Expert Systems with Applications 159 (2020) 113565

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

The impact of stock market price Fourier transform analysis on the Gated Recurrent Unit classifier model

Dragana Radojičić [*], Simeon Kredatus

TU Wien, Institute of Statistics and Mathematical Methods in Economics, Wiedner Hauptstr. 8/ E105-01 & -05 FAM, 1040 Vienna, Austria

ARTICLE INFO

Article history:
Available online 19 May 2020

Keywords:
Limit Order Book
Recurrent Neural Networks
Technical indicators
Fourier transform

ABSTRACT

In this paper, we suggest new feature extraction models based on the stock market price signal analysis. In particular, we study the behavior observed in signals originating from different sources, such as prices of different Limit Order Book levels and open, close, low, high prices of the preselected time intervals. We apply Fourier transformation to extract new features. Moreover, we evaluate if the performance of the model based on the Gated Recurrent Unit (GRU) architecture is improved when we select features utilizing the proposed methods. Furthermore, we benchmark the performance of new indicators on the GRU model and provide quantified results proving the significant performance improvement obtained by incorporating the suggested features.

1. Introduction

The past years have brought the new era of automation that has affected a wide range of fields which have an impact on human lives. The leverage of Machine Learning has attracted the attention of academic researchers in many fields, see Li, Jiang, Yang, and Wu (2018).

Forecasting future trends of the stock market using historical data has become very attractive for both researchers from academia and business, see Das, Mishra, and Rout (2019). Since the trading has experienced automation and digitization, stocks have substituted the physical trading concept by electronic trading. While the average human's eye blinks last up to 400 ms, the high-frequency trading (HFT) algorithms are processing information with respect to the time-scale of the microseconds. Therefore, more trades are occurring automatically and the whole role of trader is going into the direction of a data scientist, who uses algorithms based on Machine Learning for automatic trading (see lle Calvez & Cliff (2018)).

This topic is tempting for the hedge funds and financial institutions, but also for scientists, who have obtained several high-impact applications of it in finance. In Sirignano and Cont (2019) the authors presented a universal model for different stocks to predict the direction of price moves using Deep Learning Neural Networks (DLNNs) given the price and order flow history. An approach for hedging a portfolio of derivatives using reinforcement learning methods, which outperforms the hedging in a Heston model without transaction costs, is proposed in Buehler, Gonon, Teichmann, and Wood (2019). The authors in Chatzis, Siakoulis, Petropoulos, Stavroulakis, and Vlachogiannakis (2018) used deep learning and boosting approaches in order to predict stock market crisis episodes. They presented an interesting forecasting mechanism of the probability of a stock market crash event in various time frames, combining different machine learning algorithms. Moreover, financial network indicators that can be applied to global stock market investment strategies were investigated in Lee, Cho, Kwon, and Sohn (2019) and combined with several machine learning approaches. The first 3-dimensional convolutional neural networks (CNNs) model designed for stock market prediction was proposed in Hoseinzade and Haratizadeh (2019) and the presented framework significantly outperformed shallow artificial neural networks.

Many researchers have dealt with problems regarding the stock market. A stock market trading system based on fuzzy logic rules was proposed in Brzeszczyński and and Ibrahim (2019) and both direct and indirect channels of foreign information transmission were investigated. A stock-based collaborative filtering algorithm for stock prediction proposed in Zheng, Gao, Yin, and Rabarison (2019) relies on the transmission effect among different stocks. A challenging topic of the impact of investor sentiment on stock market volatility was explored in Xu, Wang, Jiang, and Zhang (2019), where the authors proposed a new model and implemented the nonlinear quantile regression on mixed frequency data by introducing a kernel function. This approach was shown to be superior

* Corresponding author.
E-mail address: dragana@fam.tuwien.ac.at (D. Radojičić).

https://doi.org/10.1016/j.eswa.2020.113565
0957-4174/© 2020 The Author(s). Published by Elsevier Ltd.

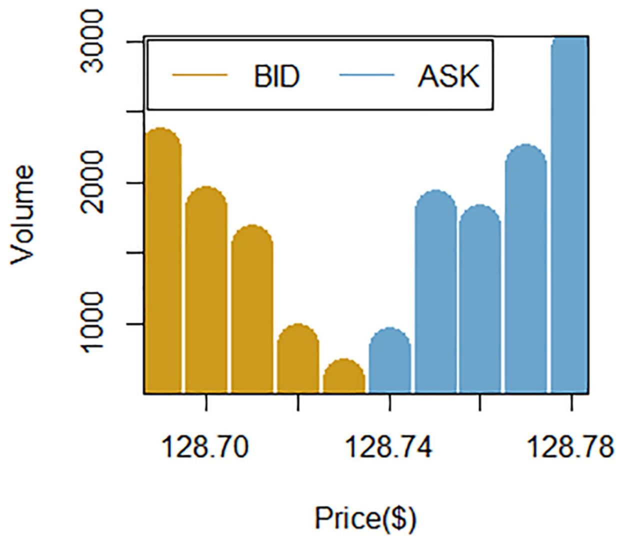

Fig. 1. Snapshot of the NASDAQ limit order book for AAPL shares (Feb 20, 2015) at 10:20:51 for 5 levels. On the bid/ask side are placed outstanding buy/sell orders (gold/blue), and the best bid price is $128.73, while the best ask price is $128.74.

to the previously used ones for the same problem, according to the empirical results.

A novel fuzzy recommendation system for stock market investors was proposed in Naranjo and Santos (2019) and it was shown that this approach is profitable and presents high stability. Furthermore, in Dixon (2018a) a simple trade execution model that evaluates the performance of a trading strategy using a trade information matrix, which is the enhanced concept of a confusion matrix, is established.

1.1. The Limit Order Book

The Limit Order Book (LOB) keeps track of all incoming and outgoing limit orders registered at the financial market. The book consists of two sides: the orders that need to be bought are placed at the bid side, while the orders that need to be sold are placed at the *ask side*. The LOB is defined on a discrete price grid, and each point represents the price level on which all active orders with that price are placed. At a given time t, the best bid price P_t^b depicts the maximum price the buyers are willing to pay for a given asset, while the best ask price P_t^a depicts the minimum price the sellers are willing to sell an asset for. Fig. 1 depicts a snapshot of the NASDAQ limit order book for an Apple stock. For a further mathematical description of the LOB concept see Section II in Gould et al. (2013).

1.2. The data used for this research

Several papers stated that the LOB is informative and conveys predictive information, see Cont, Kukanov, and Stoikov (2014), Zheng, Moulines, and Abergel (2012) and Palguna and Pollak (2016). For this research, we used high-quality limit order book data tool LOBSTER[1] (Limit Order Book System-Efficient Reconstructor). LOBSTER uses NASDAQ's Historical TotalView-ITCH files to replicate entire NASDAQ stock exchange LOB (from the 27th of June 2007 up to the two days ago from the current day). LOBSTER contains 'message' file and 'orderbook' file for the selected trading day and for the selected ticker, e.g. Apple (AAPL), Microsoft (MSFT), Tesla (TSLA), etc.

