Information Sciences 553 (2021) 305–330

Contents lists available at ScienceDirect

Information Sciences

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

Attention enhanced long short-term memory network with multi-source heterogeneous information fusion: An application to BGI Genomics

Qun Zhang [a,b], Lijun Yang [c], Feng Zhou [d,*]

[a] School of Finance, Guangdong University of Foreign Studies, Guangzhou, Guangdong 510006, China
[b] Southern China Institute of Fortune Management Research, Guangdong University of Foreign Studies, Guangzhou, Guangdong 510006, China
[c] School of Mathematics and Statistics, Henan University, Henan 475004, China
[d] School of Information, Guangdong University of Finance and Economics, Guangzhou 510320, China

ARTICLE INFO

Article history:
Received 2 July 2020
Received in revised form 10 October 2020
Accepted 12 October 2020
Available online 21 October 2020

Keywords:
Attention mechanism
Long short-term memory network
Stock price prediction
Heterogeneous information fusion
Machine learning

ABSTRACT

The recent availability of enormous amounts of both data and computing power has created new opportunities for predictive modeling. This paper compiles an analytical framework based on multiple sources of data including daily trading data, online news, derivative technical indicators, and time–frequency features decomposed from closing prices. We also provide a real-life demonstration of how to combine and capitalize on all available information to predict the stock price of BGI Genomics. Moreover, we apply a long short-term memory (LSTM) network equipped with an attention mechanism to identify long-term temporal dependencies and adaptively highlight key features. We further examine the learning capabilities of the network for specific tasks, including forecasting the next day's price direction and closing price and developing trading strategies, comparing its statistical accuracy and trading performance with those of methods based on logistic regression, support vector machine, gradient boosting decision trees, and the original LSTM model. The experimental results for BGI Genomics demonstrate that the attention enhanced LSTM model remarkably improves prediction performance through multi-source heterogeneous information fusion, highlighting the significance of online news and time–frequency features, as well as exemplifying and validating our proposed framework.

© 2020 Elsevier Inc. All rights reserved.

1. Introduction

As has been widely reported in the media, the outbreak of coronavirus disease 2019 (COVID-19) has caused global mass death and panic, and has become a worldwide public health emergency. Stock markets have also suffered a shock, especially for stocks in biotechnology and pharmaceutical industries, not only because these stocks are usually news dependent [11] but also because these industries have received growing attention from investors in recent years due to their potential relevance in diagnosing, mitigating, treating, and preventing diseases. Meanwhile, with recent technological advancements that foster vibrant creation, sharing, and collaboration among web users, the speed of information dissemination has been greatly improved. In the investment field, although the enormous amounts of information being generated show great

* Corresponding author.
E-mail address: fengzhou@gdufe.edu.cn (F. Zhou).

https://doi.org/10.1016/j.ins.2020.10.023
0020-0255/© 2020 Elsevier Inc. All rights reserved.

promise to reduce trading costs attributable to information asymmetry and financial market uncertainty, the increased quantities of data, stored in structured, semi-structured, and unstructured formats and generated from multiple sources, require further interpretation. How to develop a real-time stock prediction framework that capitalizes on all available information from multiple data sources remains an ongoing research topic. The idea of the granular models introduced by Pedrycz [30] illustrates that generalizations of numerical models are formed as a result of an optimal allocation of information granularity. Specifically, information fusion for stock price prediction is a multidisciplinary research field involving integration of information from multiple sources for data mining (subsuming statistics and machine learning), signal processing, text mining, knowledge discovery, and expert systems modeling [17,32]. However, using multiple data sources instead of a single source is a considerable challenge because solving this problem requires not only improving the efficiency of information fusion, but also dealing with high levels of uncertainty, complexity [24], nonlinearity [33], and the dynamism of the market itself. Leveraging a unique dataset collected from multiple sources and performing in-depth analyses on it, we provide a real-life demonstration of how these issues can be addressed.

BGI Genomics (stock code: 300676.SZ),[1] a part of BGI Group which is one of the world's leading life science and genomics organizations, was officially listed on the Shenzhen Stock Exchange on July 14, 2017, becoming the exchange's 2,001st listed company. Its strengths are prenatal screening, hereditary cancer screening, detection of rare diseases, and aiding precision medicine research. It is one of the world's leading providers of commercial sequencing services and genomics tests for medical institutions, research institutions, enterprises, and other public and private partners. The company's potential, in light of the current coronavirus pandemic and uncertain commercial environment, makes it an interesting candidate for in-depth exploration of efficient approaches to data analysis.

At present, data are becoming one of the most valuable resources. In general, data coming from more than one source can deliver more information or knowledge than data from a single source. Historically, numerical and textual data have been the two main types of data utilized by the financial field. Analysis of univariate or multivariate time series, whose values can be numbers, texts, or other types of data, can provide insights into the underlying data generating process. On the one hand, most common econometric models for forecasting treat each new signal as a noisy linear combination of the last few signals and independent noise terms, including the autoregressive model, the moving average model, the autoregressive moving average model, the autoregressive integrated moving average model, and stochastic volatility model. [13]. Although these models have advantages in theoretically describing the underlying data generating process based on statistical logic, they contain some strong assumptions about the noise terms (such as that variables are independent and identically distributed, or follow a *t*-distribution) or explanatory variables (such as stationarity or exogeneity) that are not fully satisfied in the real world. On the other hand, many machine learning models have been successfully developed over recent decades, without imposing restrictive assumptions, to learn from and forecast financial time series, such as the support vector machine (SVM) [34], gradient boosted decision trees (GBDTs) [47], neural networks (NNs) [1,45], and the ensemble model by cascading logistic regression (LR) onto GBDT [47]. However, one key limitation of most existing machine learning models is their lack of an explicitly declared mechanism to handle nonlinearity and non-stationarity in time series [48], which may lead to inaccurate predictions. From a signal processing perspective, time–frequency analysis methods [12,16] can be employed for extracting and utilizing inherent instantaneous amplitude and frequency/phase information in combination with other relevant morphological features. Additionally, many researchers have verified in their studies that the feature selection process is a key factor for precise predictions, especially when data types are mixed and resultant features are together fed into a classifier or regressor [28].

Meanwhile, it is challenging but also rewarding to interpret textual data and extract discriminative features from it effectively. For example, firm-specific news articles can spread information and enrich the knowledge of investors. They can consciously or unconsciously further affect investors' trading activities, which might lead to overreaction or underreaction of the stock price to the information [26]. The examination of monthly stock returns following public news also suggests that bad public news is always followed by a negative drift but less drift is observed for stocks following good news [6]. Experiments conducted for predicting the stock prices of Amazon and eBay in a framework of a multivariate Bayesian structural time series model embedded with online text mining reveal that incorporating information from financial news and Twitter feeds into sentiment predictors consistently boosts their forecasting power [19]. Therefore, for available textual data, more advanced intelligent techniques such as computational linguistics and natural language processing (NLP) techniques can be performed to structure input text, derive informative features, identify text sentiment, evaluate the output, or explore stock trading strategies in an automated framework [23].

Building on the existing formal approaches, another rewarding direction for research would be to take advantage of deep neural networks to capture temporal dependence structures in data across both short- and long-term periods, even equipped with an attention mechanism that has the ability to focus on the most relevant parts of their inputs. Artificial neural networks (ANNs) are currently revolutionizing many technological areas, and aid in addressing the difficult aspects of theoretically solvable but computationally hard problems [19,40]. ANNs are viable candidates to capture nonlinear relationships in input data without assumptions or the need for prior knowledge of the statistical distributions of the data [2]. Their capabilities in stock market analysis and prediction in emerging markets are found to be more attractive than the capabilities of Fama and French's model [5]. The recurrent neural network (RNN) is a special kind of feed-forward neural network that

[1] BGI Genomics website in English: https://en.genomics.cn/.

learns sequential patterns through internal loops by receiving input sequences [39]. The long short-term memory (LSTM) network, an RNN composed of long short-term memory blocks that is also capable of learning long-term dependencies, was first proposed by Hochreiter and Schmidhuber [15]. Many LSTM networks have been successfully implemented for sequential data modeling; for instance, an LSTM network employed to predict returns in the Chinese stock market demonstrates better performance than a random prediction method [7]. Trading strategy based on volume-weighted average prices of daily S&P 500 data from 1992 to 2015 and derived using an LSTM network also outperforms memory-free classification methods, that is, logistic regression, random forest (RF), and a deep neural network [10]. Further, a kind of dual-stage attention-based RNN for stock price prediction has been proposed by Qin et al. [35], in which the attention mechanism is incorporated into an encoder-decoder framework.

In summary, the investigation into multi-source heterogeneous information fusion and its applications in stock price prediction is still at a developmental stage. A promising direction for research is to combine the attention mechanism with the LSTM model to extract key features from various data sources, and to investigate their joint impact on the performance of stock price prediction models and trading strategy. For this purpose, we use the example of BGI Genomics. Overall, this paper contributes to the growing literature in several significant ways. (1) We compile a set of features based on multiple sources of data incorporating daily trading data, online news, technical indicators derived from trading data, and time–frequency features decomposed from closing prices, so as to provide a best performing-feature subset for information fusion and prediction. (2) In order to effectively weaken the influence of non-stationarity of data on forecasting performance, we address the problem of decomposing the original non-stationary price time series into a group of time–frequency features. (3) We adapt an attention enhanced LSTM network and verify its forecasting performance using BGI Genomics as a real-life demonstration, mainly by comparing the model's results with those of alternative models and comprehensively analyzing which model demonstrates superior performance and generalization ability. (4) A framework integrating various data preprocessing techniques is proposed for forecasting the next day's price direction and the next day's closing price, and analyzing the benefits of a long/short trading strategy. This is exemplified and validated through in-depth analyses on BGI Genomics.

The remainder of this paper is structured as follows. Section 2 details the intrinsic time-scale decomposition method, the long short-term memory network, and the attention mechanism. Section 3 proposes the study's framework and illustrates how it works. Section 4 presents the experimental results and analysis for BGI Genomics. Finally, Section 5 summarizes our findings and concludes the paper.

2. ITD, LSTM, and attention mechanism

2.1. Intrinsic time-scale decomposition (ITD)

As an adaptive non-stationary signal decomposition technique, intrinsic time-scale decomposition (ITD) has been successfully applied in the field of signal processing [36,44,46]. Through the ITD process, the original non-stationary signal can be adaptively decomposed into several proper rotation components (PRCs), whose frequencies range from high to low. In this subsection, we briefly review the ITD process. For a more detailed description of the method, please refer to Frei and Osorio [12].

For a given signal $x(t)$, let $\mathcal{B}$ and $\mathcal{L}$ be the baseline extracting operator and the PRC extracting operator, respectively. In the first step of ITD, $x(t)$ is decomposed into two components through:

$$x(t) = \mathcal{B}x(t) + \mathcal{L}x(t) = b(t) + l(t), \tag{1}$$

where $b(t)$ is a baseline and $l(t)$ is a PRC. Then, the process is repeated by using the baseline signal as a new input signal until the resulting baseline has only two extreme values or is a constant. In the end, the input signal $x(t)$ can be decomposed into a series of PRCs with a decreasing instantaneous frequency. If this process takes k steps, $x(t)$ is broken down into:

$$b^0(t) := x(t) = b^k(t) + \sum_{j=1}^{k} l^j(t), \tag{2}$$

and the baselines and PRCs satisfy:

$$b^j(t) = b^{j+1}(t) + l^{j+1}(t), j = 0, 1, \ldots, k-1. \tag{3}$$

Let $\gamma_s^j, s = 1, 2, \ldots, S$ be the extrema points of $b^j(t), \gamma_0^j = 0$is its initial point. If there are multiple consecutive data points with the same extreme value, we take γ_s^j to be the rightmost point of these extreme values. Furthermore, we define $b_s^j := b^j\left(\gamma_s^j\right)$. Then, constructing a baseline $b^{j+1}(t)$ by a piecewise linear formula in the interval $t \in \left(\gamma_s^j, \gamma_{s+1}^j\right)$ between successive extrema, that is,

$$b^{j+1}(t) = b_s^{j+1} + \frac{b_{s+1}^{j+1} - b_s^{j+1}}{b_{s+1}^j - b_s^j} \times \left[b^j(t) - b_s^j\right], \tag{4}$$

where the knots are defined as

$$b_{s+1}^{j+1} := b^{j+1}\left(\gamma_{s+1}^{j+1}\right) = \alpha \times \left[b_s^j + \frac{\gamma_{s+1}^j - \gamma_s^j}{\gamma_{s+2}^j - \gamma_s^j}\left(b_{s+2}^j - b_s^j\right)\right] + (1-\alpha) \times b_{s+1}^j, \tag{5}$$

where $\alpha \in (0,1)$ is a tunable parameter, and $\alpha = \frac{1}{2}$ in general.

After using the ITD method to decompose the input signal into a set of PRCs and a residual, the next step is to consider the instantaneous amplitude $A(t)$, phase $\theta(t)$, and frequency $Fr(t)$ of each PRC $l(t)$. Instead of the Hilbert-Huang transformation (HHT) [16], Frei and Osorio [12] propose a wave-based method to calculate the instantaneous phase and instantaneous amplitude of PRCs, ensuring a monotonic increase in phase angle. The instantaneous phase (IP) can be calculated as follows:

$$\theta(t) = \begin{cases} \left(\frac{l(t)}{A_1}\right)\frac{\pi}{2}, & t \in [t_1, t_2), \\ \left(\frac{l(t)}{A_1}\right)\frac{\pi}{2} + \left(1 - \frac{l(t)}{A_1}\right)\pi, & t \in [t_2, t_3), \\ \left(-\frac{l(t)}{A_2}\right)\frac{3\pi}{2} + \left(1 + \frac{l(t)}{A_2}\right)\pi, & t \in [t_3, t_4), \\ \left(-\frac{l(t)}{A_2}\right)\frac{3\pi}{2} + \left(1 + \frac{l(t)}{A_2}\right)2\pi, & t \in [t_4, t_5), \end{cases} \tag{6}$$

where t_1 and t_5 are the corresponding times of two successive zero up-crossing points, $t_3 \in [t_1, t_5]$ is the time of the zero down-crossing point, $t_2 \in [t_1, t_3)$ is the time of the maximum point and $t_4 \in [t_3, t_5)$ is the time of the minimum point. A_1 is the value of $A(t)$ at t_2 (i.e., the maximum on the positive half-wave) and $-A_2$ is the value of $A(t)$ at t_4 (i.e., the minimum on the negative half-wave). Then, the instantaneous amplitude (IA) is defined as follows:

$$A(t) = \begin{cases} A_1, & t \in [t_1, t_3), \\ A_2, & t \in [t_3, t_5). \end{cases} \tag{7}$$

Obviously, $A(t)$ is a piecewise function, which is determined by the extreme value of PRCs. According to the IP formula in Eq. (6), the instantaneous frequency (IF) can be calculated by

$$Fr(t) = \frac{1}{2\pi} \times \frac{d\theta(t)}{dt}. \tag{8}$$

Overall, the ITD method is adaptive and suitable for processing stock trading data, which are usually nonlinear and non-stationary time series. The panels in the first column of Fig. 1 illustrate the decomposition results produced by the ITD method, including the PRCs (denoted $c1 \sim c5$) and the residual (denoted $c6$) of the closing price of BGI Genomics from July 14, 2017 to July 21, 2020. The second and the last columns of Fig. 1 present the IAs (denoted $a1 \sim a5$) and IFs (denoted $p1 \sim p5$) of the PRCs, respectively.

