000). Ichimoku charts. *TECHNICAL ANALYSIS OF STOCKS AND COMMODITIES-MAGAZINE EDITION-*, 18(10):22–31.

[61] Murphy, J. (1999). *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance Series. New York Institute of Finance. Available from: https://books.google.fi/books?id=5zhXEqdr_IcC.

[62] Naiman, E. (2009). Small encyclopedia of trader. *Moscow: Alpina Business Books*, 456.

[63] Ng, A. (2000). Cs229 lecture notes. *CS229 Lecture notes*, 1(1):1–3.

[64] Ntakaris, A., Magris, M., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Benchmark dataset for mid-price prediction of limit order book data. *CoRR*, abs/1705.03233. Available from: http://arxiv.org/abs/1705.03233.

[65] Ntakaris, A., Mirone, G., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2019). Feature engineering for mid-price prediction with deep learning. *IEEE Access*, 7:82390–82412.

[66] Oriani, F. B. & Coelho, G. P. (2013). Evaluating the impact of technical indicators on stock forecasting. In *IEEE Symposium Series on Computational Intelligence*, pages 1–8, 2016.

[67] Pagonidis, A. S. (2014). The ibs effect: Mean reversion in equity etfs. Accessed on 2017-03-17. Available from: http://www.naaim.org/wp-content/uploads/2014/04/00V_Alexander_Pagonidis_The-IBS-Effect-Mean-Reversion-in-Equity-ETFs-1.pdf.

[68] Passalis, N., Tsantekidis, A., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Time-series classification using neural bag-of-features. In *IEEE 25th European Conference of Signal Processing*, pages 301–305, 2017.

[69] Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259 – 268. Available from: http://www.sciencedirect.com/science/article/pii/S0957417414004473.

[70] Perold, A. F. (1984). Large-scale portfolio optimization. *Management Science*, 30(10):1143–1160. Available from: https://doi.org/10.1287/mnsc.30.10.1143.

[71] Poterba, J. M. & Summers, L. H. (1988). Mean reversion in stock prices: Evidence and implications. *Journal of financial economics*, 22(1):27–59.

[72] Rayome, D. L., Jain, A., & Konku, D. (2007). Technical analysis: Donchian channels and the british pound. In *IABE-Annual Conference*, pages 302, 2007.

[73] Richman, J. S. & Moorman, J. R. (2000). Physiological time-series analysis using approximate entropy and sample entropy. *American Journal of Physiology-Heart and Circulatory Physiology*, 278(6):H2039–H2049.

[74] Rodriguez-Gonzalez, A., Garcia-Crespo, A., Colomo-Palacios, R., Iglesias, F. G., & Gomez-Berbis, J. M. (2011). Cast: Using neural networks to improve trading systems based on technical analysis by means of the rsi financial indicator. *Expert Systems with Applications*, 38(9):11489 – 11500. Available from: http://www.sciencedirect.com/science/article/pii/S0957417411004313.

[75] Ross, S. A. (1977). The capital asset pricing model (capm), short-sale restrictions and related issues. *The Journal of Finance*, 32(1):177–183.

[76] Savitzky, A. & Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639.

[77] Schafer, R. W. (2011). What is a savitzky-golay filter? [lecture notes]. *IEEE Signal Processing Magazine*, 28(4):111–117.

[78] Scholtus, M. & van Dijk, D. (2012). High-frequency technical trading: The importance of speed. Available from: http://hdl.handle.net/1765/31778.

[79] Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, 19(3):425–442.

[80] Shen, S., Jiang, H., & Zhang, T. (2012). Stock market forecasting using machine learning algorithms. *Department of Electrical Engineering, Stanford University, Stanford, CA*, pages 1–5.

[81] Sirignano, J. (2016). Deep learning for limit order books. Available from: https://arxiv.org/abs/1601.01987.

[82] Smith, C. W., Smithson, C. W., & Wilford, D. S. (1989). Managing financial risk. *Journal of Applied Corporate Finance*, 1(4):27–48. Available from: http://dx.doi.org/10.1111/j.1745-6622.1989.tb00172.x.

[83] Smith, S. W. (1999). *The scientist and engineer's guide to digital signal processing*. California Technical Pub.

[84] Song, F., Mei, D., & Li, H. (2010). Feature selection based on linear discriminant analysis. In *IEEE Proceedings of the 2010 International Conference on Intelligent System Design and Engineering Application - vol 01*, pages 746–749. Available from: http://dx.doi.org/10.1109/ISDEA.2010.311.

[85] Taylor, S. J. (2008). *Modelling financial time series*. world scientific.

[86] Teixeira, L. A. & de Oliveira, A. L. I. (2010). A method for automatic stock trading combining technical analysis and nearest neighbor classification. *Expert Systems with Applications*, 37(10):6885 – 6890. Available from: http://www.sciencedirect.com/science/article/pii/S0957417410002149.

[87] Thanh, D. T., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Tensor representation in high-frequency financial data for price change prediction. *arXiv:1709.01268*.

[88] Tillson, T. (1998). Better moving averages. Available from: http://www.technicalindicators.net/indicators-technical-analysis/150-t3-movingaverage,[ziureta20160218].

[89] Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017a). Forecasting stock prices from the limit order book using convolutional neural networks. In *IEEE 19th Conference on Business Informatics*, volume 1, pages 7–12, 2017.

[90] Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017b). Using deep learning to detect price change indications in financial markets. In *IEEE 25th European Conference of Signal Processing*, pages 2511–2515, 2017.

[91] Valcu, D. (2004). Using the heikin-ashi technique. *TECHNICAL ANALYSIS OF STOCKS AND COMMODITIES-MAGAZINE EDITION-*, 22(2):16–29.

[92] Wen, Q., Yang, Z., Song, Y., & Jia, P. (2010). Automatic stock decision support system based on box theory and svm algorithm. *Expert Systems with Applications*, 37(2):1015 – 1022. Available from: http://www.sciencedirect.com/science/article/pii/S0957417409005107.

[93] Wilder, J. W. (1978). *New concepts in technical trading systems*. Trend Research.

[94] Wilder Jr, J. W. (1986). The relative strength index ? *J. of Technical Analysis of Stocks and Commodities*, 4:343–346.

[95] Williams, B. (1). *New trading dimensions: how to profit from chaos in stocks, bonds, and commodities*, volume 72, 1998. John Wiley & Sons.

[96] Williams, L. (1985). The ultimate oscillator. *Technical Analysis of Stocks and Commodities*, 3(4):140–141.

[97] Wysocki, A. & Lawrynczuk, M. (2010). An investment strategy for the stock exchange using neural networks. In *Federated Conference on Computer Science and Information Systems*, pages 183–190, 2013.

[98] Zhang, K., Kwok, J. T., & Parvin, B. (2009). Prototype vector machine for large scale semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1233–1240. ACM.

PUBLICATION 3

Feature Engineering for Mid-Price Prediction With Deep Learning
A. Ntakaris, G. Mirone, J. Kanniainen, M. Gabbouj and A. Iosifidis

IEEE Access 7.(2019), 82390–82412

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 28, 2019, accepted June 17, 2019, date of publication June 21, 2019, date of current version July 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2924353

Feature Engineering for Mid-Price Prediction With Deep Learning

ADAMANTIOS NTAKARIS[1], GIORGIO MIRONE[2], JUHO KANNIAINEN[1], MONCEF GABBOUJ[1], (Fellow, IEEE), AND ALEXANDROS IOSIFIDIS[3], (Senior Member, IEEE)

[1]Faculty of Information Technology and Communication Sciences, Tampere University of Technology, FI-33720 Tampere, Finland
[2]Danmarks Nationalbank, 1093 Copenhagen, Denmark
[3]Department of Electrical and Computer Engineering, Aarhus University, 8000 Aarhus, Denmark

Corresponding authors: Adamantios Ntakaris (adamantios.ntakaris@tuni.fi) and Giorgio Mirone (gmi@nationalbanken.dk)

This work was supported by the H2020 Project BigDataFinance MSCA-ITN-ETN under Grant 675044.

ABSTRACT Mid-price movement prediction based on the limit order book data is a challenging task due to the complexity and dynamics of the limit order book. So far, there have been very limited attempts for extracting relevant features based on the limit order book data. In this paper, we address this problem by designing a new set of handcrafted features and performing an extensive experimental evaluation on both liquid and illiquid stocks. More specifically, we present an extensive set of econometric features that capture the statistical properties of the underlying securities for the task of mid-price prediction. The experimental evaluation consists of a head-to-head comparison with other handcrafted features from the literature and with features extracted from a long short-term memory autoencoder by means of a fully automated process. Moreover, we develop a new experimental protocol for online learning that treats the task above as a multi-objective optimization problem and predicts: 1) the direction of the next price movement and; 2) the number of order book events that occur until the change takes place. In order to predict the mid-price movement, features are fed into nine different deep learning models based on multi-layer perceptrons, convolutional neural networks, and long short-term memory neural networks. The performance of the proposed method is then evaluated on liquid and illiquid stocks (i.e., TotalView-ITCH US and Nordic stocks). For some stocks, results suggest that the correct choice of a feature set and a model can lead to the successful prediction of how long it takes to have a stock price movement.

INDEX TERMS Deep learning, econometrics, high-frequency trading, limit order book, mid-price, U.S. data.

I. INTRODUCTION

The automation of financial markets has increased the complexity of information analysis. This complexity can be effectively managed by the use of ordered trading universes like the limit order book (LOB). LOB is a formation that translates the daily unexecuted trading activity in price levels according to the type of orders (i.e., bid and ask side). The daily trading activity is a big data problem, since millions of trading events take place inside a trading session. Information extraction and digital signal (i.e., time series) analysis from every trading session provide the machine learning (ML) trader with useful instructions for orders, executions, and cancellations of trades.

The associate editor coordinating the review of this manuscript and approving it for publication was Bora Onat.

Traditional time series analysis methods have failed to capture the complexity of the contemporary trading markets adequately. For instance, the work in [1] and [2] suggest that classical machine learning and deep learning methods for financial metric predictions achieve better results compared to ARIMA and GARCH models. On the contrary, machine and deep learning methods have proved to be very effective mechanisms for time series analysis and prediction (e.g., [3], [4], [5]). The main advantage of these methods is their ability to capture non-linearities of the input data and filter them consecutively by creating new weighted features more relevant to the suggested problem.

Despite their efficacy to predict time series, machine and deep learning methods are developed mainly through empirical testing. The majority of the literature (e.g., [6], [7], [8]) that focuses on deep learning frameworks solely relies either

on raw data or a limited number of features. So far, very little attention has been paid to the information a neural network should analyze for the mid-price prediction task. In this paper, we shed light on the information that the ML trader should consider utilizing in mid-price movement prediction. To this end, we employ an extensive list of econometric features[1] for mid-price prediction and make a head-to-head comparison with indicators derived from: i) technical and quantitative analysis (i.e., [11]), ii) time-sensitive and time-insensitive features (i.e., [12] and [13]), and iii) features extracted through a fully automated process. This fully automated feature extraction process is conducted by a long short-term memory (LSTM) autoencoder (AE).

We choose econometrics as motivation for our handcrafted features since it is the field of financial engineering that captures the empirical evidence of microstructure noise and causality of the data. Our data comes with variations in prices, known in the financial literature as volatility – a measure that we incorporate into our handcrafted features. Despite the general perception in academic literature that volatility itself is not a factor that affects stock returns, ample evidence exists to support the opposite. For instance, in [14] the author finds that volatility together with other proxies that are not directly observable in the data, like liquidity premium, affect stock returns. In the same direction, Lettau and Ludvigson [15] provide evidence that consumption-to-wealth ratio offers information for excess stock market returns, with volatility explaining a significant portion of these returns. Another example is the work by Chung and Chuwonganant [16], where authors find strong evidence that market volatility affects individual stock returns. Under this light, we believe that these are reliable indicators in considering econometrics as features for the task of mid-price movement prediction.

We perform our analysis based on deep learning models which have recently been proposed for financial time series analysis. These models vary from multi-layer perceptrons (MLP) to convolutional neural networks (CNN) and recurrent neural networks (RNN) like LSTM. For our experiments, we use two TotalView-ITCH datasets from the US and the Nordic stock markets. We formulate these experiments based on two protocols: the first one (i.e., ''Protocol I'' in our experiments) is introduced here for the first time, and is based on online learning. The prediction of the mid-price movement takes place every next event and is treated as a multi-objective optimization problem, since it predicts when and in which direction the mid-price movement will happen. The second protocol (i.e., ''Protocol II'' in our experiments) is an existing protocol based on the work of Tsantekidis *et al.* [17], ,according to which the mid-price movement prediction is treated as a three-class classification problem (i.e., up, down or stationary mid-price states) for every next 10^{th} event.

[1]Econometrics features were used in the past for tasks such as identification of big changes in exchange rate volatility (i.e., [9]), or bankruptcy prediction in [10].

The main contribution of our work lies on three pillars. The first pillar refers to the utilization of an extensive econometric features list as input to deep learning models for mid-price movement prediction. The second pillar is related to an extensive evaluation of the newly introduced features with two other handcrafted feature sets and a feature set based on a fully automated process. We conduct a fair evaluation of these feature sets by using the same nine deep learning models for liquid and illiquid stocks, as well as unbalanced and balanced feature sets. Next, we test them not only on the newly introduced experimental protocol but also on a protocol suggested in the literature for the Nordic dataset (also utilized here). Our findings indicate that handcrafted features, which overperformed the fully automated feature extraction process (i.e., based on LSTM AE), transform the forecasting universe of high-frequency trading. More specifically, the present evaluation facilitates traders' task of selecting suitable features according to data, stock, and model availability. The third pillar, finally, refers to the development of a new experimental protocol that takes into consideration every trading event and is unaffected by time irregularities in high-frequency data. Our work suggests that feature extraction should be customized according to stock and model selection; similar findings can be in seen in [18]. The present research opens avenues for several other applications. For instance, the same sets of features can be tested for time series such as exchange rates or bitcoin price predictions. Furthermore, the newly introduced protocol can be the basis of every time series problem since it is event-driven and unaffected by time irregularities. Ultimately, there is no need for any type of data sampling, even for high-frequencies time resolution environments where datasets are massive.

The remainder of the paper is organized as follows. We provide a comprehensive literature review in Section II. The problem statement is provided in Section III. The list of handcrafted features follows in Section IV. In Section V, we describe the various deep learning models adopted in our analysis, while in Section VI we describe details of the datasets and the experimental protocol. In Section VII we provide the empirical results and Section VIII concludes the paper. A detailed description of the econometric features used in our experiments are provided in Appendix together with results for Protocol II.

II. LITERATURE REVIEW

High-frequency LOB data analysis has captured the interest of the machine learning community. The complex and chaotic behavior of the data inflow gave space to the use of non-linear methods like the ones that we see in the machine and deep learning. For instance, Zhang *et al.* [19] utilize neural networks for the prediction of Baltic Dry index and provide a head-to-head comparison with econometric models. The author in [20] develops a new type of deep neural network that captures the local behavior of a LOB for spatial distribution modeling. Dixon applies RNN [21] on S&P500 E-mini futures data for a metric prediction like price change

forecasting. Minh *et al.* [22] also propose RNN architecture for short-term stock predictions by utilizing successfully financial news and sentiment dictionary. In [23], authors apply a combined neural network model based on CNN and RNN for mid-price prediction.

Metrics prediction, like mid-price, can be facilitated by the use of handcrafted features. Handcrafted features reveal hidden information as they are capable of translating time-series signals to meaningful trading instructions for the ML trader. Several authors worked towards this direction, like [12], [24], [13], [25], [26], [27] and [20]. These works present a limited set of features which varies from raw LOB data to change of price densities and imbalance volume metrics. Another work that provides a wider range of features is presented by Ntakaris *et al.* [11]. The authors there extract handcrafted features based on the majority of the technical indicators and develop a new quantitative feature based on logistic regression, which outperformed the suggested feature list.

Handcrafted features represent only one part of the experimental protocol in the quest for mid-price movement prediction. Classification, via deep learning methods, is the continuation of a machine learning protocol. Many authors have used deep learning in financial literature for several problems. For example, Alberg and Lipton [28] use MLPs and RNNs for companies' future fundamentals forecasting. Qian [1] utilizes machine and deep learning methods, like support vector machines (SVM), MLPs, denoising auto-encoder (DAE), and an assembled DAE-SVM model in order to predict future trends of stock's index prices. These machine and deep learning models outperformed traditional time series models like ARIMA and generalized autoregressive conditional heteroskedasticity (GARCH). Sezer *et al.* [29] use MLPs and the three most commonly used technical indicators as inputs for stock price movement predictions.