The 'message' file carries information of the kind of event which had updated the LOB and each raw has the following columns: Time, Type of Event, Order ID, Size of the order, Price, Direction of the trade (buy/sell). The column 'Type of Event' represents the type of event that caused the update of the LOB (i.e. the following events: submission of a new order, cancellation, deletion of an order, execution of a visible order, execution of a hidden order, trading halt).

1.3. Motivation and main contribution

The LOB is updating up to 18 quadrillion messages per day, requiring nanosecond precision. Because of the mentioned frequencies, market behavior has become seemingly harder to model, and therefore, our idea is to employ advanced signal processing techniques accompanied by non-linear machine learning models used for decision making. In order to apply machine learning approaches, such as the Long short-term memory (LSTM) network (initially introduced in Hochreiter & Schmidhuber (1997)) or Gated recurrent unit networks (initially introduced in Cho et al. (2014)), we have to extract relevant features from the market data. Apart from having features extracted from the limit order book data, we also include standardly known technical analysis indicators.

The main idea is to classify the vector of market data, which consists of various extracted market data information and technical analysis indicators, into one of the labels from the sets $S_+ = \{\text{buy}, \text{idle}\}$ or $S_- = \{\text{sell}, \text{idle}\}$, if we consider semi-strategy emitting buy/sell signals respectively.

The true price of the equity and the number of orders sitting at each LOB level can be perceived as a discrete signal over time. Therefore, our idea is to process it the same way we process signals out of electronic devices.

As every time series, the signal from the data produced by the stock market can be decomposed onto the series of sines and cosines which can uncover some latent properties which had not been visible before.

There exists a large set of standardly known technical indicators that are regularly used in the trading industry such as, but not exclusively, Bollinger Bands, Moving average, Relative Strength Index, Oscillation indicators (for more information see Colby & Meyers (1988)). However, we propose a few new indicators that employ Fourier transformations (defined later in Section 4). The explanation of each of these well known technical indicators exceeds the scope of this paper, but more on this topic can be found in Murphy (1999) and Colby and Meyers (1988).

Initially, we introduce new technical indicators for the stock market utilizing Fourier Transformation. Secondly, we evaluate the quality of new features and prove that these features enhance the performance of our benchmark model based on the GRU.

As another contribution, we show that the proposed GRU model fed with features obtained with Stochastic Universal Sampling accompanied with one of the two proposed feature ranking methods (explained later in Section 3) overperforms the model fed with randomly selected features.

The rest of the paper is organized as follows. After introducing the data transformation and the limit order book features extraction concept in Section 2, the proposed procedure and the feature selection approaches are presented in Section 3. Further, we present the newly proposed features in Section 4. The experimental results and statistical verification are presented in Section 5. The general conclusions and prospective future work are stated in Section 6. All acronyms used in the paper can be found in Table 5.

2. The LOB features extraction

In order to prepare the data for the main part of our research, the very first step is to dynamically reconstruct the LOB data and

[1] Lobster academic research data. https://lobsterdata.com. Accessed: 2018–08-05.

extract the features of interest that are relevant, such as the number of canceled limit orders, the number of executed hidden orders, the best bid price, technical analysis indicators, etc. The order book reconstruction system for LOBSTER and order flow visualization is, in a structured way, proposed in Huang and Polak (2011). However, we have used our customized approach to prepare data for this research, we incorporated the processing engine proposed in Radojičić, Kredatus, and Rheinländer (2018).

The technical pre-processing of the data Define x_t as the vector depicting the LOB shape at time t together with a corresponding event that caused an update of the LOB status. At this stage, the vector is obtained by simply merging each row from the 'orderbook' file with the respective row from the 'messagebook' file, and it looks like:

$x_t = (bidLevel_1, bidVolume_1, askLevel_1, askVolume_1, bidLevel_2, bidVolume_2, askLevel_2, askVolume_2, \ldots, bidLevel_n, bidVolume_n, askLevel_n, askVolume_n, Time, EventType, OrderId, Size, Price, Direction)$.

Note that the shape of the vector x_{t+1} is determined by the event that is encapsulated in the vector x_t. At this position of the data reconstruction we are working with the following data set:

$$\mathscr{D} = \{x_t | 0 \leqslant t \leqslant amount\ of\ events\ per\ day\}.$$

The data transformation The pipeline of the data transformation process consists of three major parts:

1. Raw aggregation: Since there is a huge amount of events per day (over one million) for each trading ticker, there is an enormous amount of data. Hence, in order to be able to extract the insights a data aggregation with respect to the 180 s time interval is involved. Firstly, we define a complete data partitioning over a single trading day data as $\mathscr{P}_{180} = \left\{\mathscr{T}_{j\cdot 180}^{(j+1)\cdot 180} | 0 \leqslant j < \lfloor \frac{trading\ day\ duration}{180} \rfloor \right\}$, where $\mathscr{T}_{j\cdot 180}^{(j+1)\cdot 180}$ is the part of a data identified by the start time $j \cdot 180$ and the end time $(j+1)\cdot 180$, i.e. $\mathscr{T}_{j\cdot 180}^{(j+1)\cdot 180} = \{x | x \in \mathscr{D} \ \wedge \ x_{time} \geqslant j \cdot 180 \ \wedge \ x_{time} \leqslant (j+1)\cdot 180\}$.
   During the raw aggregation stage, from each time interval $\mathscr{T}_{j\cdot 180}^{(j+1)\cdot 180}$ features of interest are extracted, such as: a number of canceled limit orders, open price, close price, etc.
   Formally, we define the set $\mathscr{A} = \{a_i\}_{i=1}^{m}$ as a set of functions for the raw aggregation, which contains functions such as: extract open price, extract close price, extract maximum price, extract minimum price, calculate the number of submitted ask (bid) trades, calculate the number of executed ask (bid) trades, calculate the number of executed hidden trades, etc.
   The aggregation functions have as the input the part of data $\mathscr{T}_{j\cdot 180}^{(j+1)\cdot 180}$. Then, the output vector $y_{j\cdot 180}^{(j+1)\cdot 180} = \left(a_1\left(\mathscr{T}_{j\cdot 180}^{(j+1)\cdot 180}\right), \ldots, a_m\left(\mathscr{T}_{j\cdot 180}^{(j+1)\cdot 180}\right)\right)$ is obtained by applying custom aggregation functions over the provided data. We iterate over the set $\mathscr{A}$, and apply each function from the set $\mathscr{A}$ on the data set $\mathscr{T}_{j\cdot 180}^{(j+1)}$.
   By denoting $y_j = y_j^{(j+1)\cdot 180}$, at the end of this stage we have the following chronologically ordered data set $\mathscr{D}_{180} = \{y_j | j = 0, \ldots, |\mathscr{P}_{180}| - 1\}$.
2. Technical indicators: In order to be able to get insights about the market behavior we need to involve technical analysis indicators (see Murphy (1999) for a description). Those technical indicators are computed via an open-source library *ta-lib*[2], which contains the algorithms' implementations for standardly known technical indicators used in the trading industry. It is emphasized in Menkhoff (2010) that the technical analysis is the foremost approach in forecasting analysis for the short term. In order to compute technical indicators for the provided data set $\mathscr{D}_{180}$, we introduce a new partitioning such that for each row in the data a new partitioning item is created and it contains chronologically all previous rows including the current row. Thus, we define the data set $\mathscr{W}_{180}$ as:
$$\mathscr{W}_{180} = \left\{(y_1), (y_1, y_2), \ldots, \left(y_1, y_2, \ldots, y_{|\mathscr{P}_{180}|-1}\right)\right\}.$$
Let $\mathscr{M} = (m_j)_j$ be the set of inherent functions present in the *ta-lib*. For each $w_i = (y_1, y_2, \ldots, y_i) \in \mathscr{W}_{180}$ we iterate over the set $\mathscr{M} = (m_j)_j$, and apply each function on a w_i, so the output $m_j(w_i)$ is a technical indicator. Formally, we introduce the data set consisting of technical indicators
$$\mathscr{L}_{180} = \left\{(m_1(w_i), \ldots, m_n(w_i))_i | w_i \in \mathscr{W}_{180} \wedge n = |\mathscr{M}|\right\}. \quad (1)$$
By employing the described procedure, we managed to construct the new data set $\mathscr{F}_{180} = [\mathscr{L}_{180}, \mathscr{D}_{180}]$ which contains the information of the features from the aggregated LOB enhanced by the technical indicators.
3. Data labeling: In order to prepare our dataset for learning, we assign a label to each vector in a training set. Since we consider semi-strategy omitting buy signals we label each vector of a training set with 1 if it is a good time for a *buy* trade with respect to the targeted risk-reward ratio (RRR) until the end of each trading day; and with 0 if the price goes below our risk level (stop-loss level). We have labelled with respect to the $REWARD = 0.08$ and $RISK = 0.04$, see algorithm in Appendix A.1.