2.2. Long short-term memory (LSTM)

An RNN composed of several long short-term memory (LSTM) blocks is commonly called an LSTM network. In each memory block, there exists a memory cell that stores the state. Strictly speaking, the main difference between the blocks of an LSTM and a traditional RNN is that the former can use newly introduced gates to decide whether to keep the existing memory or forget unnecessary information so as to ensure that the gradient of the long-term dependencies cannot vanish, whereas the latter overwrites its content at each time step. The structure of LSTM as described in previous studies [7,10,15] is illustrated in Fig. 2.

Given the current input X_t, the state H_{t-1} that the previous step generated, and the memory state of the cell C_{t-1} (peephole), the LSTM cell can be synoptically expressed as:

$$Y_t^{LSTM}, H_t, C_t = LSTM(X_t, H_{t-1}, C_{t-1}). \tag{9}$$

Specifically, the detailed formulae of the *LSTM* function for the decisions whether to forget the stored memory, to take the inputs, and to output the state generated are given as follows:

$$C_t = F_t \odot C_{t-1} + I_t \odot \tanh(W_{xc}X_t + W_{hc}H_{t-1} + b_c), \tag{10}$$

$$H_t = O_t \odot \tanh(C_t), \tag{11}$$

$$Y_t^{LSTM} = \sigma(W_{hy}H_t + b_y), \tag{12}$$

where F_t, I_t, and O_t are the forget gate, input gate, and output gate at time t, respectively. The forget gate is represented by $F_t = \sigma(W_{xf}X_t + W_{hf}H_{t-1} + b_f)$, where W_{xf} and W_{hf} are the corresponding weight matrices and b_f is the bias. The input gate is represented by $I_t = \sigma(W_{xi}X_t + W_{hi}H_{t-1} + b_i)$ with the corresponding weight matrices W_{xi} and W_{hi} and the bias b_i. The output gate is represented by $O_t = \sigma(W_{xo}X_t + W_{ho}H_{t-1} + b_o)$ where W_{xo} and W_{ho} are the corresponding weight matrices and b_o is the bias. C_t and C_{t-1} are the current and prior states of the cell, respectively. W_{xc} and W_{hc} represent the corresponding weight

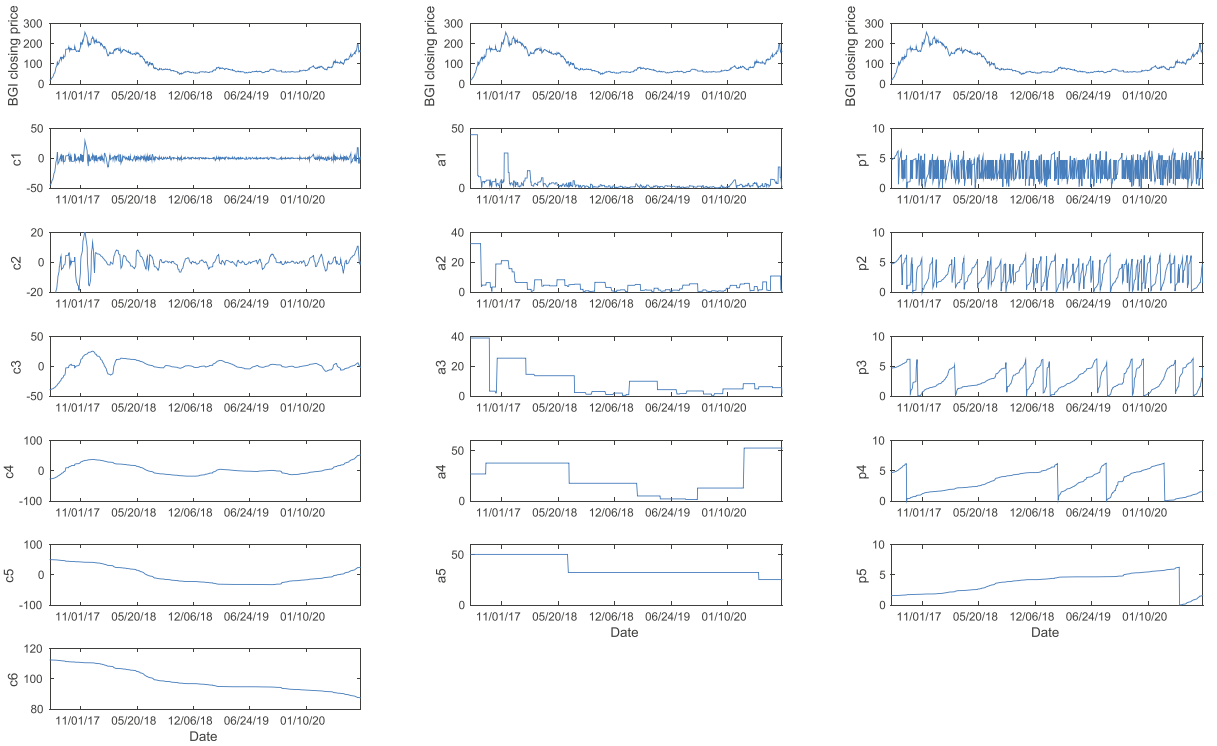


Fig. 1. The results decomposed from the closing price of BGI Genomics by the ITD method from July 14, 2017 to July 21, 2020. Panels in the first column: The top panel is the original signal (i.e., closing price time series), the remaining panels are the PRCs ($c1 \sim c5$) and the residual ($c6$). Panels in the second column: The top panel is the original signal (i.e., closing price time series), and the remaining five panels are the corresponding IAs of the PRCs ($a1 \sim a5$). Panels in the last column: The top panel is the original signal (i.e., closing price time series), and the remaining five panels are the corresponding IFs of the PRCs ($p1 \sim p5$).

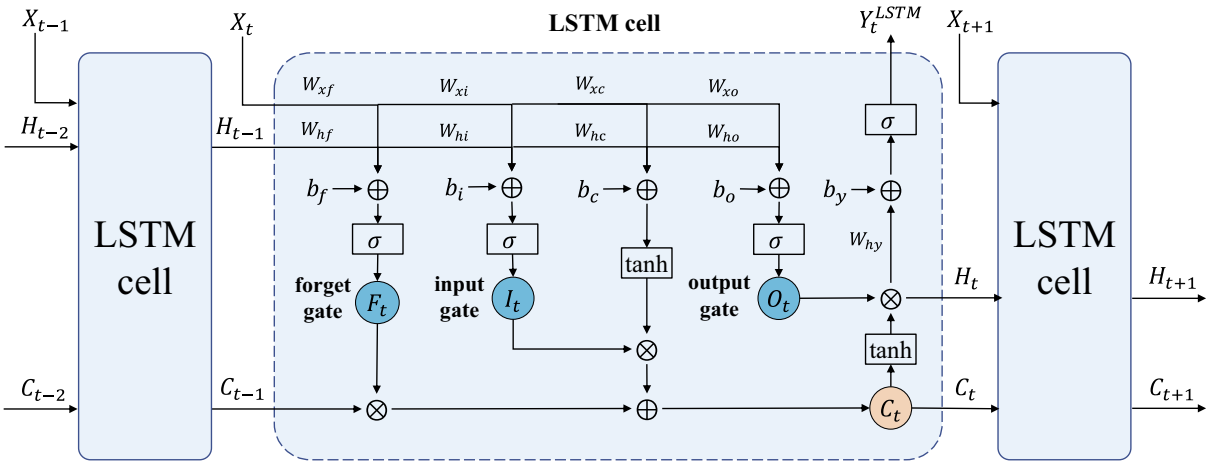


Fig. 2. Graphical illustration of the structure of LSTM.

matrices, and b_c is the bias. $\odot$ represents an element-wise multiplication operator, $\tanh(\cdot)$ is a hyperbolic tangent function, and $\sigma(\cdot)$ is a sigmoid activation function.

According to Eq. (10), forget gate F_t controls the amount of information in the past cell state C_{t-1} in updating the cell state at time t, and the input gate I_t determines how much new information is stored in the cell state C_t. Finally, the hidden state H_t in Eq. (11) is determined by passing through the output gate O_t and filtering at C_t, and C_t passes into the hyperbolic tangent function. The prediction represented by Y_t^{LSTM} in Eq. (12) is determined by H_t where W_{hy} is the weight matrix and b_y is the bias.

Overall, the LSTM network can handle not only the large dimensionality of the system, but also a very general functional form of the states while allowing for lags of unknown and potentially long duration in the time series, which makes it very suitable for capturing long-term dependencies. It serves the purpose of finding hidden states in the time series, summarizing them in a small number of state processes, and applying the most appropriate transformation to the non-stationary time series in a data-driven way.

2.3. Attention mechanism

Attention has been proven to be a powerful mechanism for embedding categorical inference in a deep neural network. Its main concept is to choose "where to look" by assigning a weight or importance to each lower position when computing an upper level representation [18,27]. An overview of the architecture of attention enhanced LSTM is provided in Fig. 3.

Let $\{H_t\}_{t=1}^{N}$ be the hidden states obtained from the LSTM layer, where N is the number of data points. All these states are fed into a subsequent Attention layer, and the output of the Attention layer can be synoptically regarded as

$$Y^{Attention} = Attention(H_1, H_2, \ldots, H_t). \tag{13}$$

Specifically, the *Attention* function is formed by a weighted sum of all the hidden vectors, calculated as

$$Y^{Attention} = \sum_t \alpha_t H_t, \tag{14}$$

where the alignment vector $\alpha_t \in [0, 1]$ is defined as

$$\alpha_t = softmax(u_t) = \frac{exp(u_t)}{\sum_t exp(u_t)}, \tag{15}$$

and

$$u_t = \eta^T \tanh(W_{attn} H_t + b_{attn}), \tag{16}$$

where α_t denotes attention weights satisfying the constraint of $\sum_t \alpha_t = 1$; ,$softmax(\cdot)$ is a softmax function; η is a trained parameter vector and η^T is its transpose; W_{attn} represents the learnable matrix and b_{attn} is the bias.

3. Our proposed framework

3.1. Overview

We illustrate the architecture of our proposed framework in Fig. 4, which can be synoptically divided into four stages. In the first stage, we collect data related to BGI Genomics from multiple sources, consisting of daily trading data, online news, derivative technical indicators, and time–frequency features decomposed from the closing price. Because the collected data include both numerical (e.g., trading data) and textual data (e.g., online news), they pose a challenge for our prediction purposes. The second stage is feature engineering. This stage mainly involves the implementation of data cleaning, feature encoding, dimension reduction, and normalization. In the third stage, the proposed prediction model (i.e., attention enhanced LSTM, denoted LSTM-Attention for simplicity), is trained on the training dataset for various prediction tasks, including forecasting the next day's price direction and the next day's closing price. In the last stage, hyper-parameters that appear in the prediction model are selected according to their performance on the validation dataset. In addition, prediction performance is evaluated on the testing dataset at this stage.

3.2. Multi-source heterogeneous data collection

This study considers a dataset related to BGI Genomics for the period from 14 July 14, 2017 to July 21, 2020, which encompasses the outbreak of COVID-19 that created high market volatility and had complex implications for the biotechnology and pharmaceutical industries, and thus represents a challenge to our model. The dataset contains two main types of data: daily trading data (numerical data) and daily online news data (textual data). First, the daily trading data are downloaded from the Wind Financial Terminal,[2] including opening prices, lagged opening prices, high prices, low prices, closing prices, previous day's closing prices, returns, trading volumes, daily average prices, total market capitalization, and number of shares outstanding. These data are used as proxies to capture value anomalies and trading information. Second, the daily online news is acquired from Wind Financial Terminal, Baidu News,[3] and Sina Finance[4] for the same period, totaling 1,556,

[2] Wind Financial Terminal website: http://www.wind.com.cn.
[3] Baidu News website: http://news.baidu.com.
[4] Sina Finance website: http://finance.sina.com.cn.

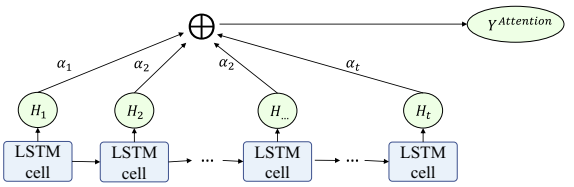


Fig. 3. Graphical illustration of the architecture of attention enhanced LSTM.

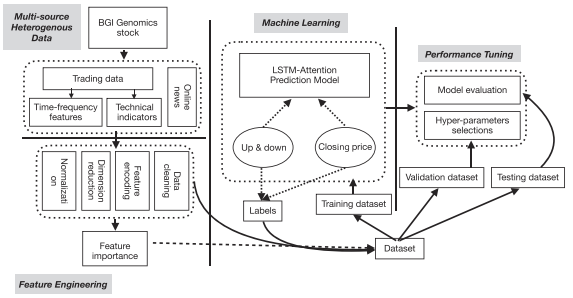


Fig. 4. Graphical illustration of the architecture of the proposed framework.

552, and 2,554 stock-related news articles for BGI Genomics from each source, respectively. The average daily number of articles is 6.13.

Because financial markets are usually chaotic, the inherent nonlinearity and non-stationarity in trading data pose challenges to the trend-based prediction of closing prices. To reduce the impact of non-stationarity, we use the ITD method introduced in Section 2.1 to decompose the original non-stationary closing price data into several quasi-stationary components. In the process of constructing the time–frequency features from the closing price series, in order to avoid using future information, we use the ITD method to process each subsequence of closing prices following a sliding method with given parameters regarding the minimum length of time series h and the number of PRCs L.[5] The process of the improved ITD method is summarized in Algorithm 1. Then, the resulting PRCs, residual, IAs, and IFs together form the time–frequency features that can be regarded as a new data source to improve forecasting performance.

Algorithm 1: Improved ITD method for constructing time–frequency features from the closing price series.

Input: The size of the daily closing price time series N, the minimum length of time series h ($h < N$), the number of PRCs L.

1: Set the PRCs $c1, c2, \ldots, cL = \{\}$, the residual $c(L+1) = \{\}$, the IFs $p1, p2, \ldots, pL = \{\}$, and the IAs $a1, a2, \ldots, aL = \{\}$.

2: **for** $t = h, \ldots, N$ **do**

3: Decompose the subsequence $sub\_close := \{close_i\}_{i=1}^{t}$ into L PRCs and compute the corresponding IA and IF for each PRC by the ITD method. Here, we use PRC_j, sub_IA_j, and sub_IF_j to represent the j-th PRC, IA, and IF, respectively, $j = 1, 2, \ldots, L$.

4: Compute the residual of the subsequence sub_close and call it Res, i.e., $Res = sub\_close - \sum_{j=1}^{L} PRC_j$.

5: Obtain the time–frequency features at time t, that is, $c1_t = PRC_{1,t}, \ldots, cL_t = PRC_{L,t}$, $c(L+1)_t = Res_t$; $a1_t = sub\_IA_{1,t}, \ldots, aL_t = sub\_IA_{L,t}$ and $p1_t = sub\_IF_{1,t}, \ldots, pL_t = sub\_IF_{L,t}$.

6: **end for Output:** The time–frequency features: $c1, c2, \ldots c(L+1)$; $a1, a2, \ldots, aL$; $p1, p2, \ldots, pL$.

Further, because constructing new features from existing data is also a reliable method to improve prediction performance in the field of machine learning [28], several technical indicators based on trading data and previously studied by financial experts are generated as another novel data resource. As illustrated in Fig. 4, two categories of sophisticated quantitative indicators presented by Kakushadze [21] and Kingma and Ba [25] are adopted in this study. We refer to these as Alpha 101 and Alpha 191 indicators, because they contain indicators of size 101 and 191, respectively.