Many authors utilize LOB data as input to their models. For instance, Nousi *et al.* [4] examine the performance of several machine learning methods, like autoencoders (AE), bag-of-features algorithm, single hidden layer feedforward neural networks (SLFN), and MLPs for mid-price prediction. Han *et al.* [30] apply decision trees on LOB data and outperform support vector machines (SVM) for the problem of mid-price prediction. In the same direction, authors in [31] apply similar methods on market order book data for market movement predictions. Doering *et al.* [32] utilize event flow and limit order datasets for price-trend and price-volatility predictions based on a deep learning architecture. Mäkinen *et al.* [33] predict price jumps with the use of LSTM, where the input data is based on LOB data. A similar work, in terms of the neural model, is conducted in [34] in order to forecast LOB's mid-price.

To the best of our knowledge, this is the first time that an extensive list of econometric features based on high-frequency LOB data is proposed as input to several neural networks for mid-price prediction. We conduct a head-to-head comparison with state-of-the-art handcrafted features is conducted together with features based on a fully automated process; Finally, we report results extracted from two high-frequency datasets with two US and five Nordic stocks for both balanced and unbalanced sets.

TABLE 1. Message list example.

Timestamp	Id	Price	Quantity	Event	Side
1275386347944	6505727	126200	400	Cancellation	Ask
1275386347981	6505741	126500	300	Submission	Ask
1275386347981	6505741	126500	300	Cancellation	Ask
1275386348070	6511439	126100	17	Execution	Bid
1275386348070	6511439	126100	17	Submission	Bid
1275386348101	6511469	126600	300	Cancellation	Ask

TABLE 2. Order book example.

			Level 1				...
			Ask		Bid		
Timestamp	Mid-price	Spread	Price	Quantity	Price	Quantity	
1275386347944	126200	200	126300	300	126100	17	...
1275386347981	126200	200	126300	300	126100	17	...
1275386347981	126200	200	126300	300	126100	17	...
1275386348070	126050	100	126100	291	126000	2800	...
1275386348070	126050	100	126100	291	126000	2800	...
1275386348101	126050	100	126100	291	126000	2800	...

III. PROBLEM STATEMENT

The problem under consideration is the mid-price movement prediction based on high-frequency LOB data. More specifically, we use message and limit order books as input for the suggested features. Message book (MB), as seen in Table 1, contains the flow of information which takes place at every event occurrence. The information displayed by every incoming event includes the timestamp of the order, execution or cancellation, the id of the trade, the price, the volume, the type of the event (i.e., order, execution or cancellation), and the side of the event (i.e., ask or bid).

LOB (Table 2) works under specific rules based on the operation of the trading system-exchange. The main advantage of an order book is that it accepts orders under limits (i.e., limit orders) and market orders. In the former case, the trader/broker is willing to sell or buy a financial instrument under a specific price. In the latter case, the action of buying or selling a stock at the current price takes place. LOBs accept orders by the liquidity providers who submit limit orders and the liquidity takers who submit market orders. These limit orders, which represent the unexecuted trading activity until a market order arrives or cancellation takes place, construct the LOB that is divided into levels. The best level consists of the highest bid and the lowest ask price orders, and their average price defines the so-called mid-price, whose movement we try to predict.

We treat the mid-price movement prediction as a multi-objective optimization problem with two outputs – one is related to classification and the other one to regression. The first part of our objective is to classify whether the mid-price will go up or down and the second part – the regression part is to predict in how many events in the future this movement will happen. To further explain this, let us consider the following example: in order to extract the intraday labels, we measure

starting from time t_k, in how many events the mid-price will change and in which direction (i.e., up or down). For instance, the mid-price will change in 10 events from now, and will go up. This means that our label at time k is going to be $\{1,10\}$, where 1 is the direction of mid-price and 10 is the number of events that need to pass in order to see that movement taking place.

We depart from this labeling system to answer the critical question of whether handcrafted features derived from econometrics can boost deep learning classification and regression performance. We conduct extensive experiments based on nine neural topologies (i.e., five MLPs, two CNNs, and two LSTMs) and two TotalView-ITCH datasets, and compare the performance of econometric features to three other feature sets. The first set is based on time-sensitive and time-insensitive features as presented in [12] and [13], the second feature set is based on technical and quantitative analysis, introduced in [11], and the third one is based on feature representations extracted automatically for the train of an LSTM AE with a description provided in Section V-D .

IV. HANDCRAFTED FEATURE POOL

In this section we provide the nominal list (see Table 3) of the extensive econometric feature list together with the two other state-of-the-art handcrafted feature sets from the literature that are based on technical and quantitative analysis and time-insensitive and time-sensitive indicators. Description of the econometric features is seen in Appendix while the description of technical and quantitative feature set and time-sensitive and time-insensitive set extracted from the LOB can be found in [11].

We extract our econometric features from both MB and LOB and divide them into four main categories: Statistical features, volatility measures, noise measures, and price discovery features. The first category encompasses basic statistical features that are widely used in the literature (e.g., [12], [20]). The logic behind the choice of the volatility measure features is the intimate relation between the volatility of the price process and the price movement itself. As such, we regard the volatility measures included in the present article to retain information useful to real-time price prediction. This is particularly true when the predicted objective is the next price movement. Additionally, the econometric literature widely evidences the significant detrimental impact of the so-called microstructure noise in the measurement of fundamental quantities when working at the highest frequencies. Furthermore, the noise process directly affects the underlying price process itself and as such contributes to the observed price movements. For these reasons we implement a number of estimates of the characteristics of the noise process, which we identify as the noise measures features set.[2] The last group of features includes all those features related to the price discovery process; i.e., those that take into account the interaction of the two sides of the LOB. Several articles in the literature (e.g., [11], [33]) have focused and demonstrated the importance of accounting for the differences between the ask and bid side in order to improve the mid-price forecasting accuracy.

Each of the features in Table 3 operates under a different time duration. Time duration of the features plays an important role in capturing information about underline behavior of time series. More specifically, the feature extraction process consists of low frequency (e.g., technical indicators based on interpolation) and high-frequency features (e.g., adaptive logistic regression), which complement each other. Low frequency features identify long-term trends and structural data components, while high-frequency features capture discontinuities and rapid metric changes. This combination of features facilitates improves neural network perfromance (e.g., [12] and [38]).

[2]Most of the presented measures have been developed and are consistent estimators under broad assumptions on the underlying price process and contaminating noise process; we will not discuss these assumptions into details here as outside the scope of the article. Interested readers are referred to [37] and references within for an exhaustive review of the literature

V. DEEP LEARNING

The goal of this paper is to forecast the movement of the mid-price. The predicted output has dual information: the direction of the mid-price movement and the prediction of the number of events taking the mid-price to move up or down. An efficient way to do that is by using deep learning architectures. We consider three different neural networks types (i.e., MLPs, CNNs, and LSTMs) and run them seperately. We, then, examine their validity with respect to our optimization problem.

A. MLP FOR CLASSIFICATION AND REGRESSION

MLP (i.e., [39]) is a type of neural network that shows a high degree of connectivity among its components/neurons (see Fig. 1). The strength of this connectivity is determined by the synaptic weights of the neural network. These synaptic weights are determined by a differentiable nonlinear activation function. These basic characteristics of the neural network complicate the analysis of MLPs' behavior. As a result, several MLP architectures have to be examined in order to see whether input data (i.e., handcrafted features) affect the outcome/prediction. The way that an MLP can be trained is based on a sequential data feeding process called batch learning. Batch learning is a process according to which the neural network adjusts the synaptic weights after the presentation of all the samples $J = \{x(i), d(i)\}_{i=1}^{N}$ in the training process, where $x(i)$ is the input multi-dimensional vector and $d(i)$ the response vector of the supervised problem at instance i, and the error function at instance i is:

$$e^{(i)} = d^{(i)} - y^{(i)} \tag{1}$$

where $d^{(i)}$ is the i^{th} element of the $d(i)$ and $y^{(i)}$ is the produced output term at instance i. The error function that we use for our experiments is bespoke to our supervised problem and its components are based on the binary cross entropy (for the classification task) and the mean squared error (for the

TABLE 3. Feature list for the three feature sets: Description for the newly introduced, based on Econometrics, handcrafted features can be found in Appendix, where description for the Tech & Quant and LOB feature sets can be found in [11].

Econometric features	Tech & Quant features	LOB features
Statistical Features	**Technical Indicators**	**Basic**
Mid-Price	Accumulation Distribution Line	n LOB Levels
Financial Duration	Awesome Oscillator	
Average Mid-Price Financial Duration	Accelerator Oscillator	**Time-Insensitive**
Log-Returns	Average Directional Index	Spread & Mid-Price
	Average Directional Movement Index Rating	Price Differences
Volatility Measures	Displaced Moving Average	Price& Volume Means
Realized Volatility	Absolute Price Oscillator	Accumulated Differences
Realized Kernel	Aroon Indicator	
Realized Pre-Averaged Variance	Aroon Oscillator	**Time-Sensitive**
Realized Semi-Variance	Average True Range	Price & Volume Derivation
Realized Bipower Variation	Bollinger Bands	Average Intensity per Type
Realized Bipower Variation (lag 2)	Ichimoku Clouds	Relative Intensity Comparison
Realized Bipower Semi-Variance	Chande Momentum Oscillator	Limit Activity Acceleration
Jump Variation	Chaikin Oscillator	
Spot Volatility	Chandelier Exit	
Average Spot Volatility	Center of Gravity Oscillator	
	Donchian Channels	
Noise and Uncertainty Measures	Double Exponential Moving Average	
Realized Quarticity	Detrended Price Oscillator	
Realized Quarticity Tripower	Heikin-Ashi	
Realized Quarticity Quadpower	Highest High and Lowest Low	
Noise Variance [35]	Hull MA	
Noise Variance [36]	Internal Bar Strength	
	Keltner Channels	
Price Discovery Features	Moving Average Convergence/Divergence Oscillator	
Weighted Mid-Price by Order Imbalance	Median Price	
Volume Imbalance	Momentum	
Bid-Ask Spread	Variable Moving Average	
Normalized Bid-Ask Spread	Normalized Average True Range	
	Percentage Price Oscillator	
	Rate of Change	
	Relative Strength Index	
	Parabolic Stop and Reverse	
	Standard Deviation	
	Stochastic Relative Strength Index	
	T3-Triple Exponential Moving Average	
	Triple Exponential Moving Average	
	Triangular Moving Average	
	True Strength Index	
	Ultimate Oscillator	
	Weighted Close	
	Williams %R	
	Zero-Lag Exponential Moving Average	
	Fractals	
	Linear Regression Line	
	Digital Filtering: Rational Transfer Function	
	Digital Filtering: Savitzky-Golay Filter	
	Digital Filtering: Zero-Phase Filter	
	Remove Offset and Detrend	
	Beta-like Calculation	
	Quantitative Indicators	
	Autocorrelation	
	Partial Correlation	
	Cointegration based on Engle-Granger test	
	Order Book Imbalance	
	Logistic Regression for Online Learning	

regression task), as follows:

$$\mathcal{L}_{all} = \underset{\mathcal{L}_1,\mathcal{L}_2}{arg\,min}\,\{\lambda\mathcal{L}_1 + (1-\lambda)\mathcal{L}_2\} \tag{2}$$

where $\mathcal{L}_1 = -t\ log\ \hat{y}^{(i)} - (1-t)\ log(1-\hat{y}^{(i)}), t \in \{0,1\}$ and $\mathcal{L}_2 = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$ with a free parameter λ, where $y^{(i)}$ and $\hat{y}^{(i)}$ be the ground truth and the predicted values of the i^{th} training sample which belongs to $\mathbb{R}^N$, respectively. This customized function is part of the backpropagation algorithm that helps the neural network (e.g. MLP) to correct the synaptic weights in order to optimize Eq. 2. Backpropagation in our case follows the automatic differentiation (AD) reverse mode (i.e., [40]). Reverse AD facilitates the process of correcting the synaptic weights and it can be done as follows: Initially we define the input variables as $v_{i-n} = x_i, i = 1, ..., n$, all the intermediate variables of the neural network as $v_i, i = 1, ..., k$ and $y_{m-i} = v_{k-i}, i = m-1, ..., 0$ be the output variables. Derivatives calculation is a two-step process. During the first phase the intermediate variables v_i are populated and create the graph trace, whereas during the second phase derivatives are calculated based on the propagation of the adjoints $\bar{v}_i = \frac{\partial y_l}{\partial v_i}$. In general, the reverse AD performs the calculations from the output to the input starting from the output as seed:

$$\frac{\partial f}{\partial y_{m-i}} \leftarrow 1$$

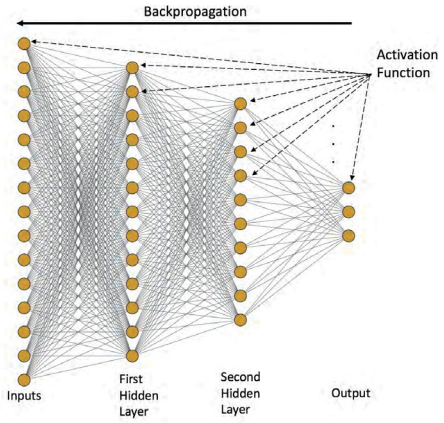


FIGURE 1. Example of an MLP neural network with two hidden layers and 4 units output.

and moves to the inputs via the intermediate states based on the calculation:

$$x_i \leftarrow \sum_{j:i\in Pa(j)} \frac{\partial f}{\partial x_j}\frac{\partial g_j}{\partial x_i}$$

where $Pa(j)$ denotes the parent formation of node j and g_j the intermediate functions of the graph. The next part of the MLP training is the learning process, which is defined as the method through which the loss function will reach the optimal solution via proper parameter updates. For this reason we choose the Nesterov accelerated gradient (NAG) method incorporated into the adaptive moment estimation (Adam) method named as Nadam by Dozat [41]. Nadam applies the momentum step only once and takes into consideration the current momentum – rather than the previous momentum – vectors. This gives us the Nadam update parameters rule:

$$\theta_{t+1} := \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}(\beta_1\hat{m}_t + \frac{(1-\beta_1)\nabla_{\theta_t}\mathcal{L}_{all}(\theta_t)}{1-\beta_1^t}) \quad (3)$$

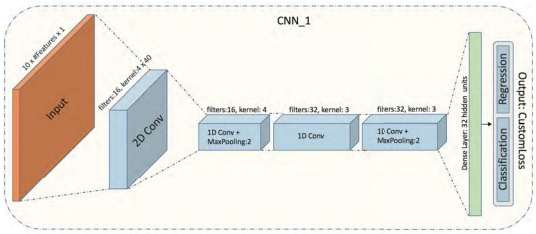
where the first (i.e., mean) and second (i.e., variance) moment for the current momentum vector are, respectively:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{u}_t = \frac{v_t}{1-\beta_2^t} \quad (4)$$

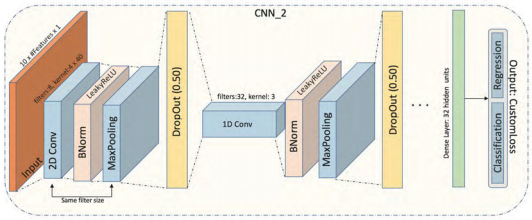
with $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_{\theta_t}\mathcal{L}_{all}(\theta_t)$, $v_t = \beta_2 v_{t-1} + (1-\beta_1)\nabla^2_{\theta_t}\mathcal{L}_{all}(\theta_t)$ and learning rate $\eta = 0.002$.

B. CNN FOR CLASSIFICATION AND REGRESSION

CNN, as described in [42], is a type of neural network that handles time series of multidimensional data for metric prediction. The main motivation for choosing this type of neural network is its capability for sparse connectivity between neural layers, for sharing the so-called tied weights and equivariant representation properties. More specifically, sparse connectivity can be achieved by using a kernel smaller than the sample input. This action reduces the amount of memory that is required for the training process. The second



(a) CNN based on the work presented in [43].



(b) CNN with deeper topology that will be used later in the experimental protocols.

FIGURE 2. Two CNN examples that demonstrate their operation mechanisms. These two CNNs (i.e., CNN_1 and CNN_2) will later on utilized in the experiments.

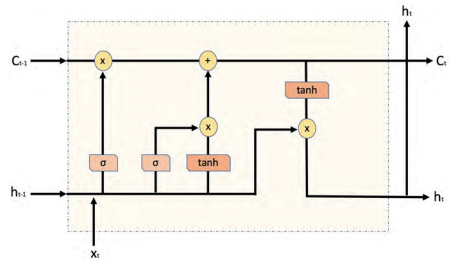


FIGURE 3. Visual representation of LSTM's internal cell calculations.

advantage of a CNN is the use of tied weights. Tied weights are shared among the inputs since the same amount of weights is applied to the inputs.