3. Proposed methodology

3.1. Background

The beginning of this section contains introduction theory used later in the paper.

Stochastic Universal Sampling Stochastic Universal Sampling (SUS), introduced in Baker (1987), is an approach for selecting solutions from the population in genetic algorithms, using a single random value sample by choosing them at evenly spaced intervals. This approach provides that individuals are being selected with respect to their fitness, but still giving a chance even for individuals with lower fitness values to be selected.

Gated Recurrent Unit Gated Recurrent Unit (GRU), introduced in Cho et al. (2014), is a variation on Long Short Term Memory (LSTM) recurrent neural network (initially introduced in Hochreiter & Schmidhuber (1997)). While both LSTM and GRU networks have the ability to seize long-term and short term dependencies, GRU networks are faster and employ fewer parameters. GRU is a powerful tool that can be applied to various problems. For example, in order to perform answer selection, the authors in Tang, Rong, Qin, Yang, and Xiong (2020) applied the review mechanism on the GRU proposing a new n-Gated Recurrent Unit with Review (nGRUR) model.

Recurrent Neural Networks (RNNs) have many applications. For example, the author in Chow (2020) constructed a deep recurrent neural network to predict the prices of vehicle license plates. On the other hand, in Liu (2019) a new unbiased evaluation of the two modeling techniques for regression problems: Support Vector Machines and Long Short Term Memory Recurrent Neural Networks (LSTM RNNs), and the experimental results using financial data showed that LSTM RNNs had a very good performance. RNNs have already been used, as a powerful tool, in modeling LOB dynamics. An application of the deep learning architectures for a classification modeling in high-frequency trading used to predict

[2] Ta-lib: Technical analysis library. https://www.ta-lib.org/. Accessed: 2018–08-30.

short-term price movement based on the information of the market depth is developed in Dixon, Polson, and Sokolov (2017). Furthermore, the next price-flip event in limit order book using the ability of the RNN is modeled in Dixon (2018b).

3.2. Feature selection

We work with the set of features $\Phi = \{f_1, f_2, f_3, \ldots, f_n\}$, which contains features extracted from the LOB shape enhanced with the technical indicators.

By plugging all the features from Φ into the model, the model tends to overfit. Therefore, our mission strains down to select a subset $\Phi_1 \subset \Phi$ of features, such that the F_1 score of our buy semi-strategy (a strategy emitting *buy* or *idle* signals) is maximized on the training dataset and yet preserving similar performance on the test dataset (i.e. holding good generalization properties).

Since each feature provides unique information, we want to navigate thought this huge vector space towards those that could help us decide if we want to open position or idle. For the feature selection process, we assign a *SCORE* value to each feature. This *SCORE* value is assigned with respect to the different measures, namely:

1. Autocorrelation.
2. Mutual-information.

We assume the following two hypothesises.

1. The higher absolute value of the feature autocorrelation means higher likely-hood of the indicator imposing repetitive pattern thus we can infer knowledge easier.
2. Another hypothesis is that the feature having the higher value of mutual information with respect to the close price holds more information relevant for the decision buy/idle.

In order to determine the optimal lag, i.e. time-period with respect to which we calculate autocorrelation of each feature, we observe the trade duration distribution. We have obtained that most of the trades duration falls between 22 and 40 time-periods. Note that one time-period is three minutes long, as we employ data aggregation over 180 s time interval, i.e. 40 time-periods is equal to 120 min. Thus, in order to calculate a score value for each feature, the following calculation is performed. For each lag from the $[22, 40]$, interval we calculate the autocorrelation value and then the average value from that set of the autocorrelation values is assigned as a score value for the corresponding feature.

Regarding the second hypothesis, given a feature X (e.g. moving average), we want to compute mutual information between feature X and the close price. As each close price value is a discrete value from the broad range, it potentially can happen that the $P(close\_price = C)$ converges to 0, given that C could be the unique price of the stock that never repeats itself.

In order to mitigate this situation when many stock close prices would have a marginal probability of 0, rather than computing mutual information between X and the close price itself, we fit 400 clusters via the K-means algorithm (see Krishna & Murty (1999)) into the close price series. Therefore, we assign each close price value efficiently into one of 400 clusters. We treat this as a new categorical variable, and rather than computing mutual information between the close price and the feature X, we compute the mutual information between feature X and our newly introduced categorical variable.

Once a *SCORE* value is assigned to each feature, we implemented Stochastic universal sampling (SUS) in order to select features with respect to their score (higher the score, higher the likelihood for the feature to be selected).

3.3. Two groups of features

In our experiment we consider two groups of features:

1. The old-fashioned features – consisting of the features extracted from the LOB and standardly known technical indicators;
2. The Fourier transformation based features – this set of features contains old-fashioned-ones and features that employ Fourier transformation (defined in the later Section 4).

3.4. Proposed procedure

Having the whole training set labeled as described in the paragraph 'Data labelling' in Section 2, we train a GRU model to predict the respective label.

We propose the following procedure:

1. We establish a benchmark GRU model that assigns the label to the market vector by employing only the old-fashioned features (no Fourier transform based ones). In order to maximize the accuracy, we select only a subset of features by assigning the *SCORE* value and then run the SUS algorithm to select the most appropriate features (the higher *SCORE* value more likely for the feature to be selected). Since we use the SUS, we repeat the process multiple times to obtain a good statistical sample. As a key performance indicator, we measure F_1 score, more precisely its mean and standard deviation.
2. We propose and implement new Fourier transformation based features as part of every market vector data.
3. Then, we re-run the measurement as described in the first step, but with the difference that for this step the data set additionally contains Fourier based transformation features.
4. Finally, we compare the mean values of the F_1 score obtained for the measurement of the model containing the proposed Fourier transformation features (from step 3) with the benchmark model trained on top of the data-set not containing the Fourier transformation features (from step 1).