[5] Note that the length of the time series processed by the improved ITD algorithm is at least h, so the time–frequency features of the first h days cannot be derived.

3.3. Feature engineering

3.3.1. Data cleaning

In this subsection, we first clean the news data and technical indicators; other data are already structured and complete. The online news articles are sorted by date as they are obtained from three platforms for the same period, as described in Section 3.2. Alpha 101 and Alpha 191 indicators with more than 3% of values missing during the sample period are excluded. Additionally, in order to ensure data consistency, we discard data (including trading data, online news, and Alpha 101 and Alpha 191 indicators) from the first h days because the time–frequency features of these h days are unavailable.

3.3.2. Feature encoding

One major challenge in handling the online news is how to convert articles' content into numerical vectors that can be processed by a prediction model. This challenge belongs to the Chinese text feature encoding problem in the field of NLP. Therefore, we apply NLP techniques to solve it by the following three steps.

First, the online news articles sorted by date are passed to TextRank4ZH,[6] an abstract extraction toolkit for Chinese text, to withdraw the most significant sentences from the daily online news based on the built-in sophisticated Pagerank algorithm [29]. We limit each abstract to a maximum of ten sentences. Second, both SnowNLP[7] and Senta[8] tools are employed for sentiment analysis on each sentence of these abstracts. SnowNLP is a class library written in Python, inspired by the TextBlob library, that can handle Chinese text content including tasks such as Chinese word segmentation, sentences segmentation, part-of-speech tagging, sentiment analysis, text categorization, conversion of pinyin, traditional simplification, and text similarity analysis. The Senta (also called SKEP) model is trained to learn a unified sentiment representation for multiple sentiment analysis tasks by embedding sentiment information at the word, polarity, and aspect levels into a pre-trained sentiment representation [41]. Third, we collect the results of sentiment analysis on the abstracts of daily news articles obtained from SnowNLP and Senta by date, and compute their daily mean values and daily standard deviations. Therefore, the resulting four numerical features generated from the daily online news articles are not only able to quantify the sentiment of daily news to a certain extent, but also can be the inputs fed into the prediction model.

3.3.3. Dimension reduction

In the fields of statistics, machine learning, and information theory, dimension reduction is a process of reducing the number of random variables under consideration to obtain a set of principal variables [38]. As suggested by Pestov and Vladimir [31] and Rico-Sulayes [37], the advantages of dimension reduction include: (1) Saving storage space and time required; (2) Eliminating multicollinearity to improve the interpretation of machine learning model parameters; (3) When scaling down to very small sizes (such as 2-dimension or 3-dimension), the data become much easier to visualize; (4) It avoids the curse of dimensionality.

Feature projection (also called feature extraction), a well-known approach to dimension reduction, transforms the data from a high-dimensional space to a space of fewer dimensions. The data transformation is allowed to be nonlinear [8] or linear. In the paper we adopt principal component analysis (PCA) [20,43], a commonly used method of linear transformation. The method can reduce the dimensions of both Alpha 101 and Alpha 191 indicators and extract information from them, as their dimensions are too high for direct classification or regression.

3.3.4. Normalization

To ensure prediction performance is not impacted by differences in the scales on which features' values are measured, data normalization techniques are commonly used to transform the values of different scales to a notionally common scale. There are different types of normalization in statistics, such as min–max feature scaling, studentized residual, and standard score [49]. In this paper, we use the standard score for normalization, thus the values of features are scored by subtracting the sample or estimated mean and dividing by the sample standard deviation or another estimate of standard deviation.

3.3.5. Feature importance

High dimensionality of features is likely to cause redundancy, which may negatively affect prediction performance. Unlike feature projection methods that convert a high-dimensional feature space to a low-dimensional space, the computation of feature importance is a vital method that selects features according to their significance, from high to low, to achieve feature dimensionality reduction. In addition, the calculation and visualization of feature importance also help data mining analysts to understand the contribution of features. Ensemble decision-trees-based techniques, such as GBDT[9] and RF,[10] are

[6] TextRank4ZH github website: https://github.com/letiantian/TextRank4ZH.
[7] SnowNLP github website: https://github.com/isnowfy/snownlp.
[8] Senta github website: https://github.com/baidu/Senta.
[9] GBDT algorithm in Scikit-learn library: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier.
[10] RF algorithm in Scikit-learn library: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.

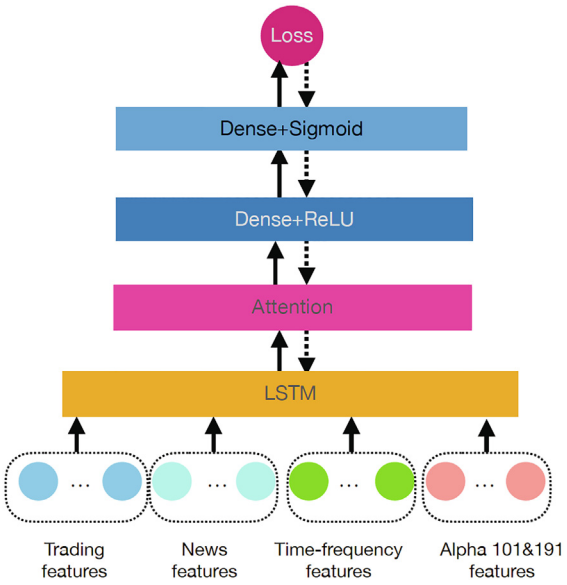


Fig. 5. Schematic of the LSTM-Attention prediction model.

common methods for computing feature importance by counting the number of occurrences in the trees of the features as input candidates. If a feature appears more frequently in these trees, it is more important and vice versa.

After completing the procedures above, we obtain representative features of trading data, online news, time–frequency data, and Alpha 101 and Alpha 191 technical indicators for each trading day, except for the first h days (being the minimum length of time series processed in the ITD method). Table 9 in the Appendix shows common descriptive statistics such as the mean, standard deviation, minimum, and maximum of the features from trading data, news data, and time–frequency data. The summary statistics of features from the Alpha 101 and Alpha 191 technical indicators are listed in Tables 10 and 11 in the Appendix, respectively.

In dealing with a task of forecasting, we suppose that X_t denotes all representative features that serve as input candidates for alternative prediction models at date t. In fact, we can use only the information in X_t at date t to make predictions, or use the information over the l days until that date, which can be represented as $\widehat{X}_t := [X_{t-l+1}, \ldots, X_t]$. This means that the features' time window size l can be 1 or larger than 1. Generally speaking, the richer the information, the better the prediction performance; however, if too much historical information is added during the training, it will be counterproductive due to the curse of dimensionality. We will discuss the sensitivity of prediction performance to the features' time window size using experiments conducted in Section 4.1.1.

3.4. Attention enhanced LSTM (LSTM-Attention) prediction model

Once the representative features are obtained, the corresponding labels need to be constructed as input into our model for supervised learning. However, the labels depend on the specific prediction task. For instance, they are assigned to be the next day's closing price for the task of predicting the next day's closing price; in the task of predicting price direction, the label is equal to 1 when the next day's closing price is greater than or equal to the price on the current day, and 0 otherwise. The resultant dataset with features and labels is further divided into the training, validation, and testing datasets in a specific ratio.

Training a deep neural network is a complex task due to the potential for high dimensionality and nonlinearity. Building on the existing studies discussed in Section 1, we propose an attention enhanced LSTM model (abbreviated as LSTM-Attention) for prediction of stock closing prices, which is adaptable for multi-source heterogeneous information fusion. The schematic of the LSTM-Attention model is depicted in Fig. 5.

As shown in Fig. 5, heterogeneous features are fed as inputs, then passed through the LSTM layer, the Attention layer, the fully connected layers (Dense[11]), and the activation function layers (ReLU[12] and Sigmoid[13]). The detailed process is summarized in Algorithm 2.[14] During the learning phase of the model, the losses on the training dataset are fed back layer by layer

[11] Dense layer in Keras platform: https://keras.io/api/layers/core_layers/dense.
[12] ReLU layer in Keras platform: https://keras.io/api/layers/activation_layers/relu.
[13] Sigmoid layer in Keras platform: https://keras.io/api/layers/activations/#sigmoid-function.
[14] Although step 5 reported in Algorithm 2 does not appear in the forward process of the LSTM-Attention model, we report it here because the subsequent learning process depends on it.

with the help of the back-propagating method [14], and update the undetermined weights of each layer using a gradient descent based method, such as stochastic gradient descent (SGD) [4], AdaGrad [9], or Adam [22]. In addition, there are many hyper-parameters in the LSTM-Attention model, including the number of neurons in the LSTM and Dense layers, the number of iterations, and the learning rate. The values of these hyper-parameters are selected according to their performance on the validation dataset; further details are described in Section 3.5.1.

The LSTM-Attention model can be used to handle classification and regression tasks, but needs to be adjusted slightly for these two different tasks. Specifically, for the task of classification, the loss is generally selected from probabilistic losses,[15] which include binary cross entropy, categorical cross entropy, and KL (Kullback–Leibler) divergence. For the regression task, the true labels must be transformed to between 0 and 1 because the outputs of the Sigmoid layer are ranged from 0 to 1, and the loss can be selected from regression losses,[16] which cover mean square error, mean absolute error, mean absolute percentage error, cosine similarity, and similar. In this paper, the LSTM-Attention model is used to predict the next day's stock price direction, and to predict the next day's closing price of BGI Genomics, which are essentially classification and regression tasks, respectively.

Algorithm 2: Forward process of the LSTM-Attention prediction model.

Input: The training samples $\left\{\widehat{X}_t, Y_t\right\}_{t=l}^{N}$, where $\widehat{X}_t := [X_{t-l+1}, \ldots, X_t]$ and Y_t represent the features and the label of the t-th sample, respectively, and l is the features' time window size; o_1 is the number of neurons in the LSTM layer; and o_2 is the number of neurons in the first Dense layer.

1: Compute the outputs Y_i^{LSTM} of size o_1 of the LSTM layer with the input $\widehat{X}_t := [X_{t-l+1}, \ldots, X_t]$ by Eq. (9): $Y_i^{LSTM}, H_i, C_i = LSTM(X_{t-l+i}, H_{i-1}, C_{i-1})$, $i = 1, 2, \ldots, l$.

2: Calculate the output $Y^{Attention}$ of size o_1 of the Attention layer with the inputs $\{H_i\}_{i=1}^{l}$ by Eq. (13): $Y^{Attention} = Attention(H_1, H_2, \ldots, H_l)$.

3: Compute the output Y^{Dense1} of size o_2 of the first nonlinear Dense layer with the input $Y^{Attention}$: $Y^{Dense1} = ReLU\left(W^{Dense1} Y^{Attention} + b^{Dense1}\right)$, where W^{Dense1} of size $o_1 \times o_2$ is the undetermined weights, b^{Dense1} denotes the bias, and $ReLU(\cdot)$ denotes the nonlinear activation function of ReLU.

4: Similar to step 3, obtain the output $y^{Sigmoid}$ of the second nonlinear Dense layer with the input Y^{Relu}, i.e., $Y^{Sigmoid} = \sigma\left(W^{Dense2} Y^{Relu} + b^{Dense2}\right)$, where $\sigma(\cdot)$ denotes the sigmoid function.

Output: The prediction of the LSTM-Attention model for the t-th sample: $Y^{Sigmoid}$.

5: Calculate the loss $Loss\left(Y^{Sigmoid}, Y_t\right)$ between the predicted result $Y^{Sigmoid}$ and the true label Y_t for the t-th sample, where the loss function is generally selected according to its specific prediction task.

3.5. Performance tuning and model evaluation

3.5.1. Hyper-parameters selection

Hyperopt[17] is a Python library for serial and parallel optimization over awkward search spaces for hyper-parameters, which may include real-valued, discrete, and conditional dimensions [3]. Currently, there are three algorithms implemented in Hyperopt, that is, random search, tree of Parzen estimators (TPE), and adaptive TPE. Hyperas[18] is a convenience wrapper around Hyperopt for fast prototyping with Keras models. Hyperas enables us to use the functions of Hyperopt without having to learn its syntax. Hence, we adopt Hyperas as an approach to select hyper-parameters in our experiments.

3.5.2. Model evaluation

- Metrics of classification performance

The correctness of classification can be evaluated by computing the number of correctly recognized class examples (true positives, *TP*), the number of correctly recognized examples that do not belong to the class (true negatives, *TN*), and examples that either were incorrectly assigned to the class (false positives, *FP*) or were incorrectly not recognized as class examples (false negatives, *FN*). The metrics *Accuracy*, *Precision*, *Recall*, and *F-measure* are widely used to evaluate the performance of a classification task such as the prediction of stock price direction. These metrics are defined in Table 1, where *Accuracy* is the number of correctly classified samples on total data, *Precision* gives the number of correct positive predictions divided

[15] Probabilistic losses in Keras platform: https://keras.io/api/losses/probabilistic_losses.
[16] Regression losses in Keras platform: https://keras.io/api/losses/regression_losses.
[17] Hyperopt github website: https://github.com/hyperopt/hyperopt.
[18] Hyperas github website: https://github.com/maxpumperla/hyperas.

by the number of all positive class values returned by the classifier in the test data, and *Recall* is the number of positive results divided by all relevant samples, which is also called Sensitivity or the True Positive Rate. *F-measure* is the harmonic mean of *Precision* and *Recall*, which achieves its maximum value when $Precision = Recall = 1$.

- Metrics of regression performance

Four common and highly statistical evaluation metrics are employed to assess the regression performance of the models of interest; these are the mean absolute error (*MAE*), the mean squared error (*MSE*), the root mean square error (*RMSE*), and the mean absolute percentage error (*MAPE*). These metrics are defined in Table 2.

- Metrics of trading strategy performance

Because profits are not proportional to the performance of the classification or regression, we use a simple trading strategy named long/short strategy, as suggested by Zhou et al. [47], to examine the profitability of our proposed framework. Fig. 6 shows the rules of the long/short strategy. Based on the predicted value from the LSTM-Attention model, this strategy introduces a buy-threshold and a sell-threshold to decide whether to change the position from -1 to 1, or from 1 to -1, or just to keep the position. Following Zhou et al. [47], in our experiments without considering transaction costs, the values of the buy-threshold and sell-threshold are set as 0.50.

Five metrics are chosen to evaluate the trading strategy performance. They are the Sharpe ratio (*SR*), average annual return (*PnL*), maximum drawdown (*MD*), *PnL*/*MD*, and the total number of entered trades (*TradeCount*). Note that the *SR* measures the risk-adjusted return, the *PnL* indicates the average annual return, the *MD* indicates the largest accumulated loss due to a sequence of drops over the period of investment, and *PnL*/*MD* is computed as the *PnL* divided by the *MD*. The definitions of *SR*, *PnL*, *MD*, and *PnL*/*MD* are listed in Table 3.

3.6. Proposed framework for BGI Genomics processing

In order to promote the understanding and use of our proposed framework, together with the overview in Section 3.1 and elaborations of how it works from Section 3.2 to Section 3.5, we summarize the framework's four main stages and sequential steps as below.

Stage 1. Multi-source heterogeneous data collection

Input: The minimum length of time series h and the number of PRCs L that appear in the ITD method.