CNN has three main parts: the convolution layer, the pooling layer, and the fully connected layer. The convolution layer extracts features from the input multi-dimensional signal expressed usually as a tensor or matrix. This process creates linear activations that run via a non-linear activation function such as the rectified linear activation function (ReLU) and the Leaky ReLU. Then the pooling layer will convert the local output based on a summary statistic related to the local outputs (e.g. max-pooling). The last step of the process is the connection to the fully connected layers (see examples in Fig. 2) that will perform the classification and regression tasks. These tasks are based on discrete time series events that formulate the (forward) convolution layer calculation as follows:

$$y_{i^{l+1},j^{l+1},d} = \sum_{i=0}^{H}\sum_{j=0}^{W}\sum_{d=0}^{D} f_{i,j,d} \times x^l_{i^{l+1}+i,j^{l+1}+j,d} \quad (5)$$

TABLE 4. List of the nine deep learning models that are used for the two experimental protocols. Output, in the neural networks above, means that for Protocol I the output is a dense layer with 1 unit and linear activation function for the regression task and a dense layer with two units and softmax activation function. For Protocol II, the output is a dense layer with three units and softmax activation function.

Model	Topology
MLP_1	• Dense layer with 4 units with Tanh activation • Output
MLP_2	• Dense layer with 512 units and Tanh activation • 20% Dropout • Dense layer with 256 units and Tanh activation • Output
MLP_3	• Dense layer with 256 units and Tanh activation • 20% Dropout • Dense layer with 256 units and Tanh activation • Output
MLP_4	• Dense layer with 256 units and Tanh activation • 20% Dropout • Dense layer with 256 units and Tanh activation • 20% Dropout • Dense layer with 256 units and Tanh activation • Output
MLP_5	• Dense layer with 128 units and Tanh activation • 20% Dropout • Dense layer with 128 units and Tanh activation • 20% Dropout • Dense layer with 128 units and Tanh activation • Output
CNN_1	• 2D Convolution layer with 16 filters, 4 x 40 kernel size • 1D Convolution layer with 16 filters, 4 as kernel size • Maxpooling size of 2 • 1D Convolution layer with 32 filters, 3 as kernel size • 1D Convolution layer with 32 filters, 3 as kernel size • Maxpooling size of 2 • Dense layer with 32 neurons • Output
CNN_2	• 2D Convolution layer with 8 filters, 4 x 40 kernel size, 2 x 2 stride size and same output size • BatchNormalization • LeakyReLU with 0.1 slope • 2 x 2 MaxPooling with the same output size • 50% Dropout • 1D Convolution layer with 32 filters, 3 as kernel size, 5 as stride and same output size • BatchNormalization • LeakyReLU with 0.1 slope • Maxpooling size of 2 with the same output size • 50% Dropout • 1D Convolution layer with 32 filters, 3 as kernel size, 5 as stride and same output size • LeakyReLU with 0.1 slope • Maxpooling size of 2 with the same output size • 50% Dropout • Dense layer with 8 units • Output
LSTM_1	• LSTM layer with 32 units • Dropout • PReLU • Dense layer with 64 units • Output
LSTM_2	• LSTM layer with 40 units • PReLU • Attention layer • Dense layer with 40 units • Output

where H, D,and D are the row, columns and depth dimension of the input tensor $x \in \mathbb{R}^{H^l \times W^l \times D^l}$ respectively, $f \in \mathbb{R}^{H^l \times W^l \times D^l}$ is the filter bank, and the indexing $(i^{l+1}+i, j^{l+1}+j, d)$ refers to the iterative local convolution of the filter bank on the suggested input for the l-layer. Pooling is performed right after convolution; to conduct our experiments, we choose the formation of max pooling. The final step is the use of fully connected layers. The structure of these fully connected layers is the same as in Sec. V-A. The process that we follow in order to train our CNN parameters (i.e., filter banks and synaptic/tied weights) is based on batch learning combined with reverse AD (i.e., backpropagation) as we did for the MLP case.

C. LSTM FOR CLASSIFICATION AND REGRESSION

The ML trader has to consider the temporal behavior of time series. The events that we have to deal with in the LOB universe are likewise formed in a sequential manner. Sequential systems, like RNNs, are based on computational graphs and are, thus, ideal for time series analysis. RNNs provide much flexibility in terms of architecture formation, which is described in Eq. 6:

$$h_t = f(h_{t-1}, x_t; \theta) \tag{6}$$

where h and x are the state and the input at time t and θ are the shared parameters for a transition function f at time t. Since we use RNN for empirical calculations we

choose to forecast mid-price by using gated RNNs (named LSTM) as presented in [44]. Motivation for choosing this type of gated RNN is its ability to create connections through time and account for the problem of vanishing (or exploding) gradients. Instead of applying just element-wise nonlinear input transformations, LSTM units (see LSTM's internal cell calculations in Fig. 3), contain processes which that take into consideration the sequential nature of time series. More specifically, an LSTM cell is equipped with gates that filter the information flow by applying weights internally. The first pass is the forget gate vector f_i^t:

$$f_i^{(t)} = \sigma\Big(\sum_j W_{i,j}^f h_j^{(t-1)} + \sum_j U_{i,j}^f x_j^{(t)} + b_i^f\Big) \tag{7}$$

where $x_i^{(t)}$ and $h_i^{(t)}$ are the current input and hidden state vectors of cell i at time t, respectively. The attached weight matrices to these vectors are W^f and U^f for the forget gate vector with b^f the bias term. The next pass is related to the information to be saved to the so-called "cell state". The cell state can be divided in two parts - the input vector and a *tanh* layer as follows:

$$C_i^{(t)} = f_i^{(t)} C_i^{(t-1)} + g_i^{(t)} \sigma\Big(\sum_j W_{i,j}^C h_j^{(t-1)} + \sum_j U_{i,j}^C x_j^{(t)} + b_i^C\Big) \tag{8}$$

where $g^{(t)}$ is the input gate:

$$g_i^{(t)} = \sigma\Big(\sum_j W_{i,j}^g h_j^{(t-1)} + \sum_j U_{i,j}^g x_j^{(t)} + b_i^g\Big) \tag{9}$$

The last remaining part is the filtered output. More specifically, the LSTM output/hidden state will be formulated by the output gate vector $o_i^{(t)}$ which is calculated as follows:

$$o_i^{(t)} = \sigma\Big(\sum_j W_{i,j}^o h_j^{(t-1)} + \sum_j U_{i,j}^o x_j^{(t)} + b_i^o\Big) \tag{10}$$

and the final output $h_i^{(t)}$ is equal to:

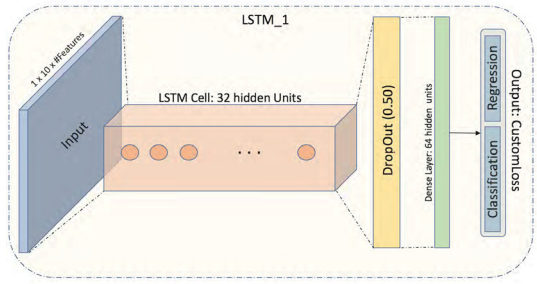
$$h_i^{(t)} = o_i^{(t)} * tanh(C_i^{(t)}). \tag{11}$$

The formation above refers to the case of a typical LSTM neural network, which we implement in Section VI. We also apply an attention mechanism to the LSTM architecture in order to weight/measure the significance of the input sequence. We follow the implementation in [45] and [33] where the sequential LSTM outputs (i.e., hidden states $H^{(t)}$, $t \in \{1, ..., T\}$) are filtered via the following steps for every K-dimensional vector w:

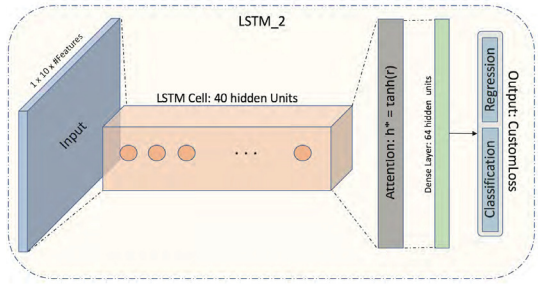
$$M = tanh(H^{(t)}) \tag{12}$$

$$\alpha = \frac{e^{w_i^T * M}}{\sum_{k=1}^{K} e^{w_k^T * M}} \tag{13}$$

$$r = H^{(t)} * \alpha \tag{14}$$



(a) LSTM based on the work in [17]



(b) LSTM with attention layer

FIGURE 4. Two LSTM examples with one main LSTM block (orange colored box) with several hidden cell units (orange cycles). These two LSTMs (i.e., LSTM_1 and LSTM_2) will later on utilized in the experiments.

and the final LSTM with attention output is:

$$h^* = tanh(r). \tag{15}$$

Here we use the same backpropagation mechanism as we did for MLPs. Examples of LSTM neural networks can be seen in Fig. 4

D. FULLY AUTOMATED FEATURE EXTRACTION BASED ON AUTOENCODERS

Autoencoders (AE) (i.e., [46], [47]) are neural networks which operate on a self-feedback loop fashion. They do not require any labeling system since they depend on this semi-supervised protocol. This type of neural network is divided in three main parts; the encoder, the latent representation, and the decoder (i.e. encoder and decoder). An example of AE can be seen in Fig. 5.

The basic structure of AE is defined as a mapping from encoder to decoder, the main objective being the following minimization problem:

$$f, g = \underset{f,g}{arg\,min}\ ||X - (f \circ g)X||^2 \tag{16}$$

where $f : X \to F$ and $g : F \to X$ with X be the input raw LOB data in the present work.

The fully automated feature extraction process is based on the latent representation. This latent representation in

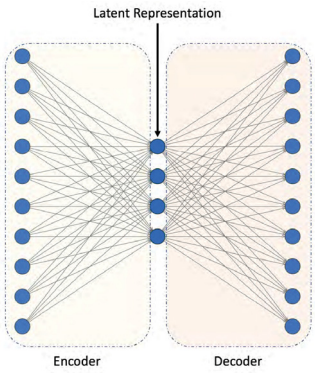


FIGURE 5. AE Example.

the present work plays the role of the vector representation, which will, later on act as input to each of the suggested nine deep neural networks. In order to use this latent space as feature set we train an LSTM AE.[3] LSTM AE has exactly the same structure as a simple AE with the difference that the filtering is based on LSTM layers for the encoding and decoding part. We choose LSTM AE since they take into consideration the temporal behavior of our time series.

VI. DATA DESCRIPTION AND EXPERIMENTAL PROTOCOLS

Our objective is to provide informative handcrafted features to ML traders and market makers for the task of mid-price movement prediction. Prediction of this movement requires in-depth analysis in terms of data selection (e.g., liquid or illiquid stocks) and experimental protocol development. For these reasons, our analysis consists of two TotalView-ITCH based on two US and five Nordic stocks and two experimental protocols. The first protocol, named Protocol I, is based on online prediction for every 10-block rolling events, and we introduce it here for the first time. The second protocol, named Protocol II, is derived from the literature (i.e., [17]) and is based on mid-price movement prediction with 10-event lag. Both protocols are event driven, which means that there are no-missing values. However, Protocol II is based on independent 10-block events, which creates a lag of 10 events. Some of the suggested features can partially overcome this problem by finding averages or other types of transformations inside these blocks, but, still some information will be parsed. A possible solution to this problem comes from Protocol I where every single trading event is taken into consideration and, as a result, there are no missing values. We should also mention that LOB data is exposed to bid-ask[4] bounce effect which may inject bias. We leave this topic for future research, where we plan to increase the rolling event block size in Protocol I since a wider block will, potentially, improve stability.

A. DATA

We utilize two TotalView-ITCH datasets based on two US and five Nordic stocks. The time resolution of the datasets is in milliseconds. For the US datasets, we use two stocks, Amazon and Google, whereas for the Nordic dataset we use Kesko Oyj, Outokumpu Oyj, Sampo Oyj, Rautaruukki, and Wartsila Oyj. We use ten business days for both datasets covering the periods: from 22.09.15 to 05.10.15 for the US dataset and from 01.06.10 to 14.06.10 for the Nordic dataset, respectively. The trading activity for these ten business days is 13,000,000 events for the US dataset and 4,000,000 events for the Nordic dataset. We use MBs in order to create relevant LOBs. We utilize super clustering computational power based on HP Apollo 6000 XL230a/SL230s supercluster to convert MBs to LOBs (i.e., LOBs are of depth 10 for both sides). We follow several pre-processing steps before we start training the deep learning models. A general description of the pre-processing process can be seen in Fig. 6

B. PROTOCOL I

Both TotalView-ITCH datasets convey asynchronous information varying from events taking place at the same millisecond to events several minutes apart from each other. In order to address this issue, we develop Protocol I, which utilizes all the given events in an online manner. More specifically, our protocol extracts feature representation every ten events with an overlap of nine events for every next feature representation. We decided to use a 10-window block for our experiments due to the frequency [5] of the stationarity present in both datasets. In order to identify whether our time series have unit roots, we perform an Engle-Granger cointegration test,[6] with focus on the augmented Dickey-Fuller test, on the pair $Ask - Bid$ prices from LOBs level I. The hypothesis test shows that there is a constant alternation between its states (i.e. 1 for non-stationarity and 0 for stationarity of the suggested time series), which occurs several times during the day. This is indicative for both datasets as seen in Figure 7. These stationarity breaks supports the initial idea, as this presented by many authors (e.g., [48], [49], [50]), that neural networks are capable of identifying underlying processes of a non-stationary time series. Neural networks are nonlinear and non-parametric adaptive-learning filters which operate with fewer assumptions compare to more traditional time series models like ARIMA and GARCH.

A visual description of our protocol can be seen in plot (a) in Fig. 8. The problem under consideration in Protocol I is to predict the movement of mid-price (i.e., classification: up or down) together with the number of events it takes for that movement to occur in the future (i.e., regression:

[3]Details of the training are provided in Section VII.

[4]Bid-ask bounce is the rapid stock's price bounce between bid and ask side.

[5]The average rate of change of the non-stationarity condition, for both TotalView-ITCH datasets, is changing in average every ten events.

[6]Test implementation can be found in [11].

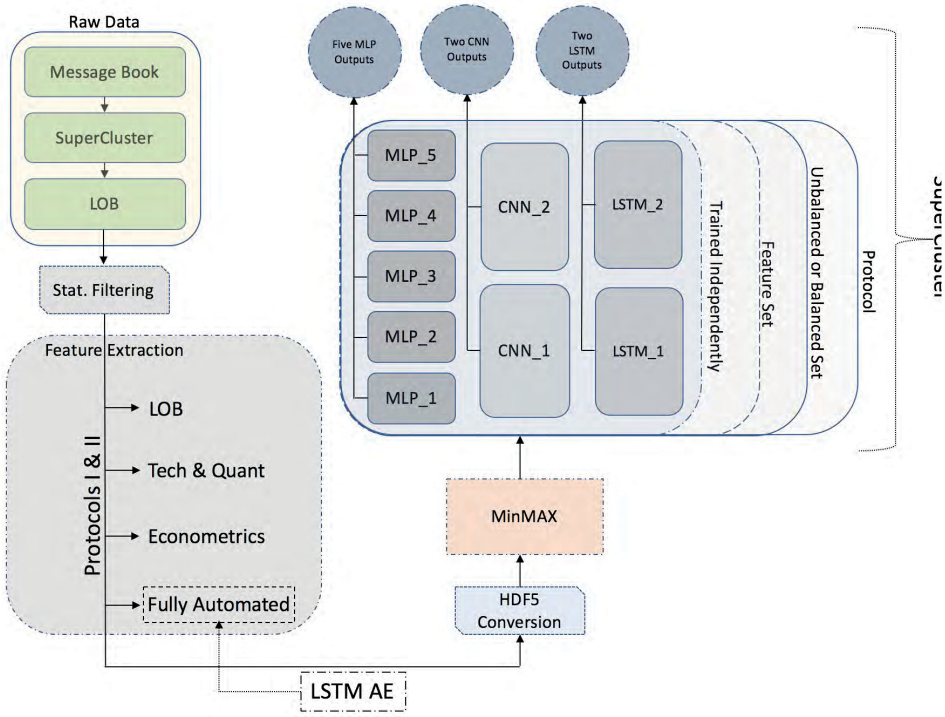


FIGURE 6. This is a higher-level explanation of the steps that we follow for the present analysis. From left top to right bottom: The first step is to obtain the datasets for the US and Nordic stocks and send raw data (i.e., message books) to CSC superclusters and obtain the LOBs. The next step is to apply statistical filtering (description can be found in Section VI-D). What follows is the process of feature extraction for the four different feature sets for Protocols I & II. An HDF5 conversion takes place right afterwards, and a MinMax normalization follows for every feature set case for both protocols. Next, each of the nine neural networks is trained independently for the four different feature lists based on unbalanced and balanced sets. The training process is based on python scripts, which are sent to CSC superclaster in order to obtain results for Protocol I & II.

number of events until next mid-price's movement change). More specifically, in order to testing performance evaluation, we utilize f1 score for the classification task and RMSE (i.e., Root Mean Square Error) for the regression task. F1 score is defined as:

$$f1 = \frac{2 \times Recall \times Precision}{Recall + Precision}, \tag{17}$$

with

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

and

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

where TP, FN, and FP are the True-Positives, False-Negatives, and False-Positives, respectively, and RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}}, \tag{20}$$

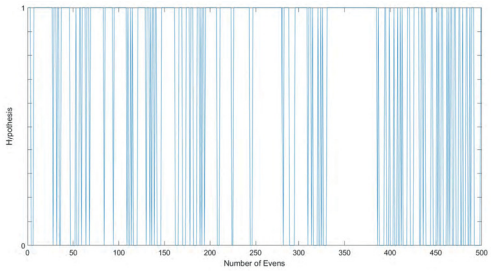
where P_i and O_i are the predicted and observed values of n samples, respectively.