As the main goal of our work, we compare the performance of the GRU model when fed with the Fourier transformation based features and the performance of the same model fed with the old-fashioned feature set only. Therefore we seek to confirm the following hypothesis:

Extracting the Fourier based properties of the stock market signal and employing the SUS with respect to the feature's assigned SCORE value to choose the features, significantly improves F_1 score of the GRU model.

Firstly, for the benchmark model, we use the dataset consisting only of the old-fashioned features. Then, we use the dataset with both old-fashioned features and the Fourier transformation based features. We split either of the datasets into three parts: Train (60% of data), Validate (20% of data) and Test (20% of data).

Further,we apply the proposed feature selection approach and train the model using the *Tensorflow*[3] defined GRU model - on top of train and validate parts of the datasets. The test split of the respective dataset is never presented to the model during training and this is the part of data we evaluate our F_1 metric score upon. This guarantees unbiased results and proves that the model generalizes well on previously unseen data.

[3] https://www.tensorflow.org/

Table 1
Summary of the GRU model used with respective input and output dimensions.

Layer name	Input shape	Output shape	Explanation
GRU layer	(batch-size, lookback-period, 200)	(batch_size, 64, 50)	Input: 200 features with lookback period of 64 market ticks
Flatten	(batch_size, 64, 50)	(batch_size, 3200)	Flattens the output of a prior step, such that we can feed it into a dense layer
Dense layer with Sigmoid activation	(batch_size, 3200)	(batch_size, 1)	Outputs the final probability, whether the trade is suitable to open up the position

Since the SUS based method is used we repeat the upper procedure ten times and compute the mean and standard deviation of the F_1 metric.

3.5. GRU Model architecture

In order to depict the time dependency among different points in time of the market signal, we decided to use a model based on the GRU topology (see Table 1). Our model consists of three parts:

1. GRU layer of 50 units – which processes extracted features from every market vector. Each market vector has a dimensionality of 200. We let the GRU see prior 64 market vectors. We set the GRU layer parameter *return_sequences* to true, so we receive also interim outputs from every single "lookback" vector.
2. Flatten layer – this layer flattens the outcome of the GRU layer such that we can feed the data to the dense layer.
3. Dense layer – the final layer receiving a vector of size 3200 and outputting a single value between 0 and 1. It uses *sigmoid* activation function and the returned value represents the probability that the strategy shall issue a buy command ($\geqslant 0.5$) or *idle* otherwise (< 0.5).

4. The newly proposed features

In this section, we discuss newly proposed features, which are built on the assumptions that the *price* we observe (e.g. open, close price aggregates) is defined as its true *value* plus some *additive* component, which we approximate by the Price Oscillator.

The Price Oscillator is very similar to Moving Average Convergence Divergence (MACD). The Price Oscillator (PO) is defined as the difference between the moving averages, one shorter-period and one longer-period. Therefore, we interpret the PO as a difference between the observed close price signal (interpreted as the moving average of it with period $n = 1$) and the moving average of it with period $n = 5$:

$$PO = price - movingAverage(price, 5). \tag{2}$$

4.1. Spectra decomposition based features

The Fourier transform has had many applications in many different fields, especially engineering, physics, and applied mathematics. For example, the authors in Kithulgoda, Pears, and Naeem (2018) presented a strategy for incremental maintenance of the Fourier spectrum to changes in the concept that take place in data stream environments that had not been addressed before. In Eberlein, Glau, and Papapantoleon (2010) authors established conditions which ensure the existence of the Fourier transform valuation formulas in a general framework. On the other hand, a new method for optical image encryption using fractional Fourier transform was presented in Farah, Guesmi, Kachouri, and Samet (2020), while in Li, Wang, Xing, Jin, and Huang (2020) a fast Fourier transform method was used to determine dendrite arm spacing in directional solidification. In Zhylyevskyy (2010), the author proposed a new kernel-smoothed fast Fourier transform technique for pricing American options under stochastic volatility and showed a very good performance of the technique.

This subsection is devoted to getting the reader introduced with the theoretical concepts on the side of Fourier decomposition and the fundamental frequency of signals which we further apply in the notion of processing stock market signals.

Given any arbitrary discrete signal defined by the function $s(n)$, for $n \in \mathbb{N}$, as suggested in Bracewell (1978), we can describe the signal as:

$$s(n) = \frac{1}{N}\sum_{k=0}^{N} c_k e^{\frac{i2\pi kn}{N}}, \tag{3}$$

where $N \in \mathbb{N}$ denotes the total number of discrete points of the series $\{c_m\}_{m\in\{1,\dots,N\}}$ and each c_m represents a complex number, i.e. $c_m = a_m + b_m i$. Inside of each complex number c_m, the amplitude and phase of a respective sinusoidal component are encoded. We further denote the magnitude by $m(c_m) = |c_m| = \sqrt{a_m^2 + b_m^2}$. Therefore, by fitting the signal into Eq. (3) we get its decomposition into the series of complex numbers $\{c_m\}_{m\in\{1,\dots,N\}}$. Each of the complex numbers represents the information about each frequency presence in the signal. Thus, we describe the PO signal by its frequency decomposition. In particular, we extract the maximum magnitude, average magnitude, maximum phase, maximum amplitude and the average amplitude. The respective features, in particular, are described in the latter part of this chapter. To extract the frequency of maximum magnitude, we follow the quadratic interpolation of spectral peaks procedure[4], so we consider the parabolic equation in the form of

$$y(x) = a(x - p)^2 + b,$$

where b is the magnitude and p is the location of the peak point (our fundamental frequency). The whole process is graphically depicted in Fig. 2.

We propose the following frequency spectra feature extraction based on the previously introduced theoretical concepts. In Fig. 8, we provide daily close prices of three minute aggregates of the Apple stock from the 20th of February 2015, and we also plotted the moving average with period $n = 5$ curve with the red color. As a next step we compute the difference among these two lines (close price and moving average with period $n = 5$). From Fig. 3, which presents a sample PO induced on the market, we can assume that the PO series is a stationary signal. Moreover, by plotting the PO series as a histogram, we can support the evidence of the stationarity since it can be approximate by the skew normal distribution as seen in Fig. 4 or Fig. 6. Furthermore, in the Fig. 5 we provide the means of daily PO values from June 2014 till December 2014. In order to support the no-trend evidence, the same figure contains a linear fit (orange line), which is described by the equation $y = 0$ and thus supports the stationarity assumption.