1: Collect the daily trading data, including the open price (*open*), lagged open price (*lag_open*), high price (*high*), low price (*low*), closing price (*close*), previous day's closing price ($prev\_close$), return (r), trading volume (vol), daily average price (avg), total market capitalization (cap), and number of shares outstanding ($share$), of BGI Genomics for the period from July 14, 2017 to July 21, 2020 from Wind platform; and crawl the news data for the same period from Wind platform, Baidu News, and Sina Finance web portals. Assume that the number of trading days during this period is N.
2: Generate the time–frequency data for *close* using the improved ITD method summarized in Algorithm 1 with the given h and L. The resulting $c1 \sim c(L+1), a1 \sim aL$, and $p1 \sim pL$ form the time–frequency features. In this process, the time–frequency features of the first h days are discarded because the length of the time series processed by the ITD method is limited to at least h.
3: Compute the Alpha 101 and Alpha 191 technical indicators, which are denoted $alpha101\_001 \sim alpha101\_101$ and $alpha191\_001 \sim alpha191\_191$, respectively.

Output: The trading data ($open, lag\_open, high, low, close, prev\_close, r, vol, avg, cap, share$), the daily online news data, the time–frequency features ($c1 \sim c(L+1), a1 \sim aL, p1 \sim pL$), and the Alpha 101 ($alpha101\_001 \sim alpha101\_101$) and Alpha 191 ($alpha191\_001 \sim alpha191\_191$) technical indicators.

Stage 2. Feature engineering

Input: The dimension of the space reduced by the PCA method k, the features' time window size l.

1: Exclude the technical indicators from $alpha101\_001 \sim alpha101\_101$ and $alpha191\_001 \sim alpha191\_191$ if their proportion of missing values exceeds 3%. Then, $PCA(k)$ is used to reduce the Alpha 101 and Alpha 191 technical indicators to k features.
2: Integrate the news data by date. Then, the TextRank4ZH toolkit is leveraged to extract abstracts from the integrated news, where each abstract has a maximum of ten sentences. Both SnowNLP and Senta techniques are further applied to compute the sentiments for each sentence of the abstracts. Their statistical mean values and standard deviations by date are calculated, i.e., $snow\_avg, senta\_avg, snow\_std$, and $senta\_std$.

Table 1
Metrics of classification performance.

Metric	Expression	Metric	Expression
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$	*Recall*	$\frac{TP}{TP+FN}$
Precision	$\frac{TP}{TP+FP}$	*F-measure*	$\frac{2 * Precision * Recall}{Precision+Recall}$

Table 2
Metrics of regression performance, where p_t and $\hat{p}_t$ are the actual and predicted values at time t, respectively. N represents the number of data sample points.

Metric	Expression	Metric	Expression
MAE	$\frac{1}{N}\sum \lvert p_t - \hat{p}_t \rvert$	MSE	$\frac{1}{N}\sum (p_t - \hat{p}_t)^2$
RMSE	$\sqrt{\frac{1}{N}\sum (p_t - \hat{p}_t)^2}$	MAPE	$\frac{1}{N}\sum \lvert \frac{p_t - \hat{p}_t}{p_t} \rvert * 100$

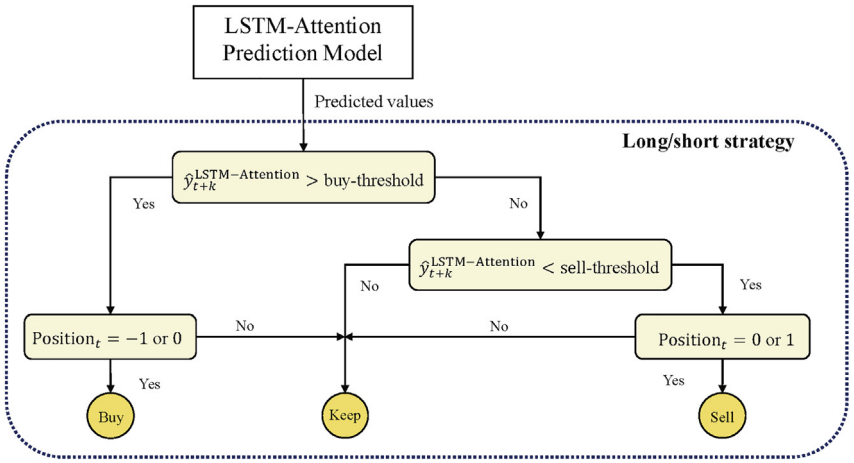


Fig. 6. Graphical illustration of the long/short trading strategy.

Table 3
Metrics of trading strategy performance, where r_i denotes the return of year i, R_t is the accumulated return until date t over a period N, $\mu(R_t)$ and $\sigma(R_t)$ are the corresponding mean and standard deviation of the return R_t.

Metric	Expression	Metric	Expression
SR	$\frac{\mu(R_t)}{\sigma(R_t)}$	*MD*	$\max_{\tau\in(0,N)}\left(\max_{t\in(0,\tau)}(R_t - R_\tau)\right)$
PnL	$\left(\prod_{i=1}^{N}(1+r_i)\right)^{\frac{1}{N}} - 1$	*PnL/MD*	$\frac{\left(\prod_{i=1}^{N}(1+r_i)\right)^{\frac{1}{N}}-1}{\max_{\tau\in(0,N)}\left(\max_{t\in(0,\tau)}(R_t-R_\tau)\right)}$

3: Construct the labels $\{Y_t\}_{t=1}^{N-1}$ according to different prediction tasks, where Y_t denotes the label of date t. Specifically, in the next day's closing price prediction task, $Y_t = close_{t+1}$; in the task of the next day's closing price direction prediction, $Y_t = 1$ if $close_{t+1} \geqslant close_t, Y_t = 0$ otherwise.
4: Integrate all the features by date. The data for the first h days and the last day are discarded because the time–frequency features of the first h days and the labels on the last day are unavailable. These features are then scored using the standard scoring method and the final set of features are derived, i.e., $\{X_t\}_{t=h}^{N-1}$. Assume that the features over the l days until date t are $\widehat{X}_t := [X_{t-l+1}, \ldots, X_t], t = h + l - 1, \ldots, N - 1$.
5: Divide the samples $\left(\widehat{X}_t, Y_t\right)_{t=h+l-1}^{N-1}$ into the training, validation, and testing datasets in the ratio $7 : 2 : 1$.

Output: The training, validation, and testing datasets, which are denoted $\left(\widehat{X}_t, Y_t\right)_{t=h+l-1}^{N_t+h+l-2}$, $\left(\widehat{X}_t, Y_t\right)_{t=N_t+h+l-1}^{N_t+N_v+h+l-2}$, and $\left(\widehat{X}_t, Y_t\right)_{t=N_t+N_v+h+l-1}^{N-1}$, respectively, where N_t and N_v represent the corresponding number of training and validation datasets.

Stage 3. Training the LSTM-Attention model

Input: The training dataset $\left(\widehat{X}_t, Y_t\right)_{t=h+l-1}^{N_t+h+l-2}$; the hyper-parameters that appear in the LSTM-Attention model, including the output size of the LSTM layer o_1, the output size of the first Dense layer o_2, the number of iterations, the learning rate.

1: Select loss function according to the specific prediction task. In this paper, the mean square error is chosen as the loss function for the regression task, and the binary cross entropy is chosen for the classification task.

2: Train the LSTM-Attention model on training dataset $\left(\widehat{X}_t, Y_t\right)_{t=h+l-1}^{N_t+h+l-2}$ using the general forward and backward feedback method. The forward process is summarized in Algorithm 2, and the back propagation process is described in Section 3.4.

Output: A trained LSTM-Attention model that can make predictions for any sample from $\left(\widehat{X}_t, Y_t\right)_{t=h+l-1}^{N-1}$.

Stage 4. Performance tuning and model evaluation

Input: The validation dataset $\left(\widehat{X}_t, Y_t\right)_{t=N_t+h+l-1}^{N_t+N_v+h+l-2}$, the testing dataset $\left(\widehat{X}_t, Y_t\right)_{t=N_t+N_v+h+l-1}^{N-1}$.

1: Use the Python toolkit "Hyperas" on the validation dataset for hyper-parameters selection.
2: Set different prediction tasks and use the corresponding metrics listed in Tables 1–3 to evaluate the prediction performance on the testing dataset.

Output: The prediction performance in different prediction tasks.

4. Empirical results and evaluation

In the section, using the minimum length of time series $h = 50$ and the number of PRCs $L = 5$, we conduct experiments to evaluate the feasibility of our framework for fusing the heterogeneous information related to BGI Genomics. In addition, we discuss the sensitivity of the prediction performance with respect to the size of the features' time window and the dimension reduction by the PCA method. Furthermore, we compare the results of our LSTM-Attention prediction model with the LR, SVM, GBDT, and original LSTM models, in terms of their performance in predicting the next day's price direction and the next day's closing price, and in developing a trading strategy. Here the original LSTM model is similar to the LSTM-Attention model depicted in Fig. 5 but without the Attention layer. To ensure a fair comparison with these models, the method of determining the hyper-parameters that appear in these models is the same as that for the LSTM-Attention model, as described in Section 3.5.1.

All the experiments are performed in Python 2.7.3 on a Dell Precision 5820 tower with Intel(R) Xeon(R) W-2102 processor (2.90 GHz), 64G memory, and Ubuntu 18.04.3 operating system. Specifically, the experiments related to the LR, SVM, and GBDT models are carried out using the open-source machine learning library Scikit-learn,[19] and the original LSTM and our proposed LSTM-Attention models are implemented using the deep learning platform Keras. Note that the L_1/L_2 regular term, as a general method to avoid overfitting, has become the standard configuration of most models in the Scikit-learn library and Keras platform, which users can configure flexibly. The source code of this study has been publicly released.[20]

4.1. Preliminary analysis

As shown in Fig. 4, before applying the proposed LSTM-Attention prediction model, we conduct a preliminary analysis to verify the feasibility of multi-source input data for fusing the heterogeneous information, the necessity of dimension reduction for filtering noise, the importance of individual features, and to optimize their combination.

4.1.1. The feasibility of multi-source data fusion

After the implementation of data cleaning, feature encoding, and normalization as described in Section 3.3, we first investigate the quality of a single source (i.e., trading data), and then study the added value of features derived from news data, time–frequency data, Alpha 101 technical indicators, Alpha 191 technical indicators, and their combinations. Based on these datasets from single or double data sources, a series of experiments are conducted. Table 4 presents and compares the performance of price direction predictions of the LR, SVM, and GBDT models through the metrics defined in Table 1 (i.e., *Accuracy*, *Precision*, *Recall*, and *F-measure*) using the testing dataset and different sources of input data. For each model in

[19] Website of Scikit-learn: https://scikit-learn.org/stable/index.html.
[20] The source code can be downloaded from: https://github.com/zhoudafa08/heterogeneous_data_processing.

Table 4
Accuracy, *Precision*, *Recall*, and *F-measure* metrics for the testing dataset derived using the LR, SVM, and GBDT models and different sources of input data, with features' window size $l = 1, 5, 10$ in the three panels.

Model	Trading data	News data	Time–frequency	Alpha 101	Alpha 191	*Accuracy*	*Precision*	*Recall*	*F-measure*
Panel A: using features' time window size $l = 1$									
LR	✓					0.5072	0.5111	0.6571	0.5750
	✓	✓				0.5652	0.5556	0.7143	0.6250
	✓		✓			**0.5797**	**0.6000**	0.7105	**0.6506**
	✓			✓		0.5362	0.3778	0.8095	0.5152
	✓				✓	0.4638	0.2000	**0.9000**	0.3273
SVM	✓					0.4058	0.1111	**0.8333**	0.1961
	✓	✓				0.4782	0.4222	0.6552	0.5135
	✓		✓			0.5217	0.4889	0.6875	0.5714
	✓			✓		**0.5797**	**0.7333**	0.6600	**0.6947**
	✓				✓	0.4782	0.3111	0.7368	0.4375
GBDT	✓					0.4202	0.2667	0.6316	0.3750
	✓	✓				0.4638	0.4667	0.6176	0.5316
	✓		✓			0.5942	**0.7111**	0.6809	**0.6957**
	✓			✓		**0.6087**	0.5556	**0.7813**	0.6494
	✓				✓	0.5507	0.5553	0.7059	0.6076
Panel B: using features' time window size $l = 5$									
LR	✓					0.5882	0.5454	0.7500	0.6316
	✓	✓				**0.6176**	0.5527	**0.8214**	0.6389
	✓		✓			0.5882	**0.7273**	0.6667	**0.6957**
	✓			✓		0.6029	0.5000	0.8148	0.6197
	✓				✓	0.5147	0.4773	0.6774	0.5600
SVM	✓					0.4264	0.1364	**0.8571**	0.2353
	✓	✓				0.5588	0.4772	0.7500	0.5833
	✓		✓			**0.6176**	**0.8864**	0.6500	**0.7500**
	✓			✓		0.5294	0.7045	0.6200	0.6596
	✓				✓	0.3823	0.0909	0.6667	0.1600
	✓					0.5714	0.5538	0.6545	0.6000
GBDT	✓	✓				0.4706	0.4773	0.6176	0.5385
	✓		✓			0.5147	0.5909	0.6341	0.6118
	✓			✓		**0.6176**	**0.9318**	0.6406	**0.7593**
	✓				✓	0.4706	0.2500	**0.7857**	0.3793
Panel C: using features' time window size $l = 10$									
LR	✓					0.5672	**0.4884**	0.7500	0.5915
	✓	✓				0.5522	0.3953	0.8095	0.5313
	✓		✓			0.4179	0.2326	0.6250	0.3390
	✓			✓		**0.5970**	0.4651	**0.8333**	**0.5970**
	✓				✓	0.5075	0.3721	0.7273	0.4923
SVM	✓					0.4029	0.0930	**0.8000**	0.1667
	✓	✓				0.4030	0.0930	**0.8000**	0.1667
	✓		✓			0.5224	0.5116	0.6667	0.5789
	✓			✓		0.4627	0.3488	0.6522	0.4545
	✓				✓	**0.5821**	**0.6977**	0.6667	**0.6818**
GBDT	✓					0.3881	0.2558	0.5500	0.3492
	✓	✓				0.3731	0.2558	0.5238	0.3438
	✓		✓			0.3881	0.2093	0.5625	0.3051
	✓			✓		0.4030	0.4186	0.5455	0.4737
	✓				✓	**0.5672**	**0.6512**	**0.6667**	**0.6588**

Table 4, the first row reports the results obtained from the trading data alone, and the other rows report the results obtained from the trading data in combination with the news data, the time–frequency data, the Alpha 101 technical indicators, and the Alpha 191 technical indicators. For each metric and model, the best outcome derived from the various sources of input data is highlighted in bold. To investigate the influence of features' time window size l, we conduct experiments using $l = 1$ and present the corresponding results in Panel A. Panels B and C show the results using $l = 5$ and $l = 10$, respectively.

Observing the results displayed in three panels of Table 4, we find the following. (1) Significant differences exist in the values of metrics for the LR, SVM, and GBDT models. Comparing the results in the three panels, the GBDT model reaches the highest value of *F-measure* (i.e., 0.7593) using $l = 5$ and the combination of trading data and Alpha 101 trading indicators. The results in Panel B also demonstrate that all three models achieve the highest value of *Accuracy* (i.e., 0.6176 for all three models) using the features' time window size $l = 5$. (2) Data source, model, and features' time window size jointly influence the performance of price direction prediction. Using the *Accuracy* and *F-measure* metrics for the three models in Panel A, we can find the value added by news data, because the values of these two metrics for the combination of news data and trading data are larger than the equivalent values using only trading data. In addition, we can use these two metrics to identify the value added by time–frequency data in many cases, such as the results derived from the three models with $l = 1$ in Panel A,

the SVM model with $l = 5$ in Panel B, and the GBDT model with $l = 10$ in Panel C. Further, for the GBDT model with $l = 10$ in Panel C, the results of all metrics suggest the importance and added value of Alpha 191 technical indicators. (3) The results derived using combinations of features from different data sources do not always outperform the results derived from the trading data alone. Indeed, adding too many features may interfere with the identification of the relevant factors for predictions and lead to worse prediction performance. Given the high dimensionality of the input features, such as in the two groups of technical indicators, it is necessary to perform feature selection or dimension reduction before invoking a classifier or a regressor.