We have a labeling system that requires classification and regression. The first part of the dual labeling format contains the binary information 1 and -1 for the up and down mid-price movement, respectively. The second part of the labeling format represents the discretization of the numeric data expressed as the steps until the next mid-price change. A pictorial example of the above labeling system is in Fig. 9. The label extraction is described as follows:

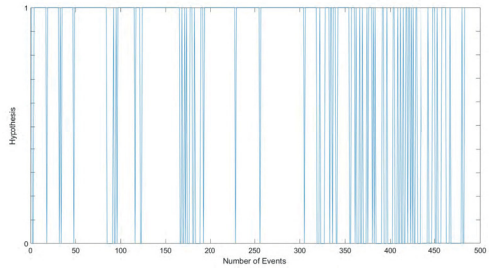
1) $d(i) = 1_{MP_{(i)} - MP_{(i-1)} > 0}$ OR $-1_{MP_{(i)} - MP_{(i-1)} < 0}$, where $i \in \mathbb{R}^{N-1}$, with N be the number of the mid-prices (MP) samples,
2) $L(p) \leq d(i) < L(p+1)$, $1 \leq p < Q$, where $L(p)$ is a vector that contains the bin limits in a monotonically increasing order and Q is the number of bins equal to the total number of the non-zero elements in the vector of mid-price differences.

C. PROTOCOL II

Protocol II is based on independent 10-event blocks for the creation of the feature representations as this can be seen in



(a) Hypothesis test for stationarity check for the Nordic stock, Kesko Oyj. The plot represents a sample of 500 consecutive events.



(b) Hypothesis test for stationarity check for the US stock, Amazon. The plot represents a sample of 500 consecutive events.

FIGURE 7. Hypothesis test for stationarity check, where constant transition from state 0 to state 1 is present.

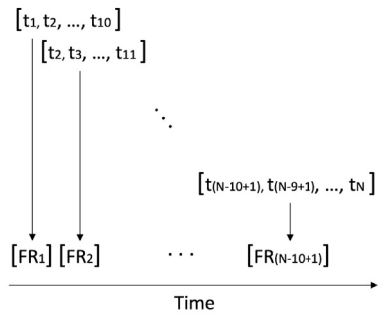
the plot (b) in Fig. 8. More specifically, feature representations are based on the information that can be extracted from 10 events each time with these 10-event blocks independent from each other. Protocol II treats the problem of mid-price movement prediction as a three-class classification problem, with three states: up, down, and stationary condition for the mid-price movement. These changes in the mid-price are defined by means of the following calculations:

$$l_t = \begin{cases} 1, & \text{if } \dfrac{m_a(t)}{MP(t)} > 1+\alpha \\ -1, & \text{if } \dfrac{m_a(t)}{MP(t)} < 1-\alpha \\ 0, & \text{otherwise} \end{cases}$$

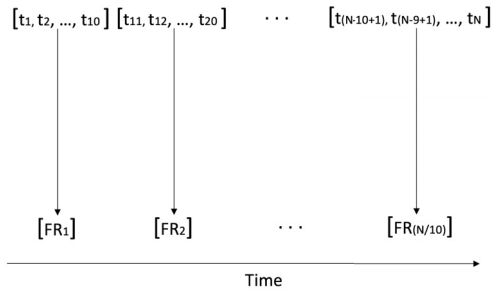
where $MP(t)$ is the mid-price at time t, $m_a(t) = \frac{1}{r}\sum_{i=1}^{r} MP(t+1)$ is the average of the future mid-price events with window size $r = 10$, and α determines the significance of the mid-price movement which is equal to 2×10^{-5}.

D. DATA NORMALIZATION AND FILTERING

The next step of the pre-processing step is data normalization. We perform a filtering and a normalization method during the feature extraction process and training. The first one is a statistical filtering method , while the second one is based on MinMax. More specifically, we perform the filtering methodology first and apply it directly on the raw MB data. The main idea of the methodology is to identify and eliminate any observation that does not reflect market activity. In the financial econometrics literature this is often referred to as data cleaning and its importance has been widely discussed



(a) **Protocol I**: Feature extraction in an online manner with zero lag delay



(b) **Protocol II**: Feature extraction with 10 events lag

FIGURE 8. Feature extraction based on the two protocols for the task of mid-price movement prediction. For given time series $(t_1, t_2, \ldots, t_N)$ there $N - 10 + 1$ feature representations (FR) for Protocol I and $\frac{N}{10}$ FR for Protocol II.

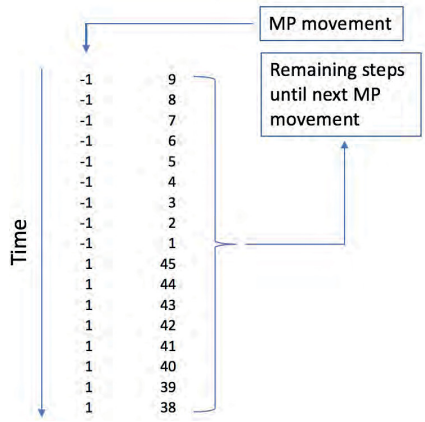


FIGURE 9. Labeling sample for the dual prediction problem of our classification and regression objective. The left part represents the direction (i.e., up or down) of the mid-price (MP) movement while the right part represents the remaining steps until the next change in MP will take place.

in the literature (e.g., [51], [52], and [53]).[7] In more detail, to filter the raw data for outliers we follow a two-step procedure. We initially remove all transactions recorded outside official trading time and clearly misrecorded transactions.[8] We then proceed by implementing a more elaborate filtering algorithm, which takes into account the statistical properties of the series to assess the validity of each observation according to its likelihood of being an outlier.[9] More specifically, for a k size window, we identify a set of (centered) neighboring observations for each data point. To avoid including prices too distant in time, the window size k should be chosen according to the trading intensity of the series. We then compute the trimmed mean of the neighboring set and mark as an outlier the considered observation if it falls more than $\alpha+\gamma$ standard deviations away from the neighbors' mean. Where γ is a granularity parameter, which should be chosen as a multiple of the tick size. The idea behind γ is to create a lower positive bound for the price variation. This is particularly important for the cleaning procedure as it is not uncommon to observe a sequence of equal mid prices in the LOB, which would lead to a zero variance and a consequent rejection of every price different from the mean value. Technically, be X_i the i^{th} element of a time series of observations X, we check:

$$(|X_i - \bar{X}_i(k)| < \alpha * s_i(k) + \gamma) \tag{21}$$

where $s_i(k)$ and $\bar{X}_i(k)$ are respectively the sample standard deviation and the trimmed mean computed over a neighborhood of k observations around X_i. Hence, we identify and remove observation X_i if (21) is true and keep it otherwise. The normalization procedure is based on MinMax for the handcrafted features, as follows:

$$MM = \frac{X_{(i)} - X_{min}}{X_{max} - X_{min}}, i \in \mathbb{R}^N, \tag{22}$$

where N is the total sample size for every feature vector X and $X_{(i)}$ is the i^{th} element of X.

VII. RESULTS & DISCUSSION

In this section, we provide results of the experiments we conducted, based on two massive LOB datasets from the US (i.e., two stocks: Amazon and Google) and Nordic (i.e., five stocks: Kesko Oyj, Outokumpu Oyj, Sampo Oyj, Rautaruukki, Wartsila Oyj) stock markets. We also discuss the perfromance of the handcrafted feature extraction universe for mid-price movement prediction and test its efficacy against a fully automated process. What is more, we make a head-to-head comparison of the three handcrafted feature sets, namely: i) "Limit Order Book (LOB):L", based on the works of [12] and [13], ii) "Tech-Quant:T-Q", based on [11], and iii) "Econ:E", which uses econometric features. Finally,

[7]While the advancement of technology has drastically reduced the number of outliers and misrecorded observations, their effect on the statistical analysis is still significant and the implementation of a cleaning procedure is, to this day, required to avoid biased results.

[8]These can be, for example, observations with a price equal to zero.

[9]The methodology follows closely [52]

(a) This is Protocol I, where we test the four sets of features (i.e., Econ, Tech-Quant, LOB, and fully automated), via nine deep learning models (i.e., five MLPs, two CNNs, and two LSTMs) for mid-price prediction. The mid-price prediction in this protocol is a combined prediction of when the next mid-price movement will happen and in which direction. This protocol is based on online learning architecture. We test this protocol for both US and Nordic stocks.

(b) This is Protocol II, where we test the four sets of features (i.e., Econ, Tech-Quant, LOB, and fully automated), via nine deep learning models (i.e., five MLPs, two CNNs, and two LSTMs) for mid-price prediction. The mid-price prediction in this protocol is a three-class problem with states for up, down, and stationary mid-price movement. Protocol I predicts every 10th event from the current mid-price state. We test this protocol for both US and Nordic stocks.

FIGURE 10. Plots (a) and (b) show the process for predicting the mid-price movement based on Protocol I and Protocol II, respectively. In both protocols, the first step is the choice of dataset. The ML trader has to choose the US or Nordic stock(s) (e.g., there is the option of choosing a stock or the 'Joint' case where all the stocks from the US or Nordic markets used for training). The second step is to choose the feature set. The ML trader has to choose one of the four suggested feature sets, which are: The newly introduced econometric set, the one that is based on technical and quantitative indicators, another one based on time-sensitive and time-insensitive LOB features, and the last one based on fully automated features. The third step is whether the prediction should be based on a balanced or unbalanced set. The fourth step is the choice of one of the suggested nine deep learning models. The final step is the one that differs in Protocol I and Protocol II. The difference lies in the fact that Protocol I is a combined classification and regression optimization problem with zero event lag and Protocol II is a three-class classification problem based on a 10-event lag.

we compare these three sets of handcrafted features with features extracted based on an LSTM autoencoder.

Latent representations are extracted after training an LSTM AE. This training employs an extensive grid search, in which the best perfromance is reported. The grid search is based on symmetrical, assymetrical, shallow, deep, overcomplete, and undercomplete LSTM AE. The provided options vary from: i) the encoder with maximum depth up to four hidden LSTM layers with different numbers of filters varying according to the list {128, 64, 18, 9}, ii) the decoder with maximum depth up to four hidden LSTM layers with different numbers of filters varying according to the list {128, 64, 18, 9}, and iii) the latent representation with different options varying according to the list {5, 10, 20, 50, and 130}. The best performance reported is based on a symmetrical and undercomplete LSTM AE of four hidden LSTM layers with 128, 64, 18, and 9 filters respectively, and 10 for the latent representation vector size. The list of the suggested grid search is limited; however, we believe it provides a wide range of combinations in order to make a fair comparison of a fully automated feature extraction process against advanced

TABLE 5. Protocol I: f1 scores for the US stocks. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		Econ		Tech-Quant		LOB	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Amazon	0.32	0.39	0.52	0.33	0.44	0.31
	Google	0.38	0.32	0.48	0.34	0.52	0.32
	Joint	0.32	0.32	0.44	0.33	0.54	0.31
MLP_2	Amazon	0.32	0.49	0.38	0.50	0.36	0.30
	Google	0.36	0.43	0.47	**0.57**	0.34	0.31
	Joint	0.33	0.52	0.38	0.33	0.35	0.32
MLP_3	Amazon	0.33	0.50	0.54	0.48	0.49	0.29
	Google	0.40	0.51	0.52	0.51	0.38	0.30
	Joint	0.32	0.52	0.40	**0.56**	0.51	0.31
MLP_4	Amazon	0.32	0.53	0.38	0.33	0.36	0.28
	Google	0.51	0.32	0.44	0.44	0.32	0.30
	Joint	0.52	0.52	0.37	0.34	0.35	0.28
MLP_5	Amazon	0.30	0.42	0.33	0.33	0.31	0.31
	Google	0.43	0.45	0.34	0.44	0.48	0.31
	Joint	0.52	0.33	0.37	0.34	0.35	0.30
CNN_1	Amazon	0.50	0.49	0.53	0.40	0.43	0.51
	Google	0.49	0.49	0.57	0.48	0.48	0.51
	Joint	0.49	0.48	0.54	0.41	0.53	0.45
CNN_2	Amazon	0.51	0.45	0.55	0.52	0.56	0.51
	Google	0.45	0.51	0.55	0.49	0.50	0.47
	Joint	0.53	0.50	0.57	0.43	0.56	0.46
LSTM_1	Amazon	0.52	0.46	0.51	0.50	0.44	0.44
	Google	0.46	0.49	**0.58**	0.49	0.42	0.51
	Joint	0.52	0.48	0.56	0.49	0.50	0.49
LSTM_2	Amazon	0.45	0.49	0.56	0.54	0.52	0.47
	Google	0.52	0.51	0.54	0.53	0.45	0.51
	Joint	0.40	0.51	**0.59**	0.55	0.55	0.51

TABLE 6. Protocol I: RMSE scores for the US stocks. *Note: Highlighted text shows the best RMSE performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		Econ		Tech-Quant		LOB	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Amazon	28.99	**33.02**	80.64	79.63	86.22	79.61
	Google	131.18	37.53	96.75	98.25	94.59	96.11
	Joint	32.32	38.43	88.88	87.70	87.49	87.72
MLP_2	Amazon	**28.39**	224.91	80.67	381.22	90.65	603.53
	Google	45.57	282.40	101.91	404.47	96.51	550.23
	Joint	33.34	266.32	92.79	628.33	87.86	608.31
MLP_3	Amazon	30.44	227.90	81.37	393.43	793.08	615.51
	Google	61.28	312.42	101.74	498.35	95.90	563.60
	Joint	33.31	255.48	88.30	474.82	87.78	671.49
MLP_4	Amazon	28.61	253.89	79.60	634.33	79.60	634.31
	Google	95.73	296.81	101.54	688.81	96.43	167.81
	Joint	**32.29**	269.86	87.71	671.49	**87.70**	671.67
MLP_5	Amazon	28.40	264.38	79.78	628.99	82.89	629.01
	Google	42.07	271.69	100.01	664.08	96.47	654.19
	Joint	33.22	496.80	87.73	663.93	87.70	663.89
CNN_1	Amazon	31.21	337.89	90.35	592.95	104.90	686.93
	Google	190.19	201.45	203.65	319.34	99.39	421.14
	Joint	34.85	367.70	289.76	278.90	260.46	836.35
CNN_2	Amazon	30.47	342.41	380.90	671.19	144.22	767.75
	Google	189.87	178.98	167.89	302.33	97.63	753.54
	Joint	37.46	362.68	200.67	165.23	352.20	737.46
LSTM_1	Amazon	29.08	277.94	110.86	454.85	84.64	774.62
	Google	137.65	207.48	123.36	599.94	99.13	418.62
	Joint	36.05	240.14	92.58	604.54	96.58	421.04
LSTM_2	Amazon	32.03	235.14	86.81	440.10	82.03	500.47
	Google	38.37	262.28	98.38	398.17	97.45	449.66
	Joint	36.65	309.80	89.69	487.73	86.95	527.83

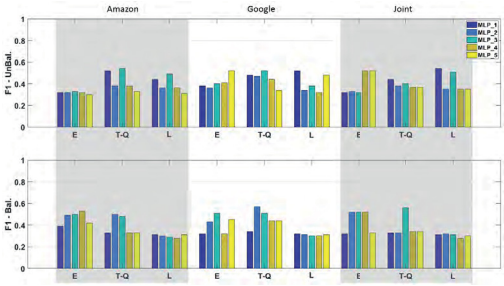
handcrafted features. We should also mention that, despite the extensive grid search on the LSTM AE, we limited our search to up to four hidden units for the encoding and decoding parts with four different filter options. Further analysis on the topic is required.

In order to scrutinize the efficacy of the handcrafted and fully automated features, we use two experimental protocols and nine deep learning models, and present results based on unbalanced and balanced inputs. In particular, we test the four feature sets according to two protocols: the newly introduced experimental protocol (i.e., Protocol I) for online learning, as we explain in Section VI, and Protocol II, that follows [17]. Protocol I is suitable for online learning, whose main objective is to predict when a change in the mid-price will happen (i.e., regression problem) and in which direction, for instance, up or down (i.e., two-class classification problem). Protocol II predicts the mid-price movement direction for every next 10^{th} event, where feature representations are based on independent 10-event blocks. Authors in [17] used a joint training set of the five Nordic stocks for seven trading days and the next three days as testing for mid-price movement prediction (i.e., up, down, and stationary movement). We incorporate the same idea here, under the name "Joint", and we also use the same 7-3 training and testing proportion for each stock individually for both US and Nordic datasets. A general idea for both protocols can be seen in Fig. 10.