[4] Quadratic Interpolation of Spectral Peaks are explained at https://ccrma.stanford.edu/jos/sasp/Quadratic_Interpolation_Spectral_Peaks.html, Accessed: 2019–10-06

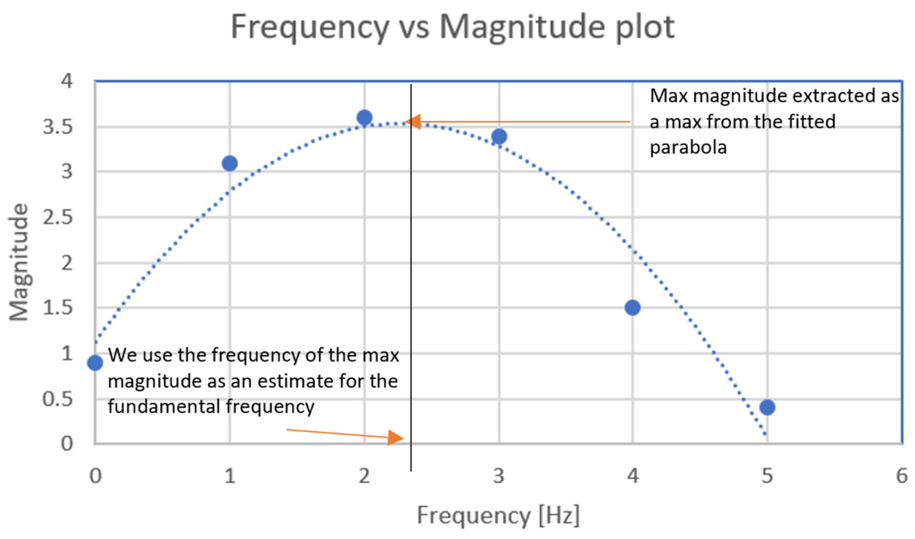

Fig. 2. Parabolic fit to identify the peak frequency and its magnitude (III, 2019).

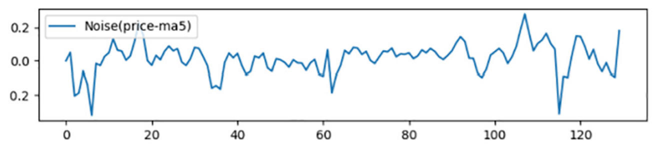

Fig. 3. The sample PO (denoted as "Noise") induced on the Apple stock three minute aggregates from the 20th of February 2015.

With aim to confirm an assumption that the PO is a stationary series we performed the statistical test, namely the Augmented Dickey-Fuller test. The obtained p-value= 0.0006468755129975428 is even less than 0.01 (confidence value 1%), so we can reject the null hypothesis (non-stationary data) with decent confidence.

Since the stationarity is confirmed, we proceed by extracting the Fourier transformation based features. We define an algorithm that has as input the finite set of functions $\mathscr{F} \subset \{f | f : \mathbb{C}^N \to \mathbb{R}^m\}$. Each function $f \in \mathscr{F}$ as an argument has an array of complex numbers denoted by $c = (c_1, c_2, \ldots, c_N) \in \mathbb{C}^N, N \in \mathbb{N}$, and returns an array of real-valued elements which extract some information from the provided complex array.

We employ the following set of functions $\mathscr{F}$:

1. *max magnitude extractor function* firstly fits the parabolic function to the values of the magnitudes computed from provided complex numbers and returns the "peak" point of the parabolic fit as an output. Given a single complex number c_i, we compute its magnitude by the following equation:

$$\sqrt{get\_amplitude(c_i)^2 + get\_phase(c_i)^2}.$$

2. *fundamental frequency extractorfunction* returns a fundamental frequency, i.e. the lowest frequency of a periodic waveform. We assume a phenomenon of Market Resonance when different resonants present in the market (e.g. every single participant can be perceived as a resonant as it executes trades with a certain specific frequency). And the maximum magnitude (which can be perceived as a significant market price jump) shall occur when the majority of these resonant frequencies intersect (i.e. many market participants trade in the same moment or period) - which happens at the multiple of the lowest frequency. We identify this frequency as a frequency of the maximum magnitude. Therefore we utilize the approach from a prior step when we extracted the maximum magnitude and simply compute the frequency of this maximum magnitude by employing an inverse function to the parabolic fit.
3. *phase of the biggest magnitude function* returns the phase of the $\text{argmax}_{c_i \in c}|c_i|$.
4. *average magnitude function* computes the average magnitude for the list of magnitudes obtained from every $c_i \in c$.

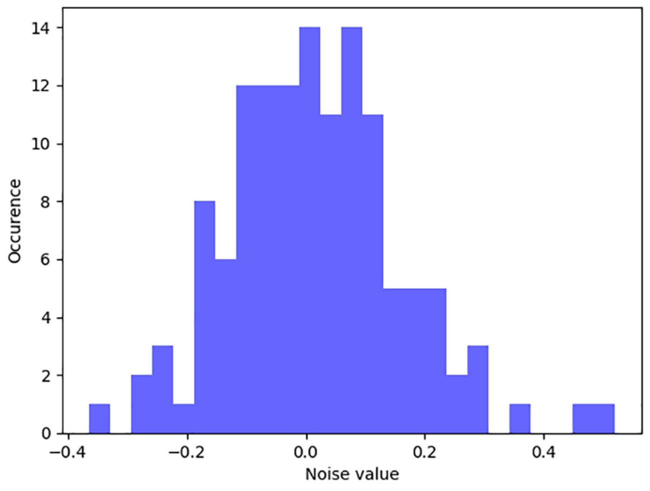

Fig. 4. Distribution of the price oscillator (denoted as "Noise") from the NASDAQ limit order book for AAPL shares (Mar. 2, 2015). 3.3.2015 with mean 0.0146 and std of 0.145.

We further denote the window size by $\mathscr{W}$. Let M denote the total time-frames within the single day, i.e. M is the length of the PO data series. Each function $f \in \mathscr{F}$ returns an $\mathbb{R}^m$-valued vector. The final array produced by the algorithm is of a dimensions $M \times |\mathscr{F}| \times m$, where $|\mathscr{F}|$ is the number of available functions.

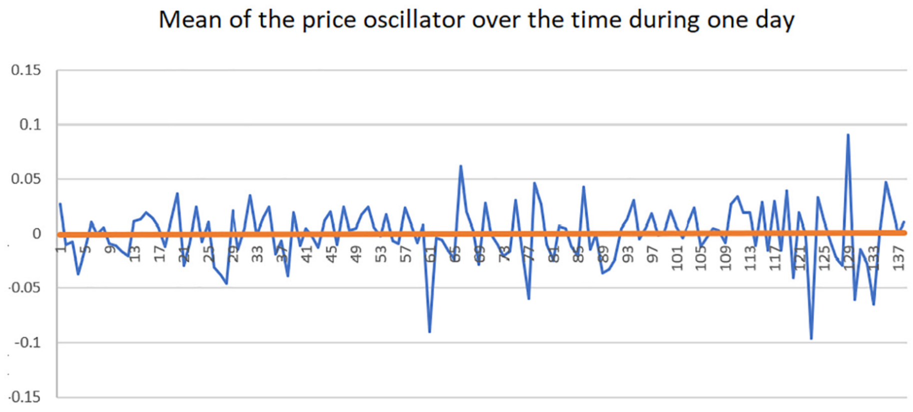

Fig. 5. Means of daily price oscillator signals from June 2014 to December 2014 of Apple stock with orange line depicting the linear fit.