4.1.2. The necessity of dimension reduction

Although the individual performance of each group of technical indicators in combination with the trading data has been tested in Table 4, here we further investigate the results obtained when a dimension reduction method is employed. PCA is now employed to extract representative features from the two groups of technical indicators by choosing sufficient eigenvectors to explain a percentage of the variance in the original data [42]. The results obtained using the LR, SVM, and GBDT models and different groups of input data related to technical indicators, with the PCA method applied, are presented in Table 5. The PCA parameter in the fifth column is the number of principal components, $k = 2,\ 3,\ 4$ or 5. Based on the results reported in Table 4, the features' time window size l has been experimentally set to 5 here. For each metric and model, the best outcome derived from the various combinations of input data is highlighted in bold.

As shown in Table 5, out of the different combinations, the highest observed value of *Accuracy* (i.e., 0.6087) is achieved by both the SVM and GBDT models with the combination of trading data, Alpha 101 technical indicators, and PCA(4). The highest value of *F-measure* (i.e., 0.7077) is obtained by the GBDT model with the same combination. However, the optimal *F-measure* scores for each model (i.e., 0.6742 for LR, 0.6957 for SVM, and 0.7077 for GBDT) are achieved with different combinations of technical indicators and number of PCA principal components. Overall, these findings indicate that for models to have a better interaction between the classification task for real-time price direction prediction and alternative dimension reduction methods, they must be designed to filter noise when dealing with multi-source heterogeneous information fusion. Among all the experiments, those conducted with PCA(5) exhibit relatively better performance in terms of *Accuracy* and *F-measure* metrics.

4.1.3. The importance of individual features

Furthermore, after reducing the number of dimensions of feature space, we can further rank these features based on a measure of the importance or contribution of each feature used in the GBDT model.[21] Fig. 7 presents the importance score of each related feature, which is calculated by the weight of the number of times that each feature is used to split the data across all trees via the GBDT model. As mentioned before, if the score is large, the corresponding feature is relatively important. The features' time window size l has been experimentally set to 1, and the number of principal components k produced by the PCA method is 5 here. We can thus see the relative importance of two features for forecasting the next day's stock price direction, which are the second principal component derived from the Alpha 101 technical indicators (i.e., *alpha*101_*v*2) and the standard deviation derived from news data by the Senta technique (i.e., *senta_std*). In addition, some features from other categories of input data also contribute to the prediction and have scores of around 0.05.

4.2. Performance of models forecasting the next day's price direction

In this subsection, following the general scheme of the algorithm provided in Section 3.6, we examine the performance of alternative models in forecasting the next day's price direction, including the LR, SVM, GBDT, LSTM, and LSTM-Attention models. After performing the individual experiment ten times for each model, we compute the average estimates and standard deviations of the metrics *Accuracy*, *Precision*, *Recall*, and *F-measure*, reported in Table 6. The features' time window size l has been experimentally set to 5 and the number of principal components k produced by the PCA method is 5 here. For each metric, the best outcome derived from the five models is highlighted in bold.

From the results in Table 6, we can conclude that the highest values of *Accuracy* and *F-measure* are achieved by the LSTM-Attention model, whereas the original LSTM model is suboptimal. This finding indicates that the approach of using a multi-level network classifier of the LSTM model has great potential for forecasting stock price direction based on the signals derived from multi-source heterogeneous information.

4.3. Performance of models forecasting the next day's closing price

We additionally evaluate the prediction performance of the models forecasting the next day's closing price. For this, we use the metrics defined in Table 2, that is, *MAE*, *MSE*, *RMSE*, and *MAPE*. Table 7 presents the values of these metrics for the differences between real closing prices and average estimates from ten repetitions of experiments using the testing dataset and the SVM, GBDT, LSTM, and LSTM-Attention models. The LR model is not considered here because it is used for classifi-

[21] In our experiments, the computation of feature importance is used to recognize the contribution of each feature, rather than to reduce feature dimensionality that has been already preprocessed by the PCA method.

Table 5
Accuracy, *Precision*, *Recall*, and *F-measure* metrics for the testing dataset derived from the LR, SVM, and GBDT models using different combinations of features and PCA methods.

Model	Trading data	Alpha 101	Alpha 191	Reduced dimension k by PCA	*Accuracy*	*Precision*	*Recall*	*F-measure*
LR	✓	✓		PCA(2)	0.5797	0.5556	0.7353	0.6329
	✓	✓		PCA(3)	0.5797	**0.6667**	0.6818	**0.6742**
	✓	✓		PCA(4)	**0.6087**	0.5333	**0.8000**	0.6400
	✓	✓		PCA(5)	0.5942	0.6000	0.7297	0.6585
	✓		✓	PCA(2)	**0.5797**	**0.5556**	**0.7353**	**0.6329**
	✓		✓	PCA(3)	0.5652	**0.5556**	0.7143	0.6250
	✓		✓	PCA(4)	0.5507	0.4889	0.7333	0.5867
	✓		✓	PCA(5)	**0.5797**	**0.5556**	**0.7353**	**0.6329**
SVM	✓	✓		PCA(2)	0.4928	0.5333	0.6316	0.5783
	✓	✓		PCA(3)	0.4493	0.3111	0.6667	0.4242
	✓	✓		PCA(4)	0.5362	**0.6667**	0.6383	**0.6522**
	✓	✓		PCA(5)	**0.5942**	0.6222	**0.7179**	**0.6667**
	✓		✓	PCA(2)	0.3913	0.0889	**0.8000**	0.1600
	✓		✓	PCA(3)	0.4783	0.3556	0.6956	0.4706
	✓		✓	PCA(4)	**0.5942**	0.5778	0.7429	0.6500
	✓		✓	PCA(5)	**0.5942**	**0.7111**	0.6809	**0.6957**
GBDT	✓	✓		PCA(2)	0.4638	0.3333	0.6818	0.4478
	✓	✓		PCA(3)	0.4058	0.3111	0.5833	0.4058
	✓	✓		PCA(4)	**0.6087**	**0.7333**	**0.6875**	**0.7077**
	✓	✓		PCA(5)	0.5797	0.6667	0.6818	0.6742
	✓		✓	PCA(2)	0.4928	0.5556	0.6250	0.5882
	✓		✓	PCA(3)	0.4928	0.5556	0.6250	0.5882
	✓		✓	PCA(4)	0.4927	0.5556	0.6250	0.5882
	✓		✓	PCA(5)	**0.5507**	**0.7111**	**0.6400**	**0.6737**

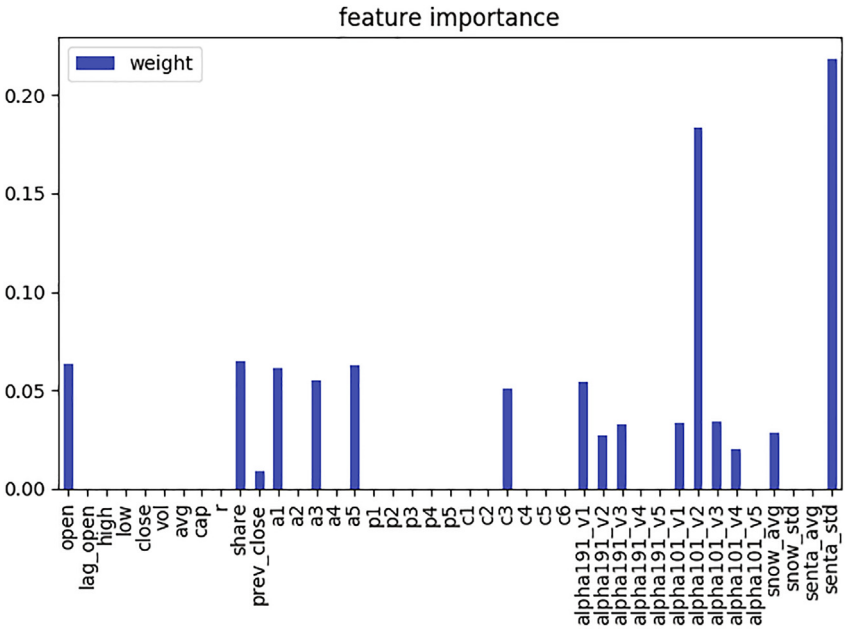


Fig. 7. The relative importance of features based on how many times each feature is used to split data in the GBDT model.

Table 6
Average estimates and standard deviations of the metrics *Accuracy*, *Precision*, *Recall*, and *F-measure* from ten repetitions of experiments using the testing dataset derived and the LR, SVM, GBDT, LSTM, and LSTM-Attention models for prediction of price direction.

Model	*Accuracy*	*Precision*	*Recall*	*F-measure*
LR	$0.5015_{\pm 0.0108}$	$0.5341_{\pm 0.0432}$	$0.6370_{\pm 0.0022}$	$0.5801_{\pm 0.0248}$
SVM	$0.5147_{\pm 0.1133}$	$0.5909_{\pm 0.3691}$	$0.6178_{\pm 0.0458}$	$0.5511_{\pm 0.2344}$
GBDT	$0.5279_{\pm 0.0633}$	$0.5068_{\pm 0.1360}$	$\mathbf{0.6989}_{\pm 0.0848}$	$0.5728_{\pm 0.0880}$
LSTM	$0.5853_{\pm 0.0710}$	$0.7500_{\pm 0.2216}$	$0.6585_{\pm 0.0272}$	$0.6847_{\pm 0.1145}$
LSTM-Attention	$\mathbf{0.6353}_{\pm 0.0267}$	$\mathbf{0.8864}_{\pm 0.0988}$	$0.6642_{\pm 0.0164}$	$\mathbf{0.7568}_{\pm 0.0329}$

Table 7
MAE, *MSE*, *RMSE*, and *MAPE* for the differences between the real closing prices and average estimates from ten repetitions of experiments using the testing dataset and the SVM, GBDT, LSTM, and LSTM-Attention models.

Model	*MAE*	*MSE*	*RMSE*	*MAPE*
SVM	10.6655	159.0070	12.6098	0.0882
GBDT	11.3953	200.9968	14.1773	0.0892
LSTM	7.2207	83.8763	9.1584	0.0578
LSTM-Attention	**6.6329**	**73.1443**	**8.5524**	**0.0518**

cation prediction rather than regression prediction. The features' time window size l has been experimentally set to 5 and the number of principal components k produced by the PCA method is 5 here. For each metric, the best outcome derived from the four models is highlighted in bold.

Comparing the results of these models in Table 7, we see that the LSTM-Attention model achieves overall highly accurate regression results, with the lowest values of *MAE*, *MSE*, *RMSE*, and *MAPE*, indicating that the attention mechanism captures some valuable information that may be ignored by the original LSTM model and the SVM and GBDT models.

In order to validate these regression results, as point estimates may be biased, a natural idea is to produce confidence intervals and further examine the forecasting accuracy. Fig. 8 shows the average estimates from ten repetitions of experiments for the SVM, GBDT, LSTM, and LSTM-Attention models associated with 95.44% confidence intervals (i.e., ± 2 standard deviations) compared with closing prices during the testing period. The red line in each figure is the real closing price during the testing period and the black line represents the average estimate of ten repetitions of experiments. The 95.44% confidence intervals are shown by the shaded areas. The multiple regression analyses conducted in these figures and Table 7 show that the LSTM-Attention model can be considered the most accurate model for the regression task, because the average of predicted prices (corresponding to the point estimation) and confidence intervals (corresponding to the interval estimation) show the most similarity with real closing prices.

4.4. Profitability of different models with long/short strategy

Table 8 shows the *SR*, *PnL*, *MD*, *PnL*/*MD*, *TradeCount*, and Trading Days of the long/short strategy using the alternative models, together with those of the "buy-and-hold strategy", denoted Benchmark, and of the "ex post trading strategy", denoted Ex post. Although it is not the main purpose of this paper to find the best trading rules by using the features based on the models, for such purposes additional considerations such as transaction costs need to be taken into account. We therefore present results in Panel B calculated with inclusion of transaction costs, taken as 0.3% for simplicity.

Comparing the results in the two panels, we find that the values of *SR*, *PnL*, and *PnL*/*MD* in Panel B are lower than corresponding values in Panel A, suggesting that transaction costs weaken the trading performance. Following inclusion of transaction costs, the value of *SR* derived from the GBDT model decreases by the most, and the value from the LSTM model decreases the least. Without considering transaction costs, in Panel A, the original LSTM model records the highest *PnL*/*MD* and the LR model records the lowest *MD* (excluding the Ex post figures). Additionally, according to the *SR* and *PnL* metrics, the LSTM-Attention model in Panel A yields superior prediction performance, evidenced by long/short strategy results, compared to other models. Taking transaction costs into account, the highest values of *SR* and *PnL*/*MD* are achieved by the original LSTM model, and the highest *PnL* is achieved by the LSTM-Attention model. Overall, the results highlight the potential of the original and attention enhanced LSTM models for developing higher quality trading rules and a more profitable trading system.

Furthermore, Fig. 9 shows the evolution of the trading performance as a time series during the testing period, based on the predicted values of the LR, SVM, GBDT, LSTM, and LSTM-Attention models. Performance metrics are those presented in Table 8 and results from the buy-and-hold strategy (i.e., Benchmark) are included for comparison. The normalized compounded profits of the long/short strategy without transaction costs (red line) and with transaction costs (blue line), and the buy-and-hold strategy (black line) are presented in the top panel of each part of the figure. The other panels show the buy and sell signals for the long/short strategy, without inclusion of transaction costs in the middle panels and with transaction costs in the bottom panels.