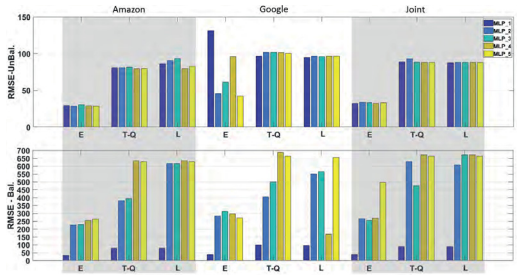
Protocol I and Protocol II use three types of deep neural networks as classifiers and regressors. In particular, we utilize five different MLPs, two CNNs, and two LSTMs. Motivation for choosing MLPs is the fact that such a simple neural network can perform extremely well when descriptive handcrafted features are used as input. The next type of neural network that we use is CNN. The first CNN, named "CNN_1" is based on [43], whereas the second one, named "CNN_2" is based on the grid search that we describe below. The last type of neural network that we utilize is LSTM. We use two different architectures: the first one, named "LSTM_1", is based on [17], and the second one, named "LSTM_2" is based on LSTM with attention mechanism. In total, we train independently nine deep neural networks for each of the two experimental protocols separately. Details of these nine topologies can be found in Table 4.

We report results for nine different neural networks, two of which are based on existing works as shown above. For the remaining seven neural networks we conduct the following grid search:

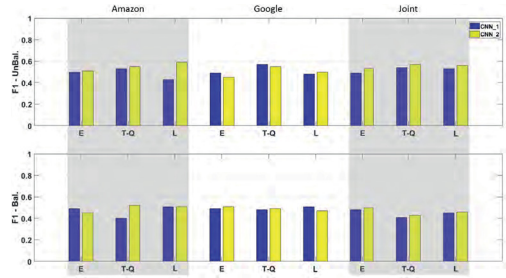
- For MLPs we set a limit up to three hidden layers, where for the number of nodes we set the options { 4, 9, 18, 64, 128, 256, and 512} nodes per layer and for dropout 20% and 50%. We report results based on five MLPs since these neural networks achieved good results for several cases (see Section VII-A for results discussion).
- For CNN we conduct an extensive grid search limited to up to three convolutional layers (with the option of 1-dimensional and 2-dimensional convolutional layer



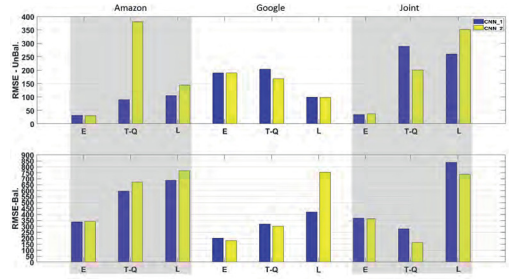
(a) F1 scores based on the unbalanced (top) and balanced (bottom) sets



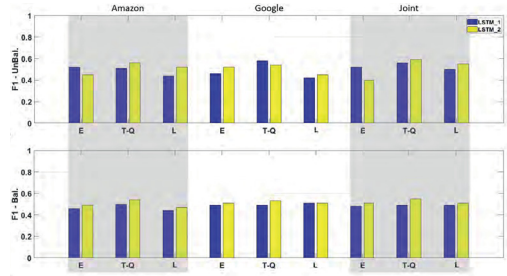
(b) RMSE scores based on the unbalanced (top) and balanced (bottom) sets



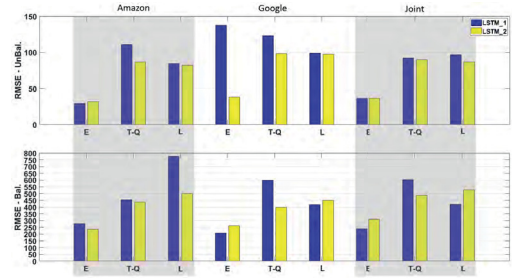
(c) F1 scores based on the unbalanced (top) and balanced (bottom) sets



(d) RMSE scores based on the unbalanced (top) and balanced (bottom) sets



(e) F1 scores based on the unbalanced (top) and balanced (bottom) sets



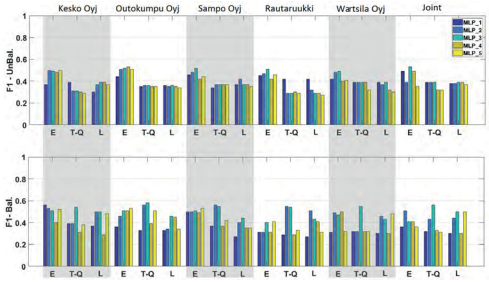
(f) RMSE scores based on the unbalanced (top) and balanced (bottom) sets

FIGURE 11. F1 (left column plots) and RMSE (right column plots) scores for the nine deep learning models based on the US data.

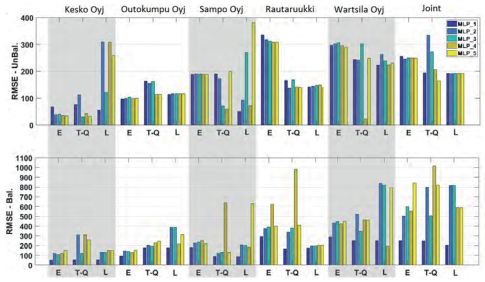
types) with 8, 16, and 32 filters and kernels size options {4×10, 4×20, 4×30, and 4×40 for the 2-dimensional case and 3 and 4 for the 1-dimensional case}. Dropout options are restricted to 20% and 50%. We report only one CNN since we noticed that shallower CNN architectures had very poor performance and no significant difference for the deeper ones.

- For LSTM we follow the same approach with up to three hidden layers and five options for hidden LSTM units {9, 18, 32, 64, 128} and the option of attention layer. We report only one LSTM performance since all other topologies performed worse for our task.

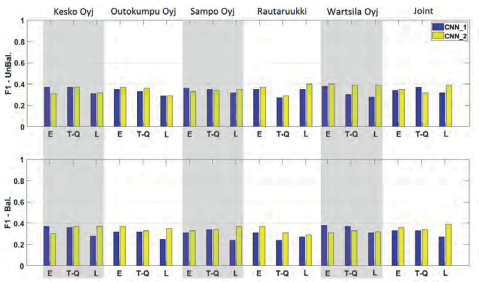
The training of these nine neural networks takes place at CSC super-cluster where we use Pascal P100 and K80 GPUs. We use multi-GPUs, under Keras (i.e., [54]) framework, in order to reduce the training time. The models, apart from CNN_1 and LSTM_1, use the Nesterov-Adam optimizer with a learning rate of 0.002, with mean squared error and binary cross-entropy for the dual output of Protocol I where this dual output is weighted by 0.01 and 0.99, respectively, and categorical cross-entropy as loss function for Protocol II. Additionally, we use 250 epochs to train our models with data shuffling and validation ratio of 0.2. Finally, in order to control overfitting we utilize Dropout to the majority of the suggested neural networks. By dropping out some nodes (i.e., a dropped out node have a zero output) from neural network topologies we control node dependencies and we achieve more robust results.



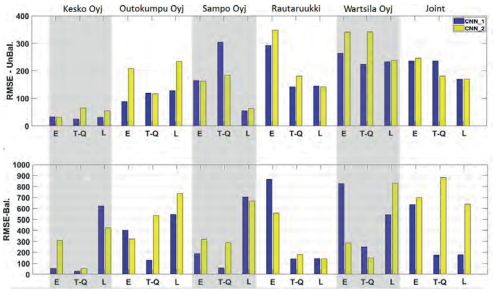
(a) F1 scores based on the unbalanced (top) and balanced (bottom) sets



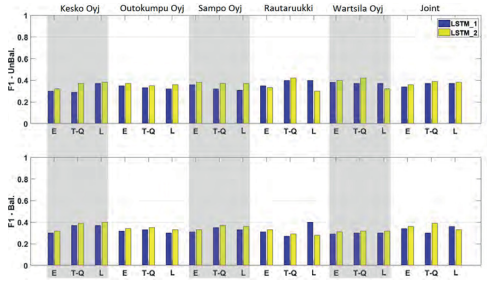
(b) RMSE scores based on the unbalanced (top) and balanced (bottom) sets



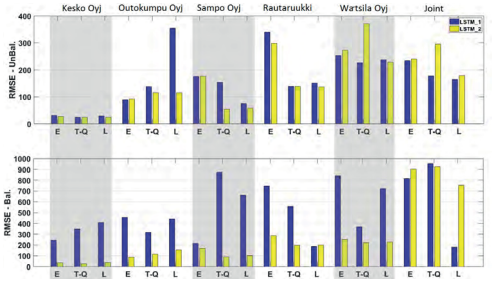
(c) F1 scores based on the unbalanced (top) and balanced (bottom) sets



(d) RMSE scores based on the unbalanced (top) and balanced (bottom) sets



(e) F1 scores based on the unbalanced (top) and balanced (bottom) sets



(f) RMSE scores based on the unbalanced (top) and balanced (bottom) sets

FIGURE 12. F1 (left column plots) and RMSE (right column plots) scores for the nine deep learning models based on the Nordic data.

A. RESULTS

We present our results in separate tables for Protocol I (see Table 5 - Table 10) and Protocol II (see Appendix VIII-B). For each protocol, we split the results (i.e., f1 score and RMSE for Protocol I and f1 scores for Protocol II) for both US and Nordic datasets. We would like to mention that results derived from the LSTM AE, for both f1 and RMSE scores, are presented in separate tables (see Tables 9 & 10 for Protocol I and Tables A.2 & A.4 for Protocol II). Since handcrafted feature results overperformed the fully automated feature set we emphasize more on their perfromance by providing tables together with bar plots (see Fig. 11 & 12). Each of the tables contains the full head-to-head comparison for the three handcrafted features sets for each of the nine different deep learning models separately. For instance, Table 7 contains f1 scores for the Nordic stocks based on Protocol I. The table has five main columns (i.e., Model, Stock, Econ, Tech-Quant, and LOB) and six subcolumns divided into three pairs (i.e., UnBal. and Bal.). The first main column contains the nine deep neural networks; the second main column contains the five independent and different Nordic stocks, in which the sixth row for every model is the joint training set based on these five stocks; and the third, fourth and fifth main columns represent the three handcrafted feature sets. Moreover, for every feature set, we present results for unbalanced and balanced cases, whereas for the balanced cases we use random undersampling for the majority class. Even though balanced datasets do not project a realistic

TABLE 7. Protocol I: f1 scores for the Nordic stocks. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

Model	Stock	Econ		Tech-Quant		LOB	
		UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Kesko Oyj	0.37	0.56	0.39	0.39	0.39	0.37
	Outokumpu Oyj	0.44	0.36	0.35	0.33	0.36	0.33
	Sampo Oyj	0.46	0.50	0.34	0.37	0.37	0.35
	Rautaruukki	0.45	0.31	0.42	0.29	0.42	0.27
	Wartsila Oyj	0.42	0.31	0.39	0.32	0.39	0.30
	Joint	0.49	0.36	0.39	0.32	0.38	0.30
MLP_2	Kesko Oyj	0.50	0.53	0.31	0.39	0.37	0.50
	Outokumpu Oyj	0.51	0.46	0.36	0.56	0.35	0.34
	Sampo Oyj	0.48	0.50	0.37	0.56	0.42	0.40
	Rautaruukki	0.47	0.31	0.29	0.55	0.32	0.51
	Wartsila Oyj	0.48	0.49	0.39	0.32	0.37	0.46
	Joint	0.39	0.51	0.39	0.43	0.38	0.44
MLP_3	Kesko Oyj	0.49	0.51	0.31	0.54	0.39	0.50
	Outokumpu Oyj	0.52	0.51	0.36	**0.58**	0.36	0.46
	Sampo Oyj	0.52	0.51	0.37	0.55	0.37	0.44
	Rautaruukki	0.51	0.40	0.29	0.54	0.29	0.43
	Wartsila Oyj	0.49	0.47	0.39	0.55	0.39	0.43
	Joint	**0.53**	0.41	0.39	**0.56**	0.39	0.50
MLP_4	Kesko Oyj	0.48	0.40	0.30	0.31	0.39	0.29
	Outokumpu Oyj	**0.53**	0.51	0.35	0.39	0.35	0.45
	Sampo Oyj	0.42	0.49	0.37	0.37	0.37	0.35
	Rautaruukki	0.42	0.31	0.30	0.29	0.29	0.41
	Wartsila Oyj	0.40	0.50	0.39	0.32	0.32	0.30
	Joint	0.49	0.41	0.32	0.33	0.39	0.30
MLP_5	Kesko Oyj	0.50	0.52	0.29	0.38	0.37	0.48
	Outokumpu Oyj	0.51	0.53	0.35	0.51	0.34	0.34
	Sampo Oyj	0.44	0.53	0.37	0.42	0.35	0.35
	Rautaruukki	0.46	0.41	0.29	0.33	0.27	0.31
	Wartsila Oyj	0.41	0.32	0.32	0.32	0.30	0.48
	Joint	0.35	0.36	0.32	0.31	0.37	0.50
CNN_1	Kesko Oyj	0.37	0.37	0.37	0.36	0.31	0.28
	Outokumpu Oyj	0.35	0.32	0.33	0.32	0.29	0.25
	Sampo Oyj	0.36	0.31	0.35	0.34	0.32	0.24
	Rautaruukki	0.35	0.31	0.27	0.24	0.35	0.27
	Wartsila Oyj	0.38	0.38	0.30	0.37	0.28	0.31
	Joint	0.34	0.33	0.37	0.33	0.32	0.27
CNN_2	Kesko Oyj	0.31	0.32	0.37	0.31	0.32	0.39
	Outokumpu Oyj	0.37	0.37	0.36	0.33	0.29	0.35
	Sampo Oyj	0.33	0.33	0.34	0.34	0.35	0.37
	Rautaruukki	0.37	0.37	0.29	0.31	0.40	0.29
	Wartsila Oyj	0.40	0.31	0.39	0.33	0.39	0.32
	Joint	0.35	0.36	0.32	0.34	0.39	0.39
LSTM_1	Kesko Oyj	0.30	0.30	0.29	0.37	0.37	0.37
	Outokumpu Oyj	0.35	0.32	0.33	0.33	0.32	0.30
	Sampo Oyj	0.36	0.31	0.32	0.35	0.31	0.33
	Rautaruukki	0.35	0.31	0.40	0.27	0.40	0.40
	Wartsila Oyj	0.38	0.29	0.37	0.30	0.37	0.30
	Joint	0.34	0.34	0.37	0.30	0.37	0.36
LSTM_2	Kesko Oyj	0.32	0.32	0.37	0.39	0.38	0.40
	Outokumpu Oyj	0.37	0.34	0.35	0.35	0.36	0.33
	Sampo Oyj	0.38	0.33	0.37	0.37	0.37	0.36
	Rautaruukki	0.33	0.33	0.42	0.29	0.30	0.28
	Wartsila Oyj	0.40	0.31	0.42	0.32	0.32	0.32
	Joint	0.36	0.36	0.39	0.39	0.38	0.33

TABLE 8. Protocol I: RMSE scores based on Nordic stocks for the handcrafted features. *Note: Highlighted text shows the best RMSE performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