Algorithm 1 The algorithm computing the Fourier-based features stored in the final matrix.

```
Result: 3D-array of dimensions $M \times |\mathcal{F}| \times m$
$\mathcal{F}$=getFunctions();
resultArray=new double[$M \times |\mathcal{F}| \times m$];
POData = getPOForDay(DAY);
for $i \leftarrow \mathcal{W}$ to $M$ do
    windowData = getWindowData(i, $\mathcal{W}$, POData);
    fourierTransformData = computeFourierTransfrom(windowData)
    foreach function $f_j \in \mathcal{F}$ do
        resultArray[i, j] = $f_j(fourierTransformData)$
    end
end
return resultArray
```

In the main part of Algorithm 1, i iterates starting from the initial window size $\mathscr{W}$ up to the length of total time frames. For each iteration i, we extract the $\mathscr{W}$ items which are indexed in the PO data series within the range of $i - \mathscr{W}$ to i. Then, we compute the Fourier spectra decomposition which gives the array of complex numbers representing information about respective frequencies. On top of this array, we apply some feature extraction methods, such as mean, standard deviation and min/max operations which are represented by the functions present in the set $\mathscr{F}$.

After extracting the result array, each of the extracted time-series is post processed by standard noise reduction techniques such as a moving average, which generate even further features. But how to check if such generated features hold any information that can help us in predicting profitable trades? We compute mutual information between the values of the feature F_i and the label. As per definition of mutual information (see Batina,

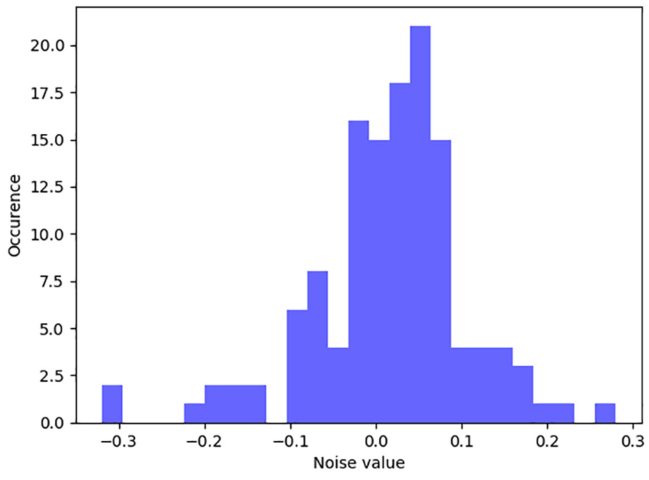

Fig. 6. Distribution of the PO from the NASDAQ limit order book for AAPL shares (Feb. 20, 2015), with mean 0.0159 and std of 0.0907.

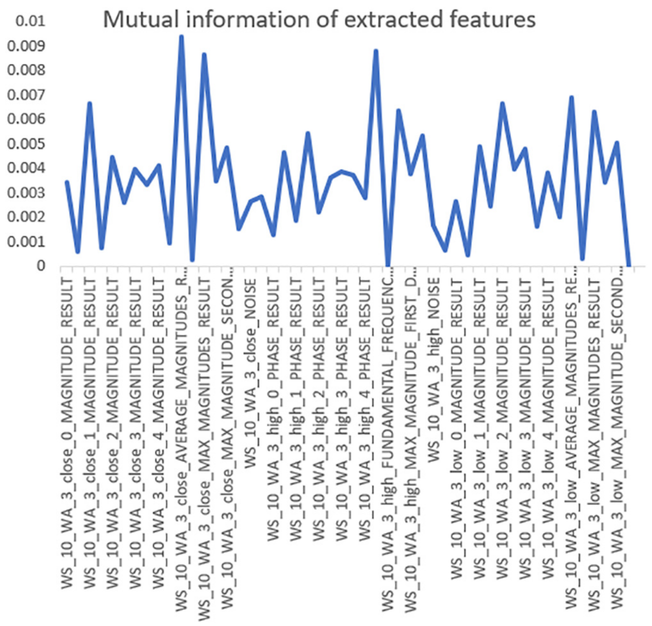

Fig. 7. Mutual information values of some of the generated features.

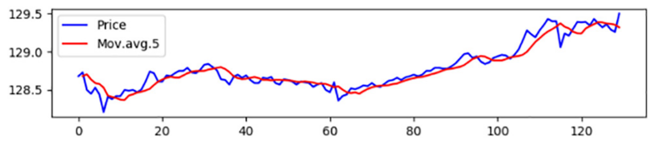

Fig. 8. A trading day of the Apple stock from the 20th of February 2015.

Table 2
F_1 mean score and F_1 standard deviation after running ten times the measurement in respective feature set and ranking method category (40 measurements overall).

Feature set	Ranking method	F_1 mean score	F_1 std
With new features	Mutual information	0.126441091	0.085182755
	Autocorrelation	0.070379998	0.064926468
Old features only	Mutual information	0.023346747	0.009447132
	Autocorrelation	0.01543692	0.005528859

Gierlichs, Prouff, Rivain, & Veyrat-charvillon (2011)) of two random variables $I(X, Y)$ is equal to 0 if and only if random variables X and Y are independent. Therefore, if the mutual information values for proposed features and target variable is bigger than 0, we can safely assume proposed features born some additional information about the target variable and therefore could help in a predictive ability of models. In Fig. 7, we provide mutual information values of some of Fourier transformation based features. As notable from the plot, some values are nearly 0, therefore nearly independent. However, there are also features that show some level of dependence. Further, the newly introduces features' impact on the performance of the GRU network is presented in Section 5.

5. Measurements and performance

In this section, we evaluate the informativeness of the newly suggested features by benchmarking the performance of the GRU model when fed with the Fourier transformation based features. As our whole set of features consists of 5996 features, we apply the SUS algorithm to select 200 features proportionally to a feature *SCORE* value. We cross-compare the strength of the following two feature quality metrics that determine *SCORE* values, namely:

1. the mutual information between the feature and the categorical variable of the close price,
2. the feature's absolute autocorrelation value.

Since we seek a balance between precision and recall to measure performance we compute the F_1 score (the harmonic mean of the precision and recall), see Patel, Shah, Thakkar, and Kotecha (2015).

We repeat measurements on top of each feature set (employing the randomized SUS selection) ten times and then apply statistics to evaluate the results.

5.1. Main hypothesis

To reiterate the main goal of the paper, we want to show that employing Fourier transform based features significantly improves the predictive capabilities of the GRU model. In order to do so, we run standard the *p-value* significance test, where we assume the following null hypothesis H_0:

”*Fourier transform based features do not improve the performance of the GRU model when measured by the F_1 score*”.

A summary of our measurements along with the observed mean and standard deviations of F_1 score can be found in Table 2. The first column of Table 2 distinguishes the feature set used (the Fourier transformation based features and the old-fashioned features), while in the second column the ranking method is given, i.e. the method used for generating the SCORE value. The third and fourth columns contain F_1 mean and F_1 standard deviation value in each of these groups of experiments. Furthermore, we compute respective *z-scores* for our measurements along with respective *p-values*, see the second and third column of Table 3. As can be noted in the fourth column of Table 3, our new features yield to significantly improved performance in our benchmark GRU model (for both *SCORE*-ing methods), with respect to the significance level of 0.01.