5. Conclusion

The topic of forecasting stock prices using deep learning interests many researchers and investors because improved prediction accuracy can be expected to bring enormous profit. The recent availability of enormous amounts of both data and computing power has created new opportunities for prediction purposes. This paper combines the attention mechanism with the LSTM model to extract features from multiple sources of data, and investigates their joint impact on the performance of stock price prediction and trading strategy in the case of BGI Genomics. Examining different combinations of features based on multiple sources of data, incorporating daily trading data, online news, technical indicators derived from trading data, and time–frequency features decomposed by the ITD method from closing prices, we identify the best-performing subset of features for information fusion and prediction. We also develop a framework by integrating various

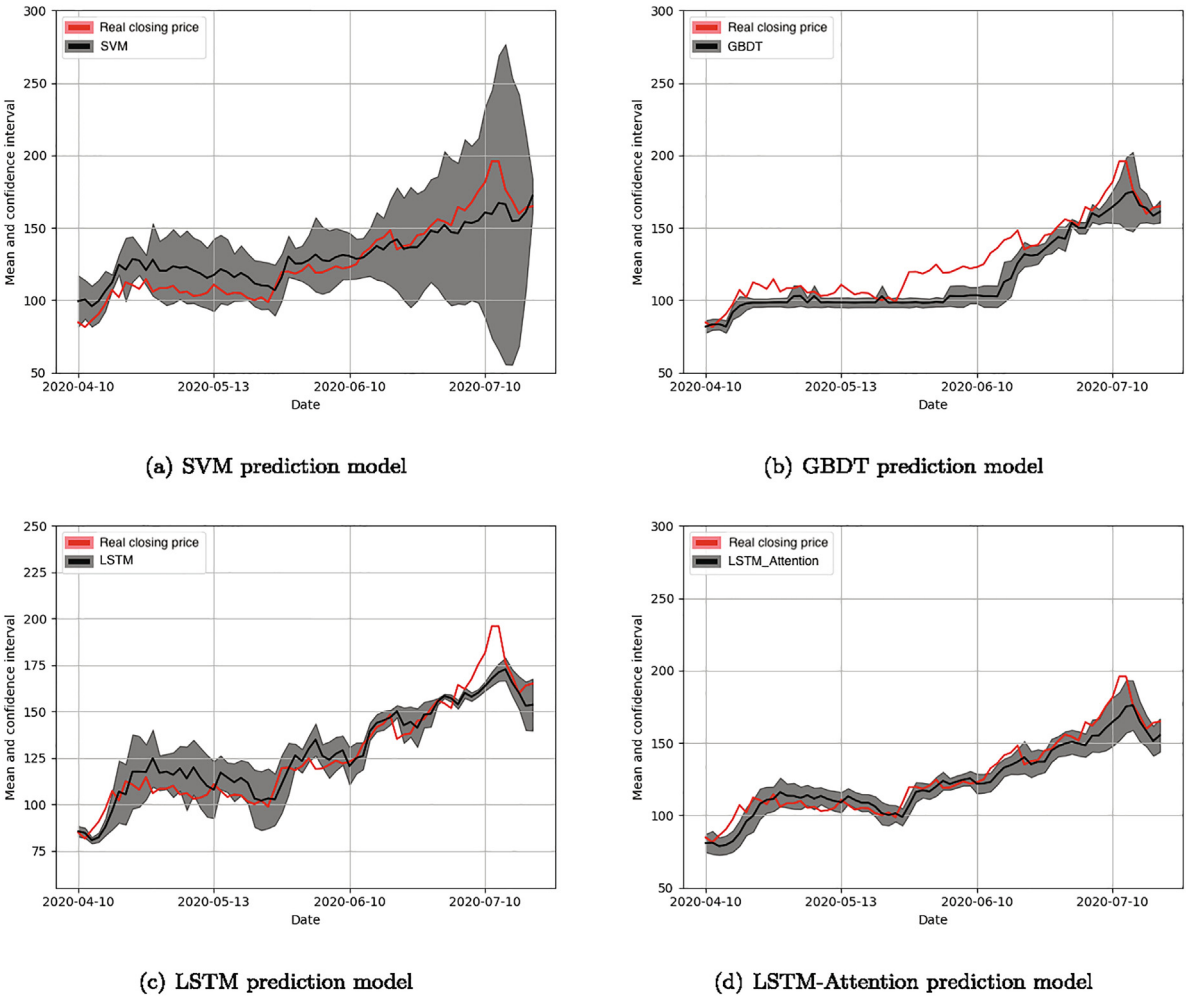
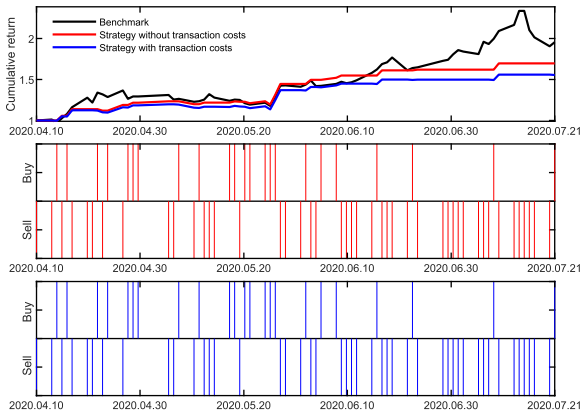


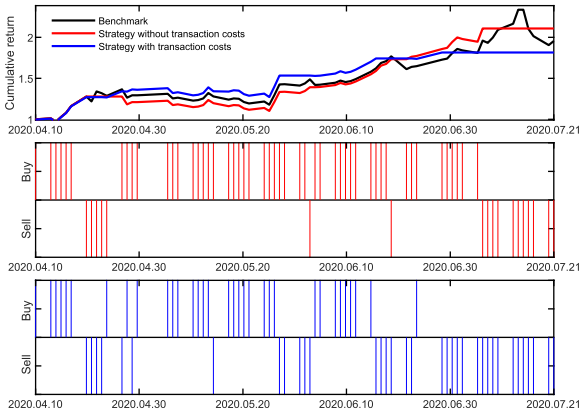
Fig. 8. Average estimates of ten repetitions of experiments for the SVM, GBDT, LSTM, and LSTM-Attention models and the corresponding 95.44% confidence intervals (±2 standard deviations), compared with the real closing price during the testing period.

Table 8
SR, *PnL*, *MD*, *PnL*/*MD*, *TradeCount*, and Trading Days of the long/short strategy using the LR, SVM, GBDT, LSTM, and LSTM-Attention models, together with those of the 'buy-and-hold strategy', denoted Benchmark, and of the 'ex post trading strategy', denoted Ex post, for the testing dataset.

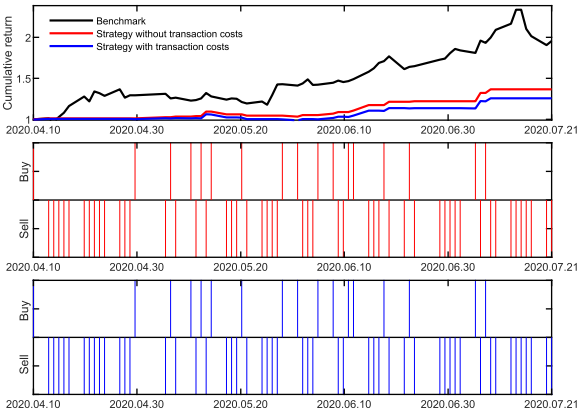
Model	*SR*	*PnL*	*MD*	*PnL*/*MD*	*TradeCount*	Trading Days
Panel A: without transaction costs						
Ex post	12.62	328.49	0	Inf	18	68
Benchmark	4.05	12.24	0.18	68.00	1	68
LR	5.52	6.66	**0.03**	222.00	14	68
SVM	5.69	16.69	0.14	119.21	4	68
GBDT	4.81	2.33	0.06	38.83	14	68
LSTM	6.42	18.78	0.06	**313.00**	10	68
LSTM-Attention	**6.61**	**26.96**	0.10	269.60	6	68
Panel B: with transaction costs						
Ex post	11.68	217.66	0	Inf	18	68
Benchmark	4.01	11.93	0.18	66.28	1	68
LR	4.70	4.47	**0.05**	89.40	14	68
SVM	5.13	8.94	0.09	99.33	7	68
GBDT	3.65	1.41	0.07	20.14	14	68
LSTM	**6.36**	11.72	0.06	**195.33**	12	68
LSTM-Attention	5.88	**16.05**	0.11	145.91	5	68



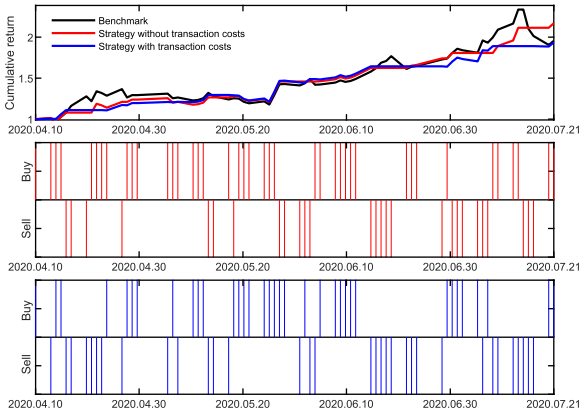
(a) LR prediction model



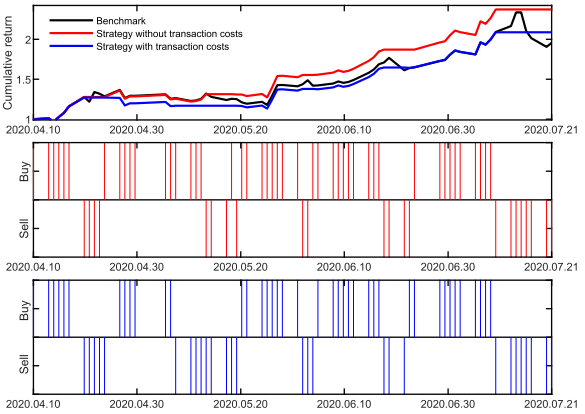
(b) SVM prediction model



(c) GBDT prediction model



(d) LSTM prediction model



(e) LSTM-Attention prediction model

Fig. 9. Time evolution of the trading performance and buy and sell signals using the LR, SVM, GBDT, LSTM, and LSTM-Attention prediction models. Top panels of each part: Time evolution of the trading performance in the testing period based on the predictions of individual models, for the strategy without transaction costs (red line), the strategy with transaction costs (blue line), and for comparison, the buy-and-hold strategy (benchmark strategy, black line). Middle panels: Buy and sell signals of the prediction models' trading strategies without transaction costs. Bottom panels: Buy and sell signals of the prediction models' trading strategies with transaction costs.

Table 9
Summary statistics of features derived from trading data, news data, and time–frequency data.

Source	Feature	Number	Mean	Std.	Min	25%	50%	75%	Max
Trading data	*open*	674	100.0351	49.3612	47.0000	61.2025	74.5500	144.0000	250.0000
	lag_open	674	100.0259	49.3478	47.0000	61.2025	74.5500	144.0000	250.0000
	high	674	102.5368	51.1283	49.4400	62.5150	76.2950	148.0825	261.9900
	low	674	97.9735	48.0490	46.5200	60.4925	73.0150	139.0100	238.0000
	close	674	100.1665	49.5963	48.1100	61.3075	74.5900	144.2725	257.0200
	prev_close	674	100.1845	49.6217	48.1100	61.3075	74.5900	144.2725	257.0200
	r	674	0.0480	3.3926	−10.0039	−1.7664	0.1190	1.6739	10.0065
	vol	674	$5.3187 * 10^6$	$4.4238 * 10^6$	$5.6910 * 10^5$	$2.5024 * 10^6$	$3.7325 * 10^6$	$6.8031 * 10^6$	$3.5532 * 10^7$
	avg	674	100.2719	49.5554	48.3543	61.3935	74.5721	143.6681	250.2272
	cap	674	$4.0077 * 10^{10}$	$1.9843 * 10^{10}$	$1.9249 * 10^{10}$	$2.4529 * 10^{10}$	$2.9843 * 10^{10}$	$5.7723 * 10^{10}$	$1.0283 * 10^{11}$
	share	674	$5.2027 * 10^8$	$5.1530 * 10^8$	$6.2851 * 10^7$	$1.9436 * 10^8$	$3.3042 * 10^8$	$6.5623 * 10^8$	$4.5992 * 10^9$
News data	*snow_avg*	674	0.6466	0.1838	0.0598	0.5000	0.6320	0.8056	0.9569
	snow_std	674	0.2208	0.1420	0.0000	0.1345	0.2638	0.3349	0.4745
	senta_avg	674	0.5002	0.1802	0.0626	0.3828	0.5000	0.6167	0.9430
	senta_std	674	0.2089	0.1287	0.0000	0.1239	0.2519	0.3071	0.4129
	*a*1	674	0.9106	2.1894	0	0	0	0.7688	18.1000
	*a*2	674	2.3299	3.8932	0	0.2591	0.9933	2.7375	32.2280
	*a*3	674	3.9086	5.1597	0	0.5178	1.9992	4.8761	36.3820
	*a*4	674	8.6015	11.2630	0.0004	0.8611	3.9426	10.2983	48.1040
	*a*5	674	14.8793	20.6166	0.0002	1.2688	4.8367	17.2045	68.9720
	*p*1	674	1.0837	1.7107	0	0	0	1.5708	4.7124
	*p*2	674	2.7827	1.7889	0	1.5708	1.5708	4.7124	4.7124
Time–frequency data	*p*3	674	3.0321	1.5856	0	1.5708	1.5708	4.7124	4.7124
	*p*4	674	3.0484	1.5692	1.5708	1.5708	1.5708	4.7124	4.7124
	*p*5	674	3.0717	1.5704	1.5708	1.5708	1.5708	4.7124	4.7124
	*c*1	674	0.2182	2.3614	−18.1000	0	0	0	16.9500
	*c*2	674	0.4006	4.5202	−12.5750	−1.0150	0	0.9438	32.2280
	*c*3	674	0.8625	6.4169	−33.1050	−1.4625	0.0531	2.4146	36.3820
	*c*4	674	−0.2290	14.1739	−48.1040	−4.3403	0.1215	3.4171	41.5710
	*c*5	674	−9.4688	23.6003	−68.9720	−15.6863	0.0389	3.4262	29.2630
	*c*6	674	108.3830	52.0642	57.2340	63.5660	73.2605	164.9075	212.4000

Table 10
Summary statistics of Alpha 101 technical indicators.