Model	Stock	Econ		Tech-Quant		LOB	
		UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Kesko Oyj	68.62	50.97	76.99	55.23	173.92	54.94
	Outokumpu Oyj	97.76	91.29	164.00	176.92	114.58	176.93
	Sampo Oyj	187.98	178.04	190.84	87.32	51.71	86.69
	Rautaruukki	334.61	292.36	166.61	167.00	142.40	173.41
	Wartsila Oyj	297.42	289.06	244.18	250.69	223.17	250.74
	Joint	255.92	252.13	195.15	250.02	192.19	205.41
MLP_2	Kesko Oyj	38.65	119.89	113.20	309.49	539.66	128.88
	Outokumpu Oyj	99.89	143.86	155.78	203.97	116.83	388.18
	Sampo Oyj	190.41	226.04	172.69	119.33	93.07	206.68
	Rautaruukki	317.66	374.41	137.48	340.20	144.91	196.54
	Wartsila Oyj	302.42	433.15	241.99	522.49	262.87	839.49
	Joint	246.80	502.29	334.22	799.74	191.54	815.36
MLP_3	Kesko Oyj	40.20	108.74	31.13	122.02	447.78	128.45
	Outokumpu Oyj	103.84	136.95	163.30	194.48	117.14	388.12
	Sampo Oyj	190.81	236.69	71.01	128.70	269.54	201.89
	Rautaruukki	312.41	390.14	168.69	378.60	149.69	196.54
	Wartsila Oyj	306.61	446.21	302.83	349.45	224.24	820.50
	Joint	249.72	599.45	272.35	504.21	192.46	815.36
MLP_4	Kesko Oyj	102.52	121.94	43.46	308.61	105.17	148.69
	Outokumpu Oyj	98.44	126.24	114.98	229.33	116.83	216.99
	Sampo Oyj	189.59	254.34	59.40	639.52	72.42	185.65
	Rautaruukki	309.03	625.19	142.40	985.50	149.69	204.93
	Wartsila Oyj	295.85	423.50	22.84	464.79	224.24	716.44
	Joint	250.11	555.76	206.55	1015.87	192.46	589.94
MLP_5	Kesko Oyj	99.79	148.71	33.19	259.19	650.56	148.43
	Outokumpu Oyj	99.79	151.41	114.83	246.73	117.14	312.73
	Sampo Oyj	189.36	220.94	199.50	129.55	379.95	629.82
	Rautaruukki	308.72	399.87	140.08	409.47	138.74	205.36
	Wartsila Oyj	290.03	450.53	248.25	460.97	230.90	793.99
	Joint	248.34	841.92	**165.29**	818.41	193.10	589.15
CNN_1	Kesko Oyj	32.73	55.43	25.32	30.37	30.98	623.58
	Outokumpu Oyj	88.04	401.17	118.89	128.44	128.44	545.72
	Sampo Oyj	164.81	189.73	304.42	59.40	54.07	704.06
	Rautaruukki	292.60	865.16	141.77	150.66	145.31	261.21
	Wartsila Oyj	262.92	827.61	224.24	248.79	232.48	544.67
	Joint	235.99	636.80	236.17	**176.25**	169.12	178.67
CNN_2	Kesko Oyj	30.75	310.08	64.88	54.35	54.85	424.82
	Outokumpu Oyj	208.71	321.86	116.56	535.32	233.86	737.63
	Sampo Oyj	163.52	317.88	184.45	288.18	62.67	669.31
	Rautaruukki	347.81	556.37	181.09	168.62	142.08	304.91
	Wartsila Oyj	340.68	284.84	340.92	149.73	238.14	832.23
	Joint	245.35	700.03	180.72	880.98	169.03	639.19
LSTM_1	Kesko Oyj	32.50	245.49	25.24	346.99	30.37	409.85
	Outokumpu Oyj	89.83	455.90	138.15	315.56	354.52	440.01
	Sampo Oyj	176.26	216.65	154.00	875.29	75.96	662.33
	Rautaruukki	339.78	746.22	138.94	556.63	150.66	187.29
	Wartsila Oyj	253.11	839.99	226.97	369.42	237.22	719.71
	Joint	234.65	816.23	178.73	952.69	166.10	180.74
LSTM_2	Kesko Oyj	28.22	34.63	**24.81**	**24.66**	24.88	39.29
	Outokumpu Oyj	92.34	89.71	116.09	114.20	115.78	153.77
	Sampo Oyj	177.53	171.03	54.07	90.62	59.23	100.59
	Rautaruukki	297.96	285.85	139.15	198.07	136.83	200.85
	Wartsila Oyj	273.83	253.96	370.96	220.45	229.40	227.30
	Joint	240.97	903.34	295.32	925.90	179.12	752.62

trading scenario (i.e., trading fees are not applicable), it is important to give an equal opportunity to the minority class, which can be an ML trader's trading position. More specifically, for Protocol I and the classification task, the Nordic dataset has 45% for the downward movement and 55% for the upward, while for the US dataset is 47% for the downward movement and 53% for the upward. The undersampling offers an 85% data reduction for the Nordic set and 90% for the US set. For better interpretation of Protocol I we provide bar plots which show the reaction of every deep learning model and dataset for the unbalanced and balanced cases (see Fig. 11 and Fig. 12). Protocol II and the Nordic dataset exhibits a 75% for the stationary condition, with the remaining 25% being equally divided to the upward and downward mid-price movement before undersampling. For the US dataset 73% belongs to the stationary condition, 20% to the upward movement and the remaining 7% to the downward movement. The undersampling offers a 30% data reduction for the Nordic dataset and 10% data reduction for the US dataset.

B. DISCUSSION

The conducted experiments reveal some interesting results for both experimental protocols and datasets selection. Both protocols forecast the mid-price movement, with Protocol I forecasting the mid-price movement every next event and Protocol I with a lag of 10 events. Protocol I provides

TABLE 9. Protocol I: f1 and RMSE scores based on US stocks for the fully-automated features. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		LSTM AE - f1		LSTM AE - RMSE	
Model	Stock	UnBal.	Bal.	UnBal.	Bal
MLP_1	Amazon	0.37	0.36	79.81	90.79
	Google	0.34	0.34	109.80	110.98
	Joint	0.31	0.35	121.49	100.78
MLP_2	Amazon	0.31	0.36	88.37	116.29
	Google	0.34	0.34	96.51	136.51
	Joint	0.35	0.35	101.44	125.44
MLP_3	Amazon	0.31	0.36	83.73	116.89
	Google	0.33	0.34	96.05	134.96
	Joint	0.31	0.35	121.54	126.52
MLP_4	Amazon	0.36	0.36	**79.50**	117.54
	Google	0.32	0.34	103.76	134.17
	Joint	0.31	0.35	133.24	126.10
MLP_5	Amazon	0.31	**0.37**	81.86	105.38
	Google	0.32	0.34	98.29	124.58
	Joint	0.35	0.35	88.80	114.09
CNN_1	Amazon	0.36	0.36	187.06	90.92
	Google	0.34	0.34	95.92	109.60
	Joint	0.35	0.35	363.45	110.31
CNN_2	Amazon	0.36	0.31	82.66	90.18
	Google	0.34	0.32	113.39	109.24
	Joint	0.36	0.35	216.62	101.36
LSTM_1	Amazon	0.31	0.31	79.89	**87.70**
	Google	0.34	0.32	98.02	109.24
	Joint	0.31	0.35	87.70	100.52
LSTM_2	Amazon	0.46	0.31	80.03	90.17
	Google	**0.53**	0.32	97.13	109.06
	Joint	0.31	0.35	90.17	87.75

more information regarding the high-frequency activity since it takes into consideration every trading event. We cannot directly compare the two protocols since both tackle the problem of mid-price forecasting from a different angle. We also observe that in general, larger data samples increase deep learning models' performance. However, by focusing on each protocol seperately, we can see that: for Protocol I, the best classification score comes from US dataset and best regression score from Nordic dataset, while, for Protocol II, the best classification score comes again from the US dataset.

Each one of the nine neural networks has to perform a dual task, regression and classification simultaneously. To begin with, the Joint (i.e., the full range of stocks is used for training) reports for the Nordic dataset the best f1 performance that comes from MLP_3, for both unbalanced and balanced datasets under the Econ feature set with 53% and 56% for the Tech-Quant set. This MLP did not perform well for the regression task where the RMSE was above 165.29. For the stock specific case: we achieve the best classification performance of 53% f1 score for Outokumpu Oyj under MLP_4 and the Econ feature set with RMSE of 98.44. Thisvstock-specific performance of the MLP_4 is the best trade-off between classification and regression for the Nordic dataset. If we want to focus on the regression task only, we can choose the more advanced model, LSTM_2, with RMSE of approximately 24 for both unbalanced and balanced Tech-Quant feature sets for Kesko Oyj.

TABLE 10. Protocol I: f1 and RMSE scores based on Nordic stocks for the fully-automated features. *Note: Highlighted text shows the best RMSE performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		LSTM AE - f1		LSTM AE - RMSE	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Kesko Oyj	0.29	0.37	39.11	48.90
	Outokumpu Oyj	0.33	0.33	186.68	197.53
	Sampo Oyj	0.35	0.35	63.78	79.56
	Rautaruukki	0.27	0.27	182.43	201.86
	Wartsila Oyj	0.37	0.34	282.11	309.47
	Joint	0.37	0.37	207.90	190.24
MLP_2	Kesko Oyj	0.37	0.29	51.70	47.35
	Outokumpu Oyj	0.34	0.34	202.91	199.90
	Sampo Oyj	0.35	0.32	87.96	146.54
	Rautaruukki	0.27	0.40	210.69	212.94
	Wartsila Oyj	0.43	0.36	299.77	395.76
	Joint	0.30	0.37	236.10	207.82
MLP_3	Kesko Oyj	0.29	0.37	47.12	52.91
	Outokumpu Oyj	0.34	0.32	204.05	183.82
	Sampo Oyj	0.32	0.32	92.06	67.94
	Rautaruukki	0.40	0.40	207.93	174.24
	Wartsila Oyj	0.37	0.36	303.98	396.01
	Joint	0.30	0.37	232.29	209.93
MLP_4	Kesko Oyj	0.29	0.37	55.23	52.55
	Outokumpu Oyj	0.34	0.32	202.99	200.38
	Sampo Oyj	0.35	0.33	90.66	58.94
	Rautaruukki	0.40	0.40	209.65	202.04
	Wartsila Oyj	0.37	0.36	299.49	327.36
	Joint	0.30	0.37	233.67	204.09
MLP_5	Kesko Oyj	0.29	0.37	45.40	52.50
	Outokumpu Oyj	0.33	0.32	195.46	202.28
	Sampo Oyj	0.35	0.33	79.83	111.58
	Rautaruukki	0.40	0.40	196.63	245.04
	Wartsila Oyj	0.37	0.37	292.99	344.89
	Joint	0.30	0.37	222.40	202.40
CNN_1	Kesko Oyj	0.33	0.38	44.30	288.28
	Outokumpu Oyj	0.33	0.37	186.07	299.16
	Sampo Oyj	0.35	0.32	61.76	655.34
	Rautaruukki	0.40	0.27	173.71	510.65
	Wartsila Oyj	0.37	0.30	279.80	300.04
	Joint	0.40	0.37	206.89	305.67
CNN_2	Kesko Oyj	0.37	0.37	40.20	70.80
	Outokumpu Oyj	0.33	0.37	185.48	181.14
	Sampo Oyj	0.35	0.35	62.34	429.33
	Rautaruukki	0.27	0.27	176.98	350.67
	Wartsila Oyj	0.43	0.30	280.23	432.86
	Joint	**0.49**	0.37	**204.61**	350.43
LSTM_1	Kesko Oyj	0.37	0.38	39.10	42.10
	Outokumpu Oyj	0.35	0.33	185.28	179.89
	Sampo Oyj	0.35	0.35	61.65	67.34
	Rautaruukki	0.35	0.43	181.27	223.70
	Wartsila Oyj	0.35	0.30	279.46	256.78
	Joint	0.37	0.37	234.78	110.54
LSTM_2	Kesko Oyj	0.42	0.37	**39.03**	40.19
	Outokumpu Oyj	**0.47**	0.45	185.24	178.67
	Sampo Oyj	0.36	0.35	61.69	69.42
	Rautaruukki	0.35	0.30	180.49	167.89
	Wartsila Oyj	0.42	0.30	279.13	243.32
	Joint	0.38	0.37	212.89	89.90

For the US dataset, the new protocol presents more interesting results. For the Joint case, where both Amazon and Google used for training, the LSTM_2 achieves 59% f1 score and RMSE of 89.69, whereas, for the stock specific case, LSTM_1 under the Tech-Quant feature set achieves 58% f1 score and high RMSE of 123.36 for Google and the unbalanced case. If we focus only on the regression part, we can choose the entire MLP universe and the Econ feature set for Amazon and the Joint case. The newly introduced Econ feature set performed very well for the regression task also

for LSTM_2 across the entire protocol for the unbalanced dataset. One more interesting observation is that the Econ feature set together with the shallower MLP_1 and the balanced set reports very low RMSE for Amazon, Google, and the Joint cases, respectively. That means that the Econ feature set, for the Amazon and Joint case, were able to predict that the mid-price will change its direction in a millisecond duration. Here, it is vital to report that the daily trading activity, for the US and Nordic stocks, contains several trades with the same timestamp/millisecond. Approximately 30% of the trades, in the US dataset, occur in a millisecond, whereas this percentage for the Nordic dataset is 36%.

For Protocol II and the Joint case we achieve the best forecasting performance of 51% f1 for the Nordic dataset based on MLP_4 (which is one of our deeper MLP architectures) under the Tech-Quant feature set and the unbalanced case. For the Joint case in the US dataset, we achieve the best f1 performance of 65% based on MLP_4 under the Tech-Quant feature set and the balanced case. In terms of individual stock performance for the Nordic case we achieve 63% f1 score for Kesko Oyj, and our shallower MLP (i.e., MLP_1) under the Tech-Quant set, while for the US dataset we achieve an f1 performance of 65% for Google based on MLP_4 for the balanced case. We can see that MLPs for Protocol II were able to retain the information that the Tech-Quant feature set carries. The majority of the Tech-Quant features was derived from technical analysis, a type of analysis which is based on geometrical pattern identification of agglutinated times series like ours. What is more, the data size affected the performance of models and feature sets. For instance, Kesko Oyj, which scored the highest f1 score, is the stock with the least daily trading activity compared to the rest of the Nordic stocks and of course compared to the massive US dataset. Finally, we would like to point out that we limited the experiments to two US and five Nordic stocks; we leave the extension of the present evaluation on wider LOB datasets for future research that will help us to identify similarities among stock categories and time periods.

VIII. CONCLUSION

In this paper, we extracted handcrafted features based on the econometric literature for mid-price prediction using deep learning techniques. Our work is the first of its kind since we do not only utilize an extensive feature set list, based on econometrics for the mid-price prediction task, but we also provide a fair comparison with two other existing state-of-the-art handcrafted and fully automated feature sets . Our extensive experimental setup, based on liquid and illiquid stocks (i.e., two US and five Nordic stocks) showed superiority of the suggested handcrafted feature sets against the fully automated process derived from an LSTM AE. What is more, our research sheds light on the area of deep learning and feature engineering by providing information based on online mid-price predictions. Our findings suggest that extensive analysis of the input signal leads to high forecasting performance even with simpler neural network architects like shallow MLPs, particularly when advanced features capture the relevant information edge. More specifically, econometric features and deep learning predicted that the mid-price would change direction in a millisecond duration for Amazon and the Joint (i.e., training on both Amazon and Google) cases. Although these results are promising, our study here also suggests that selection of features and models should be differentiated for liquid and illiquid stocks

APPENDIX

A. FEATURE POOL

See Figure 11–12 and Tables 7–10.

1) STATISTICAL FEATURES

- Mid price is defined as:

$$MP = \frac{Ask_{best} + Bid_{best}}{2} \tag{A.1}$$

- Financial duration is defined as:

$$FD = T_t - T_{t-1}, \tag{A.2}$$

where T denotes the time instance at time t.

- Average mid-price financial duration is defined as:

$$AMPD_l = \frac{\left\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\right\}_{i=1}^{N}}{\left\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\right\}_{i=1}^{N}}, \tag{A.3}$$

where $\left\{\mathcal{T}_k\right\}_{k=1}^{N}$, and $\left\{\mathcal{P}_k\right\}_{k=1}^{N}$ are the partial cumulative sums of time and price differences for every LOB level for N samples.

- Mid price deeper levels are equal to:

$$DMP = \frac{Ask_l + Bid_l}{2},\ l = 2:10 \tag{A.4}$$

where l denotes the depth of the LOB.

- Log returns are defined as:

$$r(X)_i = X_i - X_{i-1}; \tag{A.5}$$

where X_i is the logarithmic price

2) VOLATILITY MEASURES

The features in this category aim to estimate, either the integrated variance (IV), that is the process

$$IV_t = \int_0^t \sigma_s^2 \mathrm{d}s \tag{A.6}$$

or, more generally, the quadratic variation (QV)

$$[X, X]_t = \int_0^t \sigma_s^2 \mathrm{d}s + \sum_{0<s\leq t} (\zeta_s \mathrm{d}N_s)^2. \tag{A.7}$$

Here X is the logarithmic price of some given asset. We assume that X_t follows an Itô semimartingale; that is,

$$X_t = X_0 + \int_0^t b_s \mathrm{d}s + \int_0^t \sigma_s \mathrm{d}W_s + \int_o^t \zeta_s \mathrm{d}N_s, \tag{A.8}$$

where b is locally bounded, σ is cádlág and predictable, and W is a standard Weiner process, ζ is a thin (i.e., finite) process mapping the jump size, and N is the counting process associated to the jump times of X. We define Δ_n the time elapsed between two adjacent observations; specifically, if we assume the observations are equidistant in time we have $\Delta_n = \lfloor \frac{t}{n} \rfloor$. As we do not work in calendar time we will have $\Delta_n = \frac{1}{n}$.