Therefore, we can safely reject our null hypothesis and affirm that proposed Fourier transform based features improved the model performance significantly.

5.2. Cross-comparing the proposed feature selection methods with the random choice of features

Furthermore, we wanted to investigate if the simple random feature selection is outperformed by the stochastic universal sampling with either mutual information or auto-correlation feature ranking methods. We assume the following null hypothesis $\widetilde{H_0}$:

”*Random selection performs as good as either of the proposed methods (mutual information or autocorrelation ranking accompanied with the SUS algorithm)*”.

We measured the mean value of F_1 score when using random feature selection instead of the SUS algorithm with a scoring method, and observed *z-score* and *p-values* can be found in Table 4. As can be noted from Table 4, we can reject the hypothesis $\widetilde{H_0}$ since random selection underperforms the SUS algorithm when used with the mutual-information at the significance level of 0.05.

However, we observed that the stochastic universal sampling when the score value is assigned with respect to the feature's auto-correlation, is performing better in the average, but not with a significant conclusion.

6. Conclusion

To conclude, the features utilizing Fourier Transformation, when extracted on top of different price of the preselected time interval (open price, close price, high price, low price and LOB level prices) offer statistically significant improvement of the GRU model's performance. The presented statistical analysis showed that our main assumption is confirmed with a significance level of 0.01. This finding shows that observing the stock market price as a signal and employing frequency decomposition techniques leads to strong results and brings a statistically significant boost into the decision making model.

We further observed that mutual-information, when combined with the SUS algorithm outperforms the random selection of features with a statistical significance of 0.05.

Regarding the autocorrelation based feature ranking, even though the average observed F_1 score has improved, this score ranking method has not shown significant results (for statistical significance $\alpha = 0.05$).

Table 3
Table containing *z-scores* and *p-values* for respective category with the significance information.

Ranking method	z-score	p-value	Significant ($\alpha = 0.01$)?
Mutual information	3.827217641	6.479996E−05	Yes
Autocorrelation	2.676031416	0.003724983	Yes

Table 4
Z-score and p-values of singificance test comparing random selection with SUS and mutual information or autocorrelation scoring.

Ranking method	z-score	p-value	Significant ($\alpha = 0.05$)?
Mutual information	2.165600285	0.015	Yes
Autocorrelation	0.110756942	0.456	No

In this paper, we showed how to extract additional features from the limit order book and the standard market price. As notable from our results studying the behavior of the divergence from the ”true” price, and also the behavior of the different limit order levels brings an additional significant informativeness.

We used the customized approach to extract the features from the LOB and we employ the Fourier transformation to extract indicators in the stock market, Lobster datasets in particular. Thus, there are no experiments, available in the literature, showing the performance of this type of indicators on the latest models.

Note that, we experimented with the Apple stock data in this work. However, in theory, the model shall scale-out well also to other stocks as their prices can be perceived as discrete signals. It would just show a different GRU model improvement. The biggest improvement shall be shown in markets where there is a strong periodicity of the participants. By the periodicity, we mean millisecond-wise latencies of bots, who follow an expected duration of the trades. So the more automatized the market participants are, the higher the likelihood of better model performance.

Furthermore, since exploring the influence of social networks on a stock market can be beneficial, a possible enchantment of the presented research is the incorporation of the social network data, e.g. twitter data.

To conclude, this work contributes to studying the Fourier transformation based features in the context of the stock market and their impact on the GRU model performance. In addition, we provide the evaluation of results that clearly show potential in exploring this type of features even further.

Future work will be continued in several directions. We will try to incorporate the wavelet transform, which could be very informative, in order to extract new features. Moreover, we want to investigate the performance of different models that can be used instead of the GRU model, e.g. the RNN and LSTM, and compare the results of all three.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors are thankful to Professor Thorsten Rheinländer for his help in doing the LOBSTER data analysis and valuable and insightful suggestions. This research was supported by TU Wien.

Table 5
Acronyms mentioned in the paper.

ADF	Augmented Dickey-Fuller
AAPL	Apple company ticker symbol
CNN	convolutional neural network
DLNN	Deep Learning Neural Network
GRU	Gated Recurrent Unit
HFT	High-Frequency Trading
KPI	key performance indicator
LOB	Limit Order Book
LOBSTER	Limit Order Book System Efficient Reconstructor
LSTM	Long Short Term Memory
MSFT	Microsoft company ticker symbol
NASDAQ	National Association of Securities Dealers Automated Quotations
PO	Price Oscillator
RNN	Recurrent neural network
RRR	risk-reward ratio
SUS	Stochastic Universal Sampling

Appendix A. Appendix

A.1. Algorithm

Algorithm 2 The sweep forward algorithm which add label to each data vector.

```
Input: $\mathscr{F}_{180}, RISK, REWARD$
Output: the labelled data set $\mathscr{F}^*_{180}$
1:  $\mathscr{F}^*_{180} = None$
2:  for $i = 0$ to $|\mathscr{F}_{180}|$ do
3:    $inceptivePrice = extractPrice(\mathscr{F}_{180}[i])$
4:    $datavector = \mathscr{F}_{180}[i]$
5:    $stopLoss = inceptivePrice \cdot (1 - RISK)$
6:    $targetReward = inceptivePrice \cdot (1 + REWARD)$
7:    $label = False$
8:    for $forwardIndex = i + 1$ to $|\mathscr{F}_{180}|$ do
9:      $currentPrice = extractPrice(\mathscr{F}_{180}[forwardIndex])$
10:     $temporaryStopLoss = currentPrice \cdot (1 - RISK)$
11:     if $(temporaryStopLoss > stopLoss)$ then
12:       $stopLoss = tempStopLoss$
13:     end if
14:     if $(currentPrice < stopLoss)$ then
15:       $\mathscr{F}^*_{180}.append(labelPointAsIdle(datavector))$
16:       $label = True$
17:       break
18:     end if
19:     if $(currentPrice > targetReward)$ then
20:       $\mathscr{F}^*_{180}.append(labelPointAsBuy(datavector))$
21:       $label = True$
22:       break
23:     end if
24:   end for
25:   if $(label == False)$ then
26:     $\mathscr{F}^*_{180}.append(labelPointAsIdlePoint(datavector))$
27:   end if
28: end for
29: return $\mathscr{F}^*_{180}$
```

References

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In Proceedings of the second international conference on genetic algorithms (pp. 14–21). volume 206.

Batina, L., Gierlichs, B., Prouff, E., Rivain, M., & Veyrat-charvillon, N. (2011). Mutual information analysis: A comprehensive study. *Journal of Cryptology, 24*, 269–291.

Bracewell, R. (1978). *The Fourier transform and its applications* (2nd ed.). Tokyo: McGraw-Hill Kogakusha Ltd.

Brzeszczyński, J., & Ibrahim, B. M. (2019). A stock market trading system based on foreign and domestic information. Expert Systems with Applications, 118, 381–399. https://doi.org/10.1016/j.eswa.2018.08.005. URL: http://www.sciencedirect.com/science/article/pii/S095741741830513X.

Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. *Quantitative Finance, 19*, 1271–1291. https://doi.org/10.1080/14697688.2019.1571683.

le Calvez, A., & Cliff, D. (2018). Deep learning can replicate adaptive traders in a limit-order-book financial market. In 2018 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1876–1883). IEEE.

Chatzis, S. P., Siakoulis, V., Petropoulos, A., Stavroulakis, E., & Vlachogiannakis, N. (2018). Forecasting stock market crisis events using deep and statistical machine learning techniques. Expert Systems with Applications, 112, 353–371. https://doi.org/10.1016/j.eswa.2018.06.032. URL: http://www.sciencedirect.com/science/article/pii/S0957417418303798.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Chow, V. (2020). Predicting auction price of vehicle license plate with deep recurrent neural network. Expert Systems with Applications, 142, 113008. https://doi.org/10.1016/j.eswa.2019.113008. URL: http://www.sciencedirect.com/science/article/pii/S0957417419307250.

Colby, R. W., & Meyers, T. A. (1988). *The encyclopedia of technical market indicators*. IL: Dow Jones-Irwin Homewood.

Cont, R., Kukanov, A., & Stoikov, S. (2014). The price impact of order book events. *Journal of Financial Econometrics, 12*, 47–88.

Das, S. R., Mishra, D., & Rout, M. (2019). Stock market prediction using firefly algorithm with evolutionary framework optimized feature reduction for oselm method. Expert Systems with Applications: X, 4, 100016. https://doi.org/10.1016/j.eswax.2019.100016. URL: http://www.sciencedirect.com/science/article/pii/S2590188519300162.

Dixon, M. (2018a). A high-frequency trade execution model for supervised learning. *High Frequency, 1*, 32–52.

Dixon, M. (2018). Sequence classification of the limit order book using recurrent neural networks. Journal of Computational Science, 24, 277–286. https://doi.org/10.1016/j.jocs.2017.08.018. URL: http://www.sciencedirect.com/science/article/pii/S1877750317309675.

Dixon, M. F., Polson, N. G., & Sokolov, V. O. (2017). Deep learning for spatio-temporal modeling: Dynamic traffic flows and high frequency trading. arXiv preprint arXiv:1705.09851.

Eberlein, E., Glau, K., & Papapantoleon, A. (2010). Analysis of fourier transform valuation formulas and applications. *Applied Mathematical Finance, 17*, 211–240.

Farah, M. B., Guesmi, R., Kachouri, A., & Samet, M. (2020). A novel chaos based optical image encryption using fractional fourier transform and dna sequence operation. Optics & Laser Technology, 121, 105777. https://doi.org/10.1016/j.optlastec.2019.105777. URL: http://www.sciencedirect.com/science/article/pii/S0030399219306899.

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

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*, 1735–1780.

Hoseinzade, E., & Haratizadeh, S. (2019). Cnnpred: Cnn-based stock market prediction using a diverse set of variables. Expert Systems with Applications, 129, 273–285. https://doi.org/10.1016/j.eswa.2019.03.029. URL: http://www.sciencedirect.com/science/article/pii/S0957417419301915.

Huang, R., & Polak, T. (2011). Lobster: Limit order book reconstruction system. Available at SSRN 1977207.

III, J. O. S.. Quadratic interpolation of spectral peaks.https://ccrma.stanford.edu/jos/sasp/Quadratic_Interpolation_Spectral_Peaks.html. Accessed: 2019-10-06.

Kithulgoda, C. I., Pears, R., & Naeem, M. A. (2018). The incremental fourier classifier: Leveraging the discrete fourier transform for classifying high speed data streams. Expert Systems with Applications, 97, 1–17. https://doi.org/10.1016/j.eswa.2017.12.023. URL: http://www.sciencedirect.com/science/article/pii/S095741741730845X.

Krishna, K., & Murty, M. N. (1999). Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics Part B (Cybernetics), 29*, 433–439.

Lee, T. K., Cho, J. H., Kwon, D. S., & Sohn, S. Y. (2019). Global stock market investment strategies based on financial network indicators using machine learning techniques. Expert Systems with Applications, 117, 228–242. https://doi.org/10.1016/j.eswa.2018.09.005. URL: http://www.sciencedirect.com/science/article/pii/S0957417418305761.

Li, Y., Jiang, W., Yang, L., & Wu, T. (2018). On neural networks and learning systems for business computing. Neurocomputing, 275, 1150–1159. https://doi.org/10.1016/j.neucom.2017.09.054. URL: http://www.sciencedirect.com/science/article/pii/S0925231217315734.

Li, Z., Wang, J., Xing, H., Jin, K., & Huang, H. (2020). Determining dendrite arm spacing in directional solidification using a fast fourier transform method. Computational Materials Science, 173, 109463. https://doi.org/10.1016/j.commatsci.2019.109463. URL: http://www.sciencedirect.com/science/article/pii/S0927025619307621.

Liu, Y. (2019). Novel volatility forecasting using deep learning–long short term memory recurrent neural networks. Expert Systems with Applications, 132, 99–109. https://doi.org/10.1016/j.eswa.2019.04.038. URL: http://www.sciencedirect.com/science/article/pii/S0957417419302635.

Menkhoff, L. (2010). The use of technical analysis by fund managers: International evidence. *Journal of Banking & Finance, 34*, 2573–2586.

Murphy, J. J. (1999). *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin.

Naranjo, R., & Santos, M. (2019). A fuzzy decision system for money investment in stock markets based on fuzzy candlesticks pattern recognition. Expert Systems with Applications, 133, 34–48. https://doi.org/10.1016/j.eswa.2019.05.012. URL:http://www.sciencedirect.com/science/article/pii/S0957417419303318.

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

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*, 259–268.

Radojičić, D., Kredatus, S., & Rheinländer, T. (2018). An approach to reconstruction of data set via supervised and unsupervised learning. In *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)* (pp. 000053–000058). IEEE.

Sirignano, J., & Cont, R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance, 19*, 1449–1459.

Tang, D., Rong, W., Qin, S., Yang, J., & Xiong, Z. (2020). A n-gated recurrent unit with review for answer selection. Neurocomputing, 371, 158–165. https://doi.org/10.1016/j.neucom.2019.09.007. URL: http://www.sciencedirect.com/science/article/pii/S0925231219312676.

Xu, Q., Wang, L., Jiang, C., & Zhang, X. (2019). A novel umidas-svqr model with mixed frequency investor sentiment for predicting stock market volatility. Expert Systems with Applications, 132, 12–27. https://doi.org/10.1016/j.eswa.2019.04.066. URL: http://www.sciencedirect.com/science/article/pii/S0957417419303021.

Zheng, B., Moulines, E., & Abergel, F. (2012). Price jump prediction in limit order book. arXiv preprint arXiv:1204.1381.

Zheng, Z., Gao, Y., Yin, L., & Rabarison, M. K. (2019). Modeling and analysis of a stock-based collaborative filtering algorithm for the chinese stock market. Expert Systems with Applications, (p. 113006). https://doi.org/10.1016/j.eswa.2019.113006. URL: http://www.sciencedirect.com/science/article/pii/S0957417419307237.

Zhylyevskyy, O. (2010). A fast fourier transform technique for pricing american options under stochastic volatility. *Review of Derivatives Research, 13*, 1–24.