Feature	Count	Mean	Std	Min	25%	50%	75%	Max
alpha101_001	674	0.4903	0.2801	0.0761	0.2332	0.5726	0.8317	0.8317
alpha101_002	674	−0.1032	0.4681	−0.9301	−0.5027	−0.1298	0.2407	0.9520
alpha101_003	674	−0.2031	0.4235	−0.9159	−0.5350	−0.2568	0.0866	0.9113
alpha101_004	674	−5.0000	2.9679	−9.0000	−8.0000	−5.0000	−2.0000	−1.0000
alpha101_005	674	−0.2383	0.2331	−0.9986	−0.3308	−0.1449	−0.0725	−0.0014
alpha101_006	674	−0.1928	0.4217	−0.9581	−0.5280	−0.2456	0.1047	0.8588
alpha101_008	674	−0.5055	0.2822	−1.0000	−0.7475	−0.5077	−0.2665	−0.0014
alpha101_009	674	0.0984	4.2992	−22.0600	−1.2250	−0.0650	1.5175	25.7000
alpha101_010	674	0.1703	4.2970	−22.0600	−1.2250	0	1.5175	25.7000
alpha101_011	674	0.4842	0.3282	0.0021	0.2112	0.4405	0.6847	1.5865
alpha101_012	674	−0.6060	4.2574	−25.7000	−1.9875	−0.3600	0.7800	22.7800
alpha101_013	674	−0.5019	0.2918	−1.0000	−0.7569	−0.5111	−0.2431	−0.0014
alpha101_014	674	−0.0984	0.2518	−0.8778	−0.2466	−0.0655	0.0270	0.8040
alpha101_015	674	−1.5024	0.5887	−2.9433	−1.9125	−1.5041	−1.0975	−0.1438
alpha101_016	674	−0.5045	0.2892	−1.0000	−0.7555	−0.5069	−0.2611	−0.0014
alpha101_018	674	−0.4929	0.2834	−1.0000	−0.7315	−0.4965	−0.2462	−0.0014
alpha101_020	674	−0.1681	0.2042	−0.9835	−0.2261	−0.0929	−0.0256	0.0000
alpha101_021	674	0.1246	0.9929	−1.0000	−1.0000	1.0000	1.0000	1.0000
alpha101_022	674	−0.0071	0.3986	−1.5397	−0.1496	0.0009	0.1414	1.3134
alpha101_023	674	−1.0250	4.7441	−39.4100	−0.7625	0	0	36.9500
alpha101_024	674	−5.2481	10.2483	−38.9500	−10.4775	−4.3400	0	66.2100
alpha101_025	674	0.5018	0.2833	0.0028	0.2610	0.5156	0.7447	1.0000
alpha101_027	674	1.0000	0	1.0000	1.0000	1.0000	1.0000	1.0000
alpha101_029	674	3.3382	1.4471	1.0056	2.1426	3.2976	4.4937	5.9750
alpha101_030	674	0.1286	0.0794	0.0104	0.0713	0.1217	0.1768	0.5018
alpha101_033	674	0.5066	0.2873	0.0041	0.2555	0.5021	0.7569	1.0000
alpha101_034	674	0.5103	0.2873	0.0014	0.2628	0.5186	0.7579	1.0000
alpha101_038	674	−0.2380	0.2174	−0.9023	−0.3456	−0.1654	−0.0689	−0.0009
alpha101_040	674	−0.1874	0.2621	−0.8739	−0.3468	−0.1395	−0.0280	0.5930
alpha101_041	674	−0.0497	0.6592	−3.4969	−0.2506	−0.0219	0.1704	4.0343
alpha101_042	674	2.1348	3.8876	0.0014	0.5722	1.1578	1.9646	43.0000
alpha101_044	674	−0.4280	0.4949	−0.9984	−0.8268	−0.6227	−0.1033	0.9412
alpha101_046	674	0.1301	1.3917	−6.8900	−1.0000	1.0000	1.0000	19.6000
alpha101_047	674	0.6566	1.1763	−0.9893	−0.0473	0.3844	1.0322	11.4341
alpha101_049	674	0.0788	3.1977	−22.0600	−0.4800	1.0000	1.0000	22.7800
alpha101_050	674	−0.7495	0.2111	−0.9986	−0.9376	−0.7982	−0.6030	−0.0236
alpha101_051	674	0.0943	3.1974	−22.0600	−0.4425	1.0000	1.0000	22.7800
alpha101_053	674	275.9593	21514.6395	−238698.0000	−2.4702	0.0049	2.5857	238700.3418
alpha101_054	674	−0.4262	0.2606	−1.0000	−0.6431	−0.4401	−0.1988	0
alpha101_055	674	−0.2572	0.4937	−0.9841	−0.6738	−0.3605	0.1138	0.9852
alpha101_056	674	−0.2405	0.2196	−0.9285	−0.3718	−0.1661	−0.0683	−0.0005
alpha101_057	674	0.4911	6.7141	−62.9900	−1.0648	0.3137	1.8087	40.5214
alpha101_060	674	−0.0014	0.0017	−0.0052	−0.0026	−0.0016	−0.0001	0.0024
alpha101_062	674	−0.4852	0.5002	−1.0000	−1.0000	0	0	0
alpha101_064	674	−0.4407	0.4968	−1.0000	−1.0000	0	0	0
alpha101_065	674	−0.4392	0.4967	−1.0000	−1.0000	0	0	0
alpha101_068	674	0	0	0	0	0	0	0
alpha101_072	674	4.4331	25.2251	0.0057	0.4547	0.9736	1.8481	493.0000
alpha101_073	674	−9.1320	5.0477	−17.0000	−13.0000	−9.0000	−5.0000	−1.0000
alpha101_074	674	−0.4970	0.5004	−1.0000	−1.0000	0	0	0
alpha101_077	674	0.3226	0.2273	0.0014	0.1341	0.2848	0.4769	0.9862
alpha101_081	674	−0.5208	0.4999	−1.0000	−1.0000	−1.0000	0	0
alpha101_083	672	0.8673	11.0008	−70.2675	−1.6709	0.2267	3.5335	71.8665
alpha101_086	674	0	0	0	0	0	0	0
alpha101_088	674	0.5076	0.2868	0.0041	0.2583	0.5179	0.7555	1.0000
alpha101_092	674	4.1365	2.5270	1.0000	1.0000	4.0000	7.0000	7.0000
alpha101_096	674	−7.3242	1.5951	−13.0000	−8.0000	−7.0000	−7.0000	−1.0000
alpha101_098	674	−0.0070	0.4435	−0.9738	−0.3419	0.0186	0.3378	0.9159
alpha101_099	674	−0.5000	0.5004	−1.0000	−1.0000	−0.5000	0	0
alpha101_101	674	−0.0154	0.5445	−0.9999	−0.4810	0	0.4635	1.0000

data preprocessing techniques and tackle its learning capabilities for specific tasks, including forecasting the next day's price direction and the next day's closing price, and analyzing the benefits for a long/short trading strategy.

In terms of statistical accuracy and trading performance, compared with the LR, SVM, GBDT, and original LSTM models, the experimental results and in-depth analyses for BGI Genomics show that the attention enhanced LSTM model achieves remarkable improvements in prediction performance by multi-source heterogeneous information fusion, and models that

Table 11
Summary statistics of Alpha 191 technical indicators.

Feature	Count	Mean	Std	Min	25%	50%	75%	Max
alpha191_001	674	−0.1461	0.5081	−0.9890	−0.5766	−0.1890	0.2514	0.9464
alpha191_003	674	0.6696	12.2021	−68.5500	−4.6275	0.0050	4.7550	60.5400
alpha191_004	674	−0.1543	0.9888	−1.0000	−1.0000	−1.0000	1.0000	1.0000
alpha191_006	674	−0.4907	0.2500	−0.7510	−0.7510	−0.2510	−0.2510	−0.2510
alpha191_007	674	0.4842	0.3282	0.0021	0.2112	0.4405	0.6847	1.5865
alpha191_008	674	0.5200	0.2798	0.0014	0.2833	0.5250	0.7583	1.0000
alpha191_009	674	100.2593	49.3075	51.5595	61.2718	74.6724	146.0503	234.5470
alpha191_012	674	−0.2462	0.2343	−0.9875	−0.3390	−0.1685	−0.0681	−0.0001
alpha191_013	674	−0.0497	0.6592	−3.4969	−0.2506	−0.0219	0.1704	4.0343
alpha191_014	674	0.0001	10.0993	−62.7200	−3.9275	−0.1250	3.2725	46.5200
alpha191_015	674	−0.0008	0.0161	−0.1000	−0.0060	0	0.0055	0.1000
alpha191_016	674	−0.7495	0.2111	−0.9986	−0.9376	−0.7982	−0.6030	−0.0236
alpha191_018	674	1.0031	0.0797	0.7560	0.9574	0.9981	1.0416	1.3036
alpha191_019	674	−0.0001	0.0735	−0.2440	−0.0426	−0.0019	0.0400	0.2329
alpha191_020	674	0.3897	8.7536	−25.8970	−4.9171	−0.5164	4.4594	37.8756
alpha191_021	674	−0.0690	2.4703	−11.1087	−1.0000	−0.2720	1.1663	9.1747
alpha191_022	674	−98.9511	48.1224	−211.2363	−144	−74.6083	−60.4823	−54.9735
alpha191_023	674	51.8650	7.5712	30.1150	46.8357	52.2175	55.7542	79.5787
alpha191_024	674	0.0544	6.2956	−22.5832	−2.7434	−0.3185	2.2654	28.7783
alpha191_027	674	3.9642	62.3864	−153.0914	−33.9152	−6.9047	42.5038	192.7562
alpha191_028	674	46.0056	35.7541	−17.6066	12.8207	41.8047	77.5327	119.6743
alpha191_029	674	$1.9228 * 10^{5}$	$9.4215 * 10^{5}$	$-3.6381 * 10^{6}$	$-1.4198 * 10^{5}$	$-1.4679 * 10^{4}$	$1.9920 * 10^{5}$	$1.0173 * 10^{7}$
alpha191_031	674	0.1035	6.7481	−18.8284	−3.9601	−0.5696	3.6819	23.4269
alpha191_032	674	−2.0000	0.5887	−2.9433	−2.0000	−2.0000	−1.0000	−0.1438
alpha191_034	674	1.0034	0.0670	0.8102	0.9645	1.0057	1.0412	1.2320
alpha191_036	674	0.4968	0.2890	0.0014	0.2465	0.4951	0.7462	1.0000
alpha191_037	674	−216.8244	5068.2919	−23652.8933	−1701	35.9658	1789.3365	27750.8772
alpha191_038	674	−1.0000	4.7441	−39.4100	−0.7625	0	0	36.9500
alpha191_041	674	−0.4808	0.1678	−1.0000	−0.4162	−0.4162	−0.4162	−0.4162
alpha191_042	674	−0.1874	0.2621	−0.8739	−0.3468	−0.1395	−0.0280	0.5930
alpha191_043	674	$4.4103 * 10^{6}$	$1.7571 * 10^{7}$	$-7.6977 * 10^{7}$	$-3.8508 * 10^{6}$	$1.5229 * 10^{6}$	$1.0290 * 10^{7}$	$1.1420 * 10^{8}$
alpha191_047	674	21.1994	15.0507	−18.2237	9.7817	22.4605	33.2687	52.3685
alpha191_048	674	−0.1243	0.0834	−0.5074	−0.1697	−0.1145	−0.0588	−0.0061
alpha191_049	674	0.4986	0.2187	0.0536	0.3254	0.5138	0.6783	0.9289
alpha191_050	674	0.0028	0.4374	−0.8578	−0.3566	−0.0277	0.3492	0.8929
alpha191_051	674	0.5014	0.2187	0.0711	0.3217	0.4862	0.6746	0.9464
alpha191_053	674	52.3739	13.4198	16.6667	41.6667	50.0000	58.3333	91.6667
alpha191_054	674	−0.4849	0.2803	−1.0000	−0.7224	−0.4816	−0.2436	−0.0057
alpha191_055	674	−68.8866	455.1920	−1718.0416	−263.6958	−69.9731	118.9330	2016.6866
alpha191_057	674	46.1301	24.1279	5.5400	24.7380	42.9472	68.7471	92.6738
alpha191_058	674	52.4332	10.4298	25.0000	45.0000	55.0000	60.0000	85.0000
alpha191_059	674	2.7471	22.2861	−59.3400	−9.5325	1.2700	10.1375	111.3100
alpha191_062	674	−0.4280	0.4949	−0.9984	−0.8268	−0.6227	−0.1033	0.9412
alpha191_063	674	49.2838	19.2573	6.2514	34.9882	47.4137	63.1272	92.2590
alpha191_065	674	1.0017	0.0437	0.8517	0.9797	1.0011	1.0231	1.2011
alpha191_066	674	0.0219	4.3617	−16.7405	−2.2564	−0.1134	2.0674	17.4130
alpha191_067	674	49.9596	11.3874	25.3152	41.5746	48.5643	58.0425	78.7689
alpha191_068	674	$-4.9601 * 10^{-7}$	$3.0429 * 10^{-6}$	0	$-4.3910 * 10^{-7}$	$-6.3494 * 10^{-8}$	$1.5582 * 10^{-7}$	$1.7288 * 10^{-5}$

Table 11 (*continued*)

Feature	Count	Mean	Std	Min	25%	50%	75%	Max
alpha191_069	674	−0.3201	0.2244	−0.8354	−0.4851	−0.3351	−0.1517	0.2583
alpha191_070	674	$1.6154 * 10^8$	$1.6840 * 10^8$	$1.3939 * 10^7$	$5.7772 * 10^7$	$9.9917 * 10^7$	$1.9075 * 10^8$	$1.1817 * 10^9$
alpha191_072	674	53.2241	11.3734	26.7890	44.6575	52.2874	62.0824	79.2427
alpha191_076	674	0.4841	0.1535	0.2715	0.4182	0.4610	0.5140	1.6130
alpha191_079	674	49.6429	14.5054	14.6352	39.0670	48.7588	59.1861	85.2442
alpha191_080	674	19.5966	99.4696	−83.3537	−30.2964	−3.2805	31.7205	1200.5022
alpha191_081	674	$5.2474 * 10^6$	$3.3050 * 10^6$	$1.0482 * 10^6$	$2.9795 * 10^6$	$4.3135 * 10^6$	$6.0649 * 10^6$	$1.6267 * 10^7$
alpha191_082	674	53.0468	10.3792	24.2015	45.4165	52.3005	60.6852	76.0180
alpha191_083	674	−0.5045	0.2892	−1.0000	−0.7555	−0.5069	−0.2611	−0.0014
alpha191_084	674	$1.5024 * 10^7$	$3.2959 * 10^7$	$-5.6547 * 10^7$	$-4.6367 * 10^6$	$9.1038 * 10^6$	$2.4192 * 10^7$	$1.5472 * 10^8$
alpha191_086	674	0.1301	1.3917	−6.8900	−1.0000	1.0000	1.0000	19.6000
alpha191_088	674	1.6152	17.4095	−34.5442	−9.4947	−0.6739	11.9001	58.6935
alpha191_089	674	−0.0944	2.8494	−10.7743	−2.0000	0.0514	1.4320	10.8588
alpha191_090	674	−0.4981	0.2895	−0.9986	−0.7458	−0.4986	−0.2486	−0.0014
alpha191_093	674	18.9129	13.8422	2.7100	7.8750	15.2600	26.4000	67.8400
alpha191_095	674	$2.1494 * 10^8$	$1.8631 * 10^8$	$2.8660 * 10^7$	$8.2223 * 10^7$	$1.3558 * 10^8$	$3.0081 * 10^8$	$9.0899 * 10^8$
alpha191_096	674	46.3085	27.2698	5.1796	23.2278	39.2274	67.0904	112.4911
alpha191_097	674	$1.8590 * 10^6$	$1.6410 * 10^6$	$2.0596 * 10^5$	$7.8548 * 10^5$	$1.3031 * 10^6$	$2.3967 * 10^6$	$1.0228 * 10^7$
alpha191_098	674	−5.2481	10.2483	−38.9500	−10.0000	−4.3400	0	66.2100
alpha191_099	674	−0.5019	0.2918	−1.0000	−0.7569	−0.5111	−0.2431	−0.0014
alpha191_100	674	$2.1606 * 10^6$	$1.7282 * 10^6$	$3.0963 * 10^5$	$9.3818 * 10^5$	$1.6580 * 10^6$	$2.7716 * 10^6$	$9.9847 * 10^6$
alpha191_101	674	−0.4926	0.5003	−1.0000	−1.0000	0	0	0
alpha191_102	674	49.1942	10.5188	31.7694	41.5690	46.9827	54.4488	91.3050
alpha191_103	674	44.2285	35.2438	0	10.0000	40.0000	80.0000	95.0000
alpha191_104	674	−0.0071	0.3986	−1.5397	−0.1496	0.0009	0.1414	1.3134
alpha191_105	674	−0.2031	0.4235	−0.9159	−0.5350	−0.2568	0.0866	0.9113
alpha191_106	674	0.2569	19.8327	−62.4900	−9.0125	−0.4450	9.0675	95.0600
alpha191_107	674	−0.1681	0.2042	−0.9835	−0.2261	−0.0929	−0.0256	0.0000
alpha191_109	674	0.9971	0.1312	0.6941	0.9052	0.9790	1.0682	1.6409
alpha191_110	674	123.0288	63.2200	34.9001	79.7922	104.5444	148.5177	378.9331
alpha191_111	674	$1.0692 * 10^4$	$1.5747 * 10^6$	$-8.9833 * 10^6$	$-5.8243 * 10^5$	$2.0852 * 10^4$	$6.5441 * 10^5$	$8.4296 * 10^6$
alpha191_112	674	−2.0000	38.9669	−80.7871	−30.7496	−5.8300	31.4155	81.8079
alpha191_114	672	0.8673	11.0008	−70.2675	−2.0000	0.2267	3.5335	71.8665
alpha191_116	674	−0.1299	1.2192	−4.2179	−0.9192	−0.2821	0.5463	3.6647
alpha191_118	674	126.1728	49.3493	40.8141	88.3101	119.7050	155.0957	325.9664
alpha191_120	674	2.1348	3.8876	0.0014	0.5722	1.1578	1.9646	43.0000
alpha191_122	674	0.0000	0.0016	−0.0036	−0.0009	0.0000	0.0010	0.0053
alpha191_123	674	−0.5000	0.5004	−1.0000	−1.0000	−0.5000	0	0
alpha191_126	674	100.2256	49.5704	48.2833	61.3275	74.5467	143.1742	250.3000
alpha191_128	674	−20.3023	113.4759	−748.4376	−58.6368	15.9909	48.1163	94.4449
alpha191_129	674	−16.0000	13.7424	−81.3600	−20.0000	−11.0000	−6.2450	−1.0000
alpha191_132	674	$5.0590 * 10^8$	$3.8299 * 10^8$	$1.1369 * 10^8$	$2.1724 * 10^8$	$3.6020 * 10^8$	$7.0322 * 10^8$	$1.8497 * 10^9$
alpha191_133	674	−4.4955	62.7920	−95.0000	−65.0000	−20.0000	60.0000	95.0000
alpha191_134	674	$3.2286 * 10^5$	$1.3246 * 10^6$	$-2.9744 * 10^6$	$-2.1131 * 10^5$	$-3.7505 * 10^4$	$4.4652 * 10^5$	$1.2551 * 10^7$
alpha191_135	674	1.0579	0.2446	0.7656	0.9429	1.0030	1.1051	2.6757
alpha191_136	674	−0.0984	0.2518	−0.8778	−0.2466	−0.0655	0.0270	0.8040
alpha191_137	674	382.1074	1046.7995	1.2376	23.5833	78.6708	316.1064	12999.0600
alpha191_139	674	−0.1928	0.4217	−0.9581	−0.5280	−0.2456	0.1047	0.8588