- Realized variance
  The realized variance [55] is the most natural estimator of the quadratic variation process and is equal to:

$$RV_t = \sum_{i=1}^{n} (r(X)_i)^2. \tag{A.9}$$

- Realized kernel
  Realized kernels [56] are used to obtain a noise robust estimate of QV as follows:

$$RK_t = \gamma_0(X_{\Delta_n}) + \sum_{h=1}^{H} k\left(\frac{h}{H}\right)\{\gamma_h(X_{\Delta_n}) + \gamma_{-h}(X_{\Delta_n})\}, \tag{A.10}$$

  with H the kernel bandwidth, $\gamma_h(X_{\Delta_n})$ the autocovariation process, k is the kernel function of choice. In particular we use a non-flat-top Parzen and our implementation follows closely [53].
- Realized pre-averaged variance
  The pre-averaged realized variance [57] is akin to the realized kernel estimator (in fact they are asymptotically equivalent). As for the realized kernel, the pre-averaged realized variance is used to retrieve a noise-free measurement of the quadratic variation of our price process and it is calculated as follows:

$$PA - RV_t = \frac{\sqrt{\Delta_n}}{\theta\psi_2} \sum_{i=0}^{n-H+1} (\bar{X}_i^n)^2 - \frac{\psi_1 \Delta_n}{2\theta^2\psi_2} \sum_{i=0}^{N} (r(X))^2. \tag{A.11}$$

  As before we have H the kernel bandwidth and θ the pre-averaging horizon. Further, given a nonzero real-valued function $g : [0, 1] \to \mathbb{R}$ with $g(0) = g(1) = 0$ and which is further continuous and piecewise continuously differentiable such that its derivative g' is piecewise Lipschitz. Then, we define:

$$\psi_1 = \int_0^1 (g'(s))^2 \mathrm{d}s, \qquad \psi_1 = \int_0^1 (g(s))^2 \mathrm{d}s.$$

  In our application we follow [58] and set $H = \theta\sqrt{n}$ and $\theta = 1$, $g(x) = x \wedge (1 - x)$. Hence we will have $\psi_1 = 1$ and, $\psi_2 = \frac{1}{12}$.
- Realized semi-variance $(+, -)$
  Positive $(+)$ and negative $(-)$ realized semivariances [59] measure upside and downside risk respectively, as follows:

$$RSV^+(X)_t = \sum_{i=1}^{n} r(X)_i^2 1_{(r(X)_i > 0)}$$

$$RSV^-(X)_t = \sum_{i=1}^{n} r(X)_i^2 1_{(r(X)_i < 0)} \tag{A.12}$$

  where 1 is a simple indicator function.
- Realized bipower variation
  The realized bipower variation [60] measures the diffusive component of the price process, isolating it from the variation caused by the jump components and it is equal to:

$$BV(X)_t := \frac{\pi}{2} \sum_{i=2}^{n} |r(X)_i||r(X)_{i-1}| \tag{A.13}$$

- Realized bipower variation (lag 2)

$$BV(X)_t := \frac{\pi}{2} \sum_{i=3}^{n} |r(X)_i||r(X)_{i-2}| \tag{A.14}$$

- Realized bipower semivariance $(+, -)$
  Realized bipower semivariances [59] are used to measure the upside and downside risk of the diffusive component:

$$BV^+(X)_t := \frac{\pi}{2} \sum_{i=2}^{n} |r(X)_i||r(X)_{i-1}| 1_{(r(X)_i > 0)}$$

$$BV^-(X)_t := \frac{\pi}{2} \sum_{i=2}^{n} |r(X)_i||r(X)_{i-1}| 1_{(r(X)_i < 0)}. \tag{A.15}$$

- Jump variation
  We use a modified version of the jump variation estimator [58] which is both non-negative and consistent. As hinted by the name, the jump variation estimator provides a measures of the discontinuous variability component:

$$JV(X)_t := \max(RV(X)_t - BV(X)_t, 0). \tag{A.16}$$

- Spot volatility
  We only compute the spot volatility (i.e., [61] and [37]) estimates on the block. The spot volatility measures the instantaneous volatility. The definition is consistent with the terminology commonly used in the literature on parametric stochastic volatility models in continuous-time:

$$SV(X)_t := \lim_{h \to 0} \{\mathbb{E}[([X, X]_{t+h} - [X, X]_t)/h] | \mathcal{F}_t\}. \tag{A.17}$$

  with $h \to 0$ being the time interval upon which the measure is computed.
- Average spot volatility
  The average spot volatility provides an historical average of the estimated spot volatilities:

$$\overline{SV}(X)_t := \frac{1}{t} \sum_{i=0}^{t} SV(X)_i. \tag{A.18}$$

TABLE 11. Protocol II: f1 scores based on US stocks for the handcrafted features. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		Econ		Tech-Quant		LOB	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Amazon	0.28	0.26	0.20	0.62	0.31	0.45
	Google	0.28	0.28	0.18	0.63	0.30	0.55
	Joint	0.36	0.25	0.23	0.56	0.27	0.51
MLP_2	Amazon	0.35	0.26	0.20	0.58	0.25	0.31
	Google	0.31	0.35	0.19	0.35	0.32	0.50
	Joint	0.21	0.25	0.24	0.49	0.29	0.50
MLP_3	Amazon	0.25	0.26	0.26	0.46	0.26	0.44
	Google	0.27	0.33	0.19	0.47	0.26	0.48
	Joint	0.28	0.21	0.33	0.63	**0.51**	0.56
MLP_4	Amazon	0.21	0.23	0.19	0.53	0.15	0.41
	Google	0.27	0.27	0.20	**0.65**	0.27	0.56
	Joint	0.21	0.26	0.27	**0.65**	0.21	0.59
MLP_5	Amazon	0.31	0.26	0.21	0.56	0.20	0.39
	Google	0.33	0.31	0.20	0.62	0.21	0.56
	Joint	0.36	0.30	0.24	0.52	0.35	0.57
CNN_1	Amazon	0.34	0.25	0.19	0.16	0.24	0.18
	Google	0.33	0.34	0.21	0.19	0.27	0.22
	Joint	0.35	0.27	0.19	0.22	0.29	0.20
CNN_2	Amazon	0.31	0.26	0.22	0.21	0.26	0.21
	Google	0.35	0.22	0.25	0.20	0.31	0.22
	Joint	0.37	0.29	0.23	0.24	0.30	0.23
LSTM_1	Amazon	0.33	0.22	0.23	0.22	0.28	0.15
	Google	0.31	0.35	0.22	0.23	0.33	0.14
	Joint	0.35	0.23	0.19	0.25	0.21	0.19
LSTM_2	Amazon	0.34	0.26	0.21	0.25	0.26	0.21
	Google	0.32	0.37	0.24	0.26	**0.41**	0.19
	Joint	0.37	0.25	0.20	0.27	0.42	0.22

TABLE 12. Protocol II: f1 scores based on US stocks for the fully automated features. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		LSTM AE	
Model	Stock	UnBal.	Bal.
MLP_1	Amazon	0.27	0.28
	Google	0.28	**0.31**
	Joint	0.27	0.25
MLP_2	Amazon	0.11	0.11
	Google	0.15	0.18
	Joint	0.19	0.25
MLP_3	Amazon	0.22	0.22
	Google	0.27	0.24
	Joint	0.23	0.21
MLP_4	Amazon	0.25	0.25
	Google	0.24	0.23
	Joint	0.22	0.26
MLP_5	Amazon	0.21	0.21
	Google	0.23	0.28
	Joint	0.24	**0.30**
CNN_1	Amazon	0.27	0.21
	Google	0.24	0.22
	Joint	0.25	0.19
CNN_2	Amazon	0.27	0.25
	Google	0.29	0.26
	Joint	0.30	0.25
LSTM_1	Amazon	0.19	0.21
	Google	0.33	0.27
	Joint	**0.33**	0.22
LSTM_2	Amazon	0.21	0.23
	Google	**0.34**	0.21
	Joint	0.28	0.24

3) NOISE AND UNCERTAINTY MEASURES

In this category, we incorporate two kinds of measures which are intimately linked to each other. We provide three different estimates for the integrated quarticity and two different estimates for the variance of the contaminating noise process. The integrated quarticity measures the degree of estimation error in the realized variance and can be consistently estimated through the realized quarticity estimators presented below for a fixed window size of 2000 events. The noise variance estimates provide a measure of the intensity of the noise process affecting the underlying price, as follows:

$$IQ_t = \int_0^t \sigma_s^4 \mathrm{d}s \tag{A.19}$$

with the noise variance estimates providing a measure of the contaminating:

- Realized quarticity [62]:

$$RQ_t = \frac{n}{3}\sum_{i=1}^{n}(X_i - X_{i-1})^4 \tag{A.20}$$

- Realized quarticity Tripower
  The tri-power quarticity [62] is a generalization of the realized bipower variation and is a consistent estimator for the integrated quarticity in the presence of jumps:

$$RQ_t = n\mu_{4/3}^{-3}\sum_{i=3}^{n}|r(X)_i|^{4/3}|r(X)_{i-1}|^{4/3}|r(X)_{i-2}|^{4/3} \tag{A.21}$$

with $\mu_p = \mathbb{E}(|Z|^p)$,where Z denotes a standard normally distributed random variable.

- Realized quarticity Quadpower
  A generalization of multipower variation measures led to the realized quadpower quarticity estimator proposed by [62] and it is equal to:

$$RQ_t = n\mu_1^{-4}\sum_{i=4}^{n}|r(X)_i||r(X)_{i-1}||r(X)_{i-2}||r(X)_{i-3}| \tag{A.22}$$

- Noise variance [35]:

$$NV_t = -\frac{1}{n-1}\sum_{i=2}^{n}(r(X)_i r(X)_{i-1}). \tag{A.23}$$

- Noise variance [36]:

$$NV_t = \frac{1}{2n}\sum_{i=1}^{n}(X_i - X_{i-1})^2. \tag{A.24}$$

4) PRICE DISCOVERY FEATURES

- Mid price weighted by order imbalance:

$$MidPrice_t = \frac{Ask * V_{Ask} + Bid * V_{Bid}}{V_{Ask} + V_{Bid}}. \tag{A.25}$$

- Volume imbalance:

$$VolImbalance = \frac{V_{Bid}}{V_{Ask} + V_{Bid}} \tag{A.26}$$

TABLE 13. Protocol II: f1 scores based on Nordic stocks for the handcrafted features. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		Econ		Tech-Quant		LOB	
Model	Stock	UnBal.	Bal.	UnBal.	Bal.	UnBal.	Bal.
MLP_1	Kesko Oyj	0.42	0.29	**0.63**	**0.56**	0.59	0.25
	Outokumpu Oyj	0.30	0.25	0.51	0.45	0.52	0.40
	Sampo Oyj	0.32	0.30	0.56	0.48	0.57	0.36
	Rautaruukki	0.19	0.37	0.42	0.48	0.42	0.41
	Wartsila Oyj	0.30	0.35	0.43	0.37	0.36	0.41
	Joint	0.37	0.34	0.48	0.43	0.45	0.42
MLP_2	Kesko Oyj	0.45	0.38	0.55	0.53	0.51	0.44
	Outokumpu Oyj	0.30	0.31	0.46	0.45	0.51	0.45
	Sampo Oyj	0.30	0.30	0.49	0.48	0.55	0.45
	Rautaruukki	0.31	0.35	0.38	0.37	0.40	0.41
	Wartsila Oyj	0.34	0.36	0.35	0.38	0.36	0.39
	Joint	0.32	0.34	0.34	0.42	0.37	0.44
MLP_3	Kesko Oyj	0.44	0.39	0.56	0.48	0.53	0.43
	Outokumpu Oyj	0.30	0.31	0.47	0.46	0.50	0.44
	Sampo Oyj	0.30	0.29	0.50	0.48	0.54	0.52
	Rautaruukki	0.34	0.37	0.40	0.42	0.41	0.41
	Wartsila Oyj	0.30	0.37	0.36	0.41	0.34	0.33
	Joint	0.33	0.33	0.49	0.43	0.48	0.43
MLP_4	Kesko Oyj	0.44	0.36	0.54	0.52	0.52	0.41
	Outokumpu Oyj	0.30	0.29	0.45	0.45	0.52	0.46
	Sampo Oyj	0.31	0.31	0.51	0.49	0.54	0.46
	Rautaruukki	0.33	0.35	0.41	0.40	0.42	0.42
	Wartsila Oyj	0.33	0.35	0.40	0.40	0.38	0.40
	Joint	0.30	0.33	**0.51**	0.41	0.46	0.41
MLP_5	Kesko Oyj	0.45	0.37	0.49	0.49	0.53	0.39
	Outokumpu Oyj	0.30	0.30	0.49	0.48	0.51	0.43
	Sampo Oyj	0.32	0.30	0.51	0.49	0.53	0.50
	Rautaruukki	0.32	0.35	0.43	0.40	0.42	0.41
	Wartsila Oyj	0.29	0.35	0.41	0.38	0.37	0.41
	Joint	0.32	0.33	0.50	0.44	0.44	**0.45**
CNN_1	Kesko Oyj	0.42	0.26	0.51	0.46	0.48	0.09
	Outokumpu Oyj	0.31	0.26	0.46	0.29	0.48	0.18
	Sampo Oyj	0.34	0.27	0.45	0.38	0.50	0.34
	Rautaruukki	0.32	0.38	0.40	0.38	0.40	0.29
	Wartsila Oyj	0.31	0.23	0.36	0.24	0.33	0.20
	Joint	0.32	0.25	0.44	0.26	0.45	0.29
CNN_2	Kesko Oyj	0.45	0.13	0.52	0.37	0.54	0.20
	Outokumpu Oyj	0.28	0.17	0.51	0.30	0.51	0.22
	Sampo Oyj	0.33	0.19	0.52	0.26	0.55	0.20
	Rautaruukki	0.40	0.16	0.41	0.29	0.40	0.28
	Wartsila Oyj	0.33	0.30	0.36	0.26	0.38	0.28
	Joint	0.31	0.30	0.49	0.26	0.47	0.27
LSTM_1	Kesko Oyj	0.43	0.28	0.52	0.50	0.54	0.14
	Outokumpu Oyj	0.31	0.24	0.45	0.34	0.49	0.24
	Sampo Oyj	0.32	0.31	0.50	0.39	0.51	0.30
	Rautaruukki	0.28	0.24	0.38	0.31	0.40	0.30
	Wartsila Oyj	0.33	0.32	0.35	0.29	0.39	0.27
	Joint	0.32	0.27	0.46	0.27	0.45	0.29
LSTM_2	Kesko Oyj	0.44	0.21	0.54	0.47	0.48	0.13
	Outokumpu Oyj	0.32	0.23	0.45	0.19	0.49	0.19
	Sampo Oyj	0.29	0.25	0.50	0.40	0.52	0.31
	Rautaruukki	0.32	0.30	0.38	0.30	0.39	0.30
	Wartsila Oyj	0.31	0.31	0.35	0.30	0.38	0.27
	Joint	0.34	0.26	0.46	0.26	0.48	0.27

- Bid-ask spread:

$$BA_{\text{spread}} = Ask - Bid. \tag{A.27}$$

- Normalized bid-ask spread
  The normalized bid-ask spread expresses the spread as the number of ticks between the bid and the ask price:

$$BA_{\text{spread}} = \frac{Ask - Bid}{TickSize}. \tag{A.28}$$

B. PROTOCOL II RESULTS

See Tables 11–14.

TABLE 14. Protocol II: f1 scores based on Nordic stocks for the fully automated features. *Note: Highlighted text shows the best f1 performance for: 1) Joint/Unbalanced, 2) Joint/Balanced, 3) Stock-Specific/Unbalanced, and 4) Stock-Specific/Balanced cases.*

		LSTM AE	
Model	Stock	UnBal.	Bal.
MLP_1	Kesko Oyj	0.35	0.30
	Outokumpu Oyj	0.20	0.19
	Sampo Oyj	0.28	0.26
	Rautaruukki	0.19	0.21
	Wartsila Oyj	0.28	0.22
	Joint	**0.37**	0.26
MLP_2	Kesko Oyj	0.34	0.29
	Outokumpu Oyj	0.20	0.18
	Sampo Oyj	0.27	0.21
	Rautaruukki	0.31	0.21
	Wartsila Oyj	0.27	0.22
	Joint	0.32	0.20
MLP_3	Kesko Oyj	0.32	**0.33**
	Outokumpu Oyj	0.20	0.17
	Sampo Oyj	0.26	0.29
	Rautaruukki	0.26	0.23
	Wartsila Oyj	0.26	0.26
	Joint	0.33	**0.29**
MLP_4	Kesko Oyj	0.30	0.31
	Outokumpu Oyj	0.20	0.19
	Sampo Oyj	0.28	0.32
	Rautaruukki	0.26	0.28
	Wartsila Oyj	0.33	0.27
	Joint	0.30	0.25
MLP_5	Kesko Oyj	0.30	0.28
	Outokumpu Oyj	0.20	0.19
	Sampo Oyj	0.18	0.17
	Rautaruukki	0.19	0.18
	Wartsila Oyj	0.18	0.18
	Joint	0.32	0.23
CNN_1	Kesko Oyj	0.28	0.26
	Outokumpu Oyj	0.29	0.27
	Sampo Oyj	0.26	0.27
	Rautaruukki	0.31	0.21
	Wartsila Oyj	0.30	0.22
	Joint	0.32	0.19
CNN_2	Kesko Oyj	0.29	0.13
	Outokumpu Oyj	0.27	0.17
	Sampo Oyj	0.29	0.19
	Rautaruukki	**0.36**	0.16
	Wartsila Oyj	0.31	0.24
	Joint	0.31	0.21
LSTM_1	Kesko Oyj	0.28	0.28
	Outokumpu Oyj	0.32	0.24
	Sampo Oyj	0.31	0.25
	Rautaruukki	0.27	0.25
	Wartsila Oyj	0.31	0.24
	Joint	0.33	0.27
LSTM_2	Kesko Oyj	0.31	0.21
	Outokumpu Oyj	0.33	0.23
	Sampo Oyj	0.33	0.23
	Rautaruukki	0.34	0.22
	Wartsila Oyj	0.32	0.22
	Joint	0.31	0.25

ACKNOWLEDGMENT

The authors would like to thank CSC-IT Center for Science, Finland, for generous computational resources. The views and conclusions expressed in this paper are solely those of the authors and do not necessarily reflect the views of Danmarks Nationalbank.