(*continued on next page*)

Table 11 (*continued*)

Feature	Count	Mean	Std	Min	25%	50%	75%	Max
alpha191_144	674	0	0	0.0000	0	0	0	0
alpha191_148	674	−0.4377	0.4965	−1.0000	−1.0000	0	0	0
alpha191_150	674	$5.2009 * 10^{8}$	$5.1582 * 10^{8}$	$6.2851 * 10^{7}$	$1.9389 * 10^{8}$	$3.2979 * 10^{8}$	$6.5574 * 10^{8}$	$4.6288 * 10^{9}$
alpha191_151	674	1.2090	16.0095	−27.5168	−6.4558	−0.7298	6.7053	64.2788
alpha191_155	674	$4.4702 * 10^{3}$	$4.1572 * 10^{5}$	$-1.8198 * 10^{6}$	$-1.6728 * 10^{5}$	$-1.4131 * 10^{4}$	$1.2096 * 10^{5}$	$2.6387 * 10^{6}$
alpha191_156	674	−0.7181	0.1761	−1.0000	−0.8610	−0.7318	−0.6022	−0.1650
alpha191_157	674	3.5016	1.4254	1.0049	2.2632	3.4910	4.7184	5.9930
alpha191_158	674	2.4363	7.9814	−21.6317	−1.0000	0.9246	4.1404	52.4837
alpha191_160	674	3.0890	2.1346	0.6624	1.5520	2.2150	4.2154	9.5648
alpha191_161	674	4.8465	3.5734	0.9925	2.1642	3.5758	6.7469	19.3108
alpha191_162	674	−1.0000	0	−1.0000	−1.0000	−1.0000	−1.0000	−1.0000
alpha191_163	674	0.5018	0.2833	0.0028	0.2610	0.5156	0.7447	1.0000
alpha191_164	674	0.0030	0.0119	0	0.0000	0.0000	0.0008	0.1509
alpha191_167	674	15.7092	15.0387	1.3500	5.9025	9.9400	19.2700	97.5100
alpha191_168	674	−1.0000	0.5404	−7.2381	−1.0000	−0.8881	−0.6906	−0.3760
alpha191_170	674	−0.2523	0.3540	−0.9930	−0.4873	−0.2550	−0.0361	2.2602
alpha191_172	674	38.2084	21.2174	5.0479	19.8982	34.0559	54.7267	88.8475
alpha191_173	674	104.8184	51.7254	47.8337	65.5069	78.1889	144.6624	271.6095
alpha191_174	674	3.4200	2.5809	0.7963	1.4970	2.4375	4.3806	11.6159
alpha191_175	674	4.8385	3.7097	0.9050	2.1317	3.4008	6.7717	23.2633
alpha191_176	674	0.2572	0.4937	−0.9852	−0.1138	0.3605	0.6738	0.9841
alpha191_177	674	39.7329	34.7962	0	5.0000	30.0000	75.0000	95.0000
alpha191_178	674	$3.0366 * 10^{4}$	$3.4830 * 10^{5}$	$-2.5349 * 10^{6}$	$-5.1757 * 10^{4}$	$2.6616 * 10^{3}$	$6.1132 * 10^{4}$	$2.3445 * 10^{6}$
alpha191_180	672	$-2.4691 * 10^{6}$	$2.9358 * 10^{6}$	$-1.4949 * 10^{7}$	$-3.5245 * 10^{6}$	$-1.8683 * 10^{6}$	−40.7500	60.0000
alpha191_182	674	0.3447	0.1257	0.0500	0.2500	0.3500	0.4000	0.7000
alpha191_185	674	0.4974	0.2767	0.0028	0.2610	0.4986	0.7348	0.9821
alpha191_187	674	51.3542	43.3030	11.1100	21.5250	34.2750	60.9200	231.9500
alpha191_188	674	−2.0000	36.4273	−100.0000	−27.7794	−7.3642	17.2499	204.3086
alpha191_189	674	3.4204	3.3909	0.3789	1.2220	2.2408	4.2592	23.1947

include news-related features and time–frequency features often achieve the best performance metrics, indicating the relevance of online news and time–frequency features for prediction purposes and validating our proposed framework.

Due to limitations on paper length, this study only conducts experiments on data for BGI Genomics. Because numerical and textual data are the two main types processed in this paper, our approach can be applied to any company for which these two data types are available. In practical application, our framework also provides various options that can be applied for analyzing other stocks or assets, adopting other information fusion methods, developing other trading rules, or using other types of data from different sources.

CRediT authorship contribution statement

Qun Zhang: Resources, Formal analysis, Visualization, Writing - original draft, Writing - review & editing, Project administration. **Lijun Yang:** Methodology, Funding acquisition. **Feng Zhou:** Conceptualization, Data curation, Software, Investigation, Validation, Supervision.

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.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (11701144; 71801057; 11901113); the Natural Science Foundation of Guangdong Province (2018A030313968; 2019A1515011649); the Guangdong Youth Innovation Talent Project in China (natural sciences, 2018KQNCX086); the Project of Guangdong Province Innovative Team (2020WCXTD011); the Higher Education School Young Backbone Teacher Training Program of Henan Province (2017GGJS020) and Guangzhou Science and Technology Plan Project (201904010225, 202002030231).

References

[1] G. Armano, M. Marchesi, A. Murru, A hybrid genetic-neural architecture for stock indexes forecasting, Inf. Sci. 170 (1) (2005) 3–33.
[2] G.S. Atsalakis, K.P. Valavanis, Surveying stock market forecasting techniques–Part II: Soft computing methods, Expert Syst. Appl. 36 (3) (2009) 5932–5941.
[3] J. Bergstra, D. Yamins, D.D. Cox, Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures, in: 30th International Conference on Machine Learning, 2013, pp. 115–123.
[4] L. Bottou. Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier and G. Saporta, editors, Proceedings of COMPSTAT'2010, pages 177–186, Heidelberg, 2010. Physica-Verlag HD..
[5] Q. Cao, K.B. Leggio, M.J. Schniederjans, A comparison between Fama and French's model and artificial neural networks in predicting the Chinese stock market, Computers Operat. Res. 32 (10) (2005) 2499–2512.
[6] W.S. Chan, Stock price reaction to news and no-news: Drift and reversal after headlines, J. Financ. Econ. 70 (2) (2003) 223–260.
[7] K. Chen, Y. Zhou, F. Dai, A LSTM-based method for stock returns prediction: A case study of China stock market, in: Proceeding of 2015 IEEE International Conference on Big Data, 2015, pp. 2823–2824.
[8] C. Ding, X.F. He, H.Y. Zha, H.D. Simon, Adaptive dimension reduction for clustering high dimensional data, in: 2002 IEEE International Conference on Data Mining, 2002, pp. 147–154.
[9] J.C. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (7) (2011) 2121–2159.
[10] T. Fischer, C. Krauss, Deep learning with long short-term memory networks for financial market predictions, Eur. J. Oper. Res. 270 (2) (2018) 654–669.
[11] M.Z. Frank, A. Sanati, How does the stock market absorb shocks?, J Financ. Econ. 129 (1) (2018) 136–153.
[12] M.G. Frei, I. Osorio. Intrinsic time-scale decomposition: Time-frequency-energy analysis and real-time filtering of non-stationary signals. Proc. R. Soc. London A: Math., Phys. Eng. Sci., 463:321–342, 2007..
[13] J.D. Hamilton, Time series analysis, vol. 2, Princeton University Press, 1994.
[14] R. Hecht-Nielsen, Theory of the backpropagation neural network, in: H. Wechsler (Ed.), Neural Networks for Perception, Academic Press, 1992, pp. 65–93.
[15] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[16] N.E. Huang, Z. Shen, S.R. Long, M.C. Wu, H.H. Shih, Q. Zheng, N.C. Yen, C.C. Tung, H. . Liu. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 454 (1971), 903–995, 1998..
[17] G. Iovane, A. Amorosia, M. Leone, M. Nappi, G. Tortora, Multi indicator approach via mathematical inference for price dynamics in information fusion context, Inf. Sci. 373 (2016) 183–199.
[18] L. Itti, C. Koch, Computational modelling of visual attention, Nat. Rev. Neurosci. 2 (3) (2001) 194–203.
[19] S.R. Jammalamadaka, J. Qiu, N. Ning, Predicting a stock portfolio with the multivariate Bayesian structural time series model: Do news or emotions matter?, Int J. Artif. Intell. 17 (2) (2019) 81–104.
[20] I. Jolliffe, Principal component analysis, 2nd Edition., Springer-Verlag, New York, 2002.
[21] Z. Kakushadze, 101 formulaic alphas, Wilmott 2016 (84) (2016) 72–81.
[22] D.P. Kingma, J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014..
[23] B.S. Kumar, V. Ravi, A survey of the applications of text mining in financial domain, Knowl.-Based Syst. 114 (2016) 128–147.
[24] X. Lei, Y.C. Zhang, S. Cheng, F.X. Wu, W. Pedrycz, Topology potential based seed-growth method to identify protein complexes on dynamic PPI data, Inf. Sci. 425 (2018) 140–153.
[25] C. Li, F.B. Liu, Multi-factor stock picking system based on short-period price-volume characteristics, Guotai Junan Financial Eng. Special Report (in Chinese) (2017).
[26] Q. Li, T.J. Wang, P. Li, L. Liu, Q.X. Gong, Y.Z. Chen, The effect of news and public mood on stock movements, Inf. Sci. 278 (2014) 826–840.

[27] D.H. Ma, S.J. Li, X.D. Zhang, H.F. Wang, Interactive attention networks for aspect-level sentiment classification, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 4068–4074.
[28] Q.L. Ma, S. Tian, J. Wei, J.B. Wang, W.W. Ng, Attention-based spatio-temporal dependence learning network, Inf. Sci. 503 (2019) 92–108.
[29] R. Mihalcea, P. Tarau, Textrank: Bringing order into text, in: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 2004, pp. 404–411.
[30] W. Pedrycz, Allocation of information granularity in optimization and decision-making models: Towards building the foundations of granular computing, Eur. J. Oper. Res. 232 (1) (2014) 137–145.
[31] Pestov and Vladimir, Is the k-NN classifier in high dimensions affected by the curse of dimensionality?, Computers Math Appl. 65 (10) (2013) 1427–1437.
[32] C. Pozna, R.-E. Precup, Applications of signatures to expert systems modelling, Acta Polytechnica Hungarica 11 (2) (2014) 21–39.
[33] R.-E. Precup, T.-A. Teban, A. Albu, A.-B. Borlea, I.A. Zamfirache, E.M. Petriu, Evolving fuzzy models for prosthetic hand myoelectric-based control, IEEE Trans. Instrum. Meas. (2020).
[34] X.Y. Qian, S. Gao. Financial series prediction: Comparison between precision of time series models and machine learning methods. arXiv preprint arXiv:1706.00948, 2017..
[35] Y. Qin, D.J. Song, H.F. Cheng, W. Cheng, G.F. Jiang, G.W. Cottrell, A dual-stage attention-based recurrent neural network for time series prediction, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 2627–2633.
[36] J.M. Restrepo, S. Venkataramani, D. Comeau, H. Flaschka, Defining a trend for time series using the intrinsic time-scale decomposition, New J. Phys. 16 (8) (2014), 085004.
[37] A. Rico-Sulayes, Reducing vector space dimensionality in automatic classification for authorship attribution, Revista Científica de Ingeniería Electrónica, Automática y Comunicaciones 38 (3) (2017) 26–35.
[38] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[39] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.
[40] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks 61 (2015) 85–117.
[41] H. Tian, C. Gao, X. Xiao, H. Liu, B. He, H. Wu, H. Wang, F. Wu. SKEP: Sentiment knowledge enhanced pre-training for sentiment analysis. arXiv preprint arXiv:2005.05635, 2020..
[42] J. Wang, R. Hou, C. Wang, L. Shen, Improved v-support vector regression model based on variable selection and brain storm optimization for stock price forecasting, Appl. Soft Comput. 49 (2016) 164–178.
[43] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemometrics Intelligent Lab. Syst. 2 (1–3) (1987) 37–52.
[44] Z.Q. Xing, J.F. Qu, Y. Chai, Q. Tang, Y.M. Zhou, Gear fault diagnosis under variable conditions with intrinsic time-scale decomposition-singular value decomposition and support vector machine, J. Mech. Sci. Technol. 31 (2) (2017) 545–553.
[45] T. Xiong, Y.K. Bao, Z.Y. Hu, R. Chiong, Forecasting interval time series using a fully complex-valued RBF neural network with DPSO and PSO algorithms, Inf. Sci. 305 (2015) 77–92.
[46] L.J. Yang, S.J. Ding, H.M. Zhou, X.H. Yang, A strategy combining intrinsic time-scale decomposition and a feedforward neural network for automatic seizure detection, Physiol. Meas. 40 (9) (2019), 095004.
[47] F. Zhou, Q. Zhang, D. Sornette, L. Jiang, Cascading logistic regression onto gradient boosted decision trees for forecasting and trading stock indices, Appl. Soft Comput. 84 (2019), 105747.
[48] F. Zhou, H.M. Zhou, Z. Yang, L. Yang, EMD2FNN: A strategy combining empirical mode decomposition and factorization machine based neural network for stock market trend prediction, Expert Syst. Appl. 115 (2019) 136–151.
[49] D. Zill, W.S. Wright, M.R. Cullen, Advanced engineering mathematics, Jones & Bartlett Learning (2011).