REFERENCES

[1] X.-Y. Qian, "Financial series prediction: Comparison between precision of time series models and machine learning methods," 2017, *arXiv:1706.00948*. [Online]. Available: https://arxiv.org/abs/1706.00948

[2] S. Siami-Namini and A. S. Namin, "Forecasting economics and financial time series: ARIMA vs. LSTM," 2018, *arXiv:1803.06386*. [Online]. Available: https://arxiv.org/abs/1803.06386

[3] L. Chen, Z. Qiao, M. Wang, C. Wang, R. Du, and H. E. Stanley, "Which artificial intelligence algorithm better predicts the chinese stock market?" *IEEE Access*, vol. 6, pp. 48625–48633, 2018.

[4] P. Nousi, A. Tsantekidis, N. Passalis, A. Ntakaris, J. Kanniainen, A. Tefas, M. Gabbouj, and A. Iosifidis, "Machine learning for forecasting mid price movement using limit order book data," 2018, *arXiv:1809.07861*. [Online]. Available: https://arxiv.org/abs/1809.07861

[5] J. Sirignano and R. Cont, "Universal features of price formation in financial markets: Perspectives from deep learning," 2018, *arXiv:1803.06917*. [Online]. Available: https://arxiv.org/abs/1803.06917

[6] M. Velay and F. Daniel, "Stock chart pattern recognition with deep learning," 2018, *arXiv:1808.00418*. [Online]. Available: https://arxiv.org/abs/1808.00418

[7] R. Dash and P. K. Dash, "A hybrid stock trading framework integrating technical analysis with machine learning techniques," *J. Finance Data Sci.*, vol. 2, no. 1, pp. 42–57, 2016.

[8] M. U. Gudelek, S. A. Boluk, and A. M. Ozbayoglu, "A deep learning based stock trading model with 2-D CNN trend detection," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov./Dec. 2017, pp. 1–8.

[9] P. Wang, "Pricing currency options with support vector regression and stochastic volatility model with jumps," *Expert Syst. Appl.*, vol. 38, no. 1, pp. 1–7, 2011.

[10] M. Zięba, S. K. Tomczak, and J. M. Tomczak, "Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction," *Expert Syst. Appl.*, vol. 58, pp. 93–101, Oct. 2016.

[11] A. Ntakaris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, *Mid-Price Prediction Based on Machine Learning Methods with Technical and Quantitative Indicators*. 2018. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3213389

[12] A. N. Kercheval and Y. Zhang, "Modelling high-frequency limit order book dynamics with support vector machines," *Quant. Finance*, vol. 15, no. 8, pp. 1315–1329, Jun. 2015. doi: 10.1080/14697688.2015.1032546.

[13] A. Ntakaris, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods," *J. Forecasting*, vol. 37, no. 8, pp. 852–866, 2018.

[14] H. Guo, "Limited stock market participation and asset prices in a dynamic economy," *J. Financial Quant. Anal.*, vol. 39, no. 3, pp. 495–516, 2004.

[15] M. Lettau and S. Ludvigson, "Consumption, aggregate wealth, and expected stock returns," *J. Finance*, vol. 56, no. 3, pp. 815–849, 2001.

[16] K. H. Chung and C. Chuwonganant, "Market volatility and stock returns: The role of liquidity providers," *J. Financial Markets*, vol. 37, pp. 17–34, Jan. 2018.

[17] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Using deep learning for price prediction by exploiting stationary limit order book features," 2018, *arXiv:1810.09965*. [Online]. Available: https://arxiv.org/abs/1810.09965

[18] M. Göçken, M. Özçalici, A. Boru, and A. T. Dosdogru, "Integrating metaheuristics and Artificial Neural Networks for improved stock price prediction," *Expert Syst. Appl.*, vol. 44, pp. 320–331, Feb. 2016.

[19] X. Zhang, T. Xue, and H. E. Stanley, "Comparison of econometric models and artificial neural networks algorithms for the prediction of baltic dry index," *IEEE Access*, vol. 7, pp. 1647–1657, 2019.

[20] J. Sirignano, "Deep learning for limit order books," 2016, *arXiv:1601.01987*. [Online]. Available: https://arxiv.org/abs/1601.01987

[21] M. Dixon, "Sequence classification of the limit order book using recurrent neural networks," *J. Comput. Sci.*, vol. 24, pp. 277–286, Jan. 2018.

[22] D. L. Minh, A. Sadeghi-Niaraki, H. D. Huy, K. Min, and H. Moon, "Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network," *IEEE Access*, vol. 6, pp. 55392–55404, 2018.

[23] Z. Zhang, S. Zohren, and S. Roberts, "DeepLOB: Deep convolutional neural networks for limit order books," 2018, *arXiv:1808.03668*. [Online]. Available: https://arxiv.org/abs/1808.03668

[24] N. Passalis, A. Tsantekidis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Time-series classification using neural bag-of-features," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, Aug./Sep. 2017, pp. 301–305.

[25] D. T. Tran, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Tensor representation in high-frequency financial data for price change prediction," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov./Dec. 2017, pp. 1–7.

[26] D. T. Tran, A. Iosifidis, J. Kanniainen, and M. Gabbouj, "Temporal attention-augmented bilinear network for financial time-series data analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1407–1418, May 2019.

[27] B. Zheng, E. Moulines, and F. Abergel, "Price jump prediction in limit order book," 2012, *arXiv:1204.1381*. [Online]. Available: https://arxiv.org/abs/1204.1381

[28] J. Alberg and Z. C. Lipton, "Improving factor-based quantitative investing by forecasting company fundamentals," 2017, *arXiv:1711.04837*. [Online]. Available: https://arxiv.org/abs/1711.04837

[29] O. B. Sezer, A. M. Ozbayoglu, and E. Dogdu, "An artificial neural network-based stock trading system using technical analysis and big data framework," in *Proc. SouthEast Conf.*, 2017, pp. 223–226.

[30] J. Han, J. Hong, N. Sutardja, and S. F. Wong, "Machine learning techniques for price change forecast using the limit order book data," Tech. Rep., 2015. [Online]. Available: http://jcyhong.github.io/assets/machine-learning-price-movements.pdf

[31] K. Kanagal, Y. Wu, and K. Chen. (2017). *Market Making With Machine Learning Methods*. [Online]. Available: https://web.stanford.edu/class/msande448/2017/Final/Reports/gr4.pdf

[32] J. Doering, M. Fairbank, and S. Markose, "Convolutional neural networks applied to high-frequency market microstructure forecasting," in *Proc. 9th Comput. Sci. Electron. Eng. (CEEC)*, Sep. 2017, pp. 31–36.

[33] M. Mäkinen, A. Iosifidis, M. Gabbouj, and J. Kanniainen, *Predicting Jump Arrivals in Stock Prices Using Neural Networks With Limit Order Book Data*. 2018. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3165408

[34] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Using deep learning to detect price change indications in financial markets," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, Aug./Sep. 2017, pp. 2511–2515.

[35] R. C. A. Oomen, "Properties of realized variance under alternative sampling schemes," *J. Bus. Econ. Statist.*, vol. 24, no. 2, pp. 219–237, 2006. [Online]. Available: http://www.jstor.org/stable/27638871

[36] L. Zhang, P. A. Mykland, and Y. Aït-Sahalia, "A tale of two time scales: Determining integrated volatility with noisy high-frequency data," *J. Amer. Stat. Assoc.*, vol. 100, no. 472, pp. 1394–1411, 2005.

[37] T. G. Andersen, T. Bollerslev, and F. X. Diebold, "Parametric and nonparametric volatility measurement," in *Handbook of Financial Econometrics: Tools and Techniques*, vol. 1. Amsterdam, The Netherlands: Elsevier, 2010, ch. 2, pp. 67–137.

[38] S. Lahmiri, "Wavelet low- and high-frequency components as features for predicting stock prices with backpropagation neural networks," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 26, no. 2, pp. 218–227, 2014.

[39] C. M. Bishop, *Neural Networks for Pattern Recognition*. London, U.K.: Oxford Univ. Press, 1995.

[40] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic differentiation in machine learning: A survey," 2015, *arXiv:1502.05767*. [Online]. Available: https://arxiv.org/abs/1502.05767

[41] T. Dozat, "Incorporating nesterov momentum into adam," in *Proc. ICLR*, 2016. [Online]. Available: https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ

[42] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[43] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *Proc. IEEE 19th Conf. Bus. Inform. (CBI)*, vol. 1, Jul. 2017, pp. 7–12.

[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[45] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. Xu, "Attention-based bidirectional long short-term memory networks for relation classification," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2016, pp. 207–212.

[46] C.-Y. Liou, W.-C. Cheng, J.-W. Liou, and D.-R. Liou, "Autoencoder for words," *Neurocomputing*, vol. 139, pp. 84–96, Sep. 2014.

[47] C.-Y. Liou, J.-C. Huang, and W.-C. Yang, "Modeling word perception using the Elman network," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3150–3157, 2008.

[48] M. Qi and G. P. Zhang, "Trend time–series modeling and forecasting with neural networks," *IEEE Trans. Neural Netw.*, vol. 19, no. 5, pp. 808–816, May 2008.

[49] M. Butler and D. Kazakov, "The effects of variable stationarity in a financial time-series on artificial neural networks," in *Proc. IEEE Symp. Comput. Intell. Financial Eng. Econ. (CIFEr)*, Apr. 2011, pp. 1–8.

[50] T. Y. Kim, K. J. Oh, C. Kim, and J. D. Do, "Artificial neural networks for non-stationary time series," *Neurocomputing*, vol. 61, pp. 439–447, Oct. 2004.

[51] M. M. Dacorogna, R. GenÃğgay, U. A. MÃijller, R. B. Olsen, and O. V. Pictet, Eds., "4—Adaptive data cleaning," in *An Introduction to High-Frequency Finance*. San Diego, CA, USA: Academic, 2001, pp. 82–120. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780122796715500071
[52] C. T. Brownlees and G. M. Gallo, "Financial econometric analysis at ultra-high frequency: Data handling concerns," *Comput. Statist. Data Anal.*, vol. 51, no. 4, pp. 2232–2245, 2006.
[53] O. E. Barndorff-Nielsen, P. R. Hansen, A. Lunde, and N. Shephard, "Realized kernels in practice: Trades and quotes," *Econ. J.*, vol. 12, no. 3, pp. C1–C32, 2009.
[54] F. Chollet. (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras
[55] T. G. Andersen and T. Bollerslev, "Answering the skeptics: Yes, standard volatility models do provide accurate forecasts," *Int. Econ. Rev.*, vol. 39, no. 4, pp. 885–905, 1998.
[56] O. E. Barndorff-Nielsen, P. R. Hansen, A. Lunde, and N. Shephard, "Designing realized kernels to measure the ex post variation of equity prices in the presence of noise," *Econometrica*, vol. 76, no. 6, pp. 1481–1536, 2008.
[57] J. Jacod, Y. Li, P. A. Mykland, M. Podolskij, and M. Vetter, "Microstructure noise in the continuous case: The pre-averaging approach," *Stochastic Processes Appl.*, vol. 119, no. 7, pp. 2249–2276, 2009.
[58] K. Christensen, R. C. A. Oomen, and M. Podolskij, "Fact or friction: Jumps at ultra high frequency," *J. Financial Econ.*, vol. 114, no. 3, pp. 576–599, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304405X14001548
[59] O. Barndorff-Nielsen, S. Kinnebrock, and N. Shephard, "Measuring downside risk—Realised semivariance," in *Volatility and Time Series Econometrics: Essays in Honor of Robert Engle*. Oxford Scholarship Online, 2010.
[60] O. E. Barndorff-Nielsen and N. Shephard, "Power and bipower variation with stochastic volatility and jumps," *J. Financial Econ.*, vol. 2, no. 1, pp. 1–37, 2004.
[61] O. E. Barndorff-Nielsen and N. Shephard, "Econometric analysis of realized volatility and its use in estimating stochastic volatility models," *J. Roy. Stat. Soc. B*, vol. 64, no. 2, pp. 253–280, 2002.
[62] O. E. Barndorff-Nielsen and N. Shephard, "Econometrics of testing for jumps in financial economics using bipower variation," *J. Financial Econ.*, vol. 4, no. 1, pp. 1–30, 2006. doi: 10.1093/jjfinec/nbi022.

ADAMANTIOS NTAKARIS received the B.Sc. degree in mathematics from the Aristotle University of Thessaloniki, in 2009, and the M.Sc. degree in financial modeling and optimization from The University of Edinburgh, in 2014. In 2014, he completed an industrial placement at Standard Life Investments, Edinburgh. Before commencing the Ph.D. degree, from 2014 to 2016, he was an Effective Interest Rate Analyst with CitiGroup Investment Bank, Edinburgh, and from 2010 to 2013, as a Math Olympiad Coach in Thessaloniki. He is currently an Early Stage Researcher within the Marie Curie BigDataFinance training network at the Department of Signal Processing, Tampere University of Technology.

GIORGIO MIRONE received the B.Sc. degree in economics and finance from the Universitat de Barcelona and The University of Melbourne, the M.Sc. degree in finance from the Universitá degli Studi di Siena, and the Ph.D. degree from in economics and business economics the Center for Research in Econometric Analysis of Time Series (CREATES), Aarhus University, in 2018. He held a visiting position at Oxford University with an Ass. Prof. A. B. Kock. He is currently a Quantitative Analyst with Danmarks Nationalbank and a former Early Stage Researcher within the Marie Curie BigDataFinance international training network. His research interests include econometrics, financial econometrics, volatility estimation and forecasting, statistical learning, and signal processing.

JUHO KANNIAINEN received the Ph.D. degree in quantitative finance from the Tampere University of Technology, where he is currently a Professor of financial engineering with the Faculty of Information Technology and Communication Sciences. His research agenda focused on statistical computing and data science in finance and risk management. In his research, he is not only using traditional quantitative methods but also modern data science approaches, namely complex networks techniques and machine learning methods with rich data sets. His focus in finance is on derivative pricing, financial econometrics, order book dynamics and liquidity, and financial networks. He has published finance and information technology in prestigious journals including *Review of Finance* and the IEEE Transactions on Neural Networks and Learning Systems. He has been coordinating two international EU projects, BigDataFinance and HPCFinance.

MONCEF GABBOUJ (F'11) received the B.S. degree in electrical engineering from Oklahoma State University, Stillwater, OK, USA, in 1985, and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1986 and 1989, respectively. He was an Academy of Finland Professor, from 2011 to 2015. He is currently a Professor of signal processing with the Laboratory of Signal Processing, Tampere University of Technology, Tampere, Finland. He guided 40 Ph.D. students and has published 650 papers. His current research interests include multimedia content-based analysis, indexing and retrieval, machine learning, nonlinear signal and image processing and analysis, voice conversion, and video processing and coding. He is a member of the Academia Europaea and the Finnish Academy of Science and Letters. He organized several tutorials and special sessions for major IEEE conferences and EUSIPCO. He is the past Chairman of the IEEE CAS TC on DSP and a Committee Member of the IEEE Fourier Award for Signal Processing. He served as an Associate Editor and a Guest Editor of many IEEE international journals and a Distinguished Lecturer for the IEEE CASS.

ALEXANDROS IOSIFIDIS (SM'16) held Postdoctoral Researcher positions with the Tampere University of Technology, Finland, and the Aristotle University of Thessaloniki, Greece. He is currently an Associate Professor in electrical and computer engineering with Aarhus University, Denmark. He has contributed more than ten R&D projects financed by EU, Greek, Finnish, and Danish funding agencies and companies. He has coauthored 55 articles in international journals and 78 papers in international conferences proposing novel machine learning techniques and their application in a variety of problems.

Dr. Iosifidis served as an Officer of the Finnish IEEE Signal Processing-Circuits and Systems Chapter, from 2016 to 2018. He received prestigious awards, including the Academy of Finland Postdoc Fellowship and the H. C. Oersted Forskerspirer Prize for research excellence at a young age. He served as an Area Chair for the IEEE ICIP, in 2018 and 2019, EUSIPCO 2019, and the Technical Program Chair for the IEEE ICASSP 2019. He has coorganized special issues/sessions in international journals/conferences focusing on topics of machine learning and big data analysis. He is currently serving as an Associate Editor for the IEEE Access and *Neurocomputing* journals, and an Area Editor for the *Signal Processing: Image Communication* journal.