Applied Soft Computing 114 (2022) 108106

Contents lists available at ScienceDirect

Applied Soft Computing

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

Stock market forecasting using a multi-task approach integrating long short-term memory and the random forest framework

Hyun Jun Park [a,1], Youngjun Kim [b,1], Ha Young Kim [b,*]

[a] Department of Data Science, Ajou University, Worldcupro 206, Yeongtong-gu, Suwon 16499, Republic of Korea
[b] Graduate School of Information, Yonsei University, Yonsei-ro 50, Seodaemun-gu, Seoul, 03722, Republic of Korea

ARTICLE INFO

Article history:
Received 8 April 2021
Received in revised form 10 October 2021
Accepted 16 October 2021
Available online 27 November 2021

Keywords:
Financial time-series forecasting
Hybrid deep learning
Multi-task learning
Explainable AI
Overfitting

ABSTRACT

Numerous studies have adopted deep learning (DL) in financial market forecasting models owing to its superior performance. The DL models require as many relevant input variables as possible to improve performance because they learn high-level features from these inputs. However, as the number of input variables increases, the number of parameters also increases, leaving the model prone to overfitting. We propose a novel stock-market prediction framework (LSTM–Forest) integrating long short-term memory and random forest (RF) to address this issue. We also develop a multi-task model that predicts stock market returns and classifies return directions to improve predictability and profitability. Our model is interpretable because it can identify key variables using the variable importance analysis from RF. We used three global stock indices and 43 financial technical indicators to verify the proposed methods experimentally. The root mean squared errors of LSTM–Forest with multi-task (LFM) in predicting returns from S&P500, SSE, and KOSPI200 were 25.53%, 22.75%, and 16.29% lower, respectively, than those of the baseline RF model. The model's balanced accuracy in forecasting the daily return direction increased by 7.37, 1.68, and 3.79 points, respectively. Furthermore, our multi-task model outperforms our single-task model and previous DL approaches. In the trading test, LFM produced the highest profits—even compared with the long-only strategy when transaction costs were considered. The proposed framework is readily extensible to other tasks and fields that contend with high-dimensionality problems.

1. Introduction

Accurate stock-market forecasting can be immensely helpful for market participants. However, such forecasting is complicated because stock markets are influenced by nonlinear relationships of various factors—economic variables, corporate policies, psychological characteristics of investors, and political events [1–3]. Various statistical and machine learning (ML) methods have been studied to reflect these market attributes [4–8]. Furthermore, researchers have explored the use of diverse market information such as technical indicators and news as model inputs [9]. Nevertheless, these approaches have limitations. Statistical methods require excessive assumptions, while traditional ML methods are vulnerable to learning complex patterns required in real trading. Moreover, selecting features that help market predictions and find new representations through their interrelations is demanding.

Deep learning (DL) approaches have been actively studied for financial time-series forecasting because of their high representational capacity and weak assumptions. They are typically based on a convolutional neural network (CNN), long short-term memory (LSTM), or an attention mechanism. CNN approaches transform time-series data into a 2D or 3D matrix to extract local and hierarchical patterns between variables to capture temporal features [10–12]. They require fewer parameters due to local connectivity and weight sharing. However, LSTM methods are more efficient than CNN methods for learning long-range temporal patterns due to recurrent and gate mechanisms. Therefore, LSTMs are more frequently used than CNNs in many financial time-series studies [13,14]. Recently, attention mechanisms have also been applied to time-series because they explicitly handle the long-term dependency problem [13,15]. However, they require heavier parameters than LSTM and are prone to overfitting. Consequently, LSTM is still the most dominant architecture in financial market forecasting. Huang et al. [16] recently used LSTM after separating long-term and short-term time-series with variational mode decomposition.

Despite considerable research, DL models for stock market forecasting still suffer from high-dimensionality problems. Most

* Corresponding author.
E-mail addresses: rainbow7782@ajou.ac.kr (H.J. Park), kht62@yonsei.ac.kr (Y. Kim), hayoung.kim@yonsei.ac.kr (H.Y. Kim).
[1] These authors contributed equally.

https://doi.org/10.1016/j.asoc.2021.108106
1568-4946/

models require numerous technical indicators containing diverse market information to learn the dynamic and complex patterns of stock markets [11,12]. However, this makes DL models susceptible to overfitting—a concern for daily forecasting owing to the lack of data available (at most 3000–5000 data points). In such a scenario, the number of parameters increases as the number of variables increases, which can lead to overfitting. Previous studies have tried to overcome this issue using a dimension reduction method, feature selection [12], or feature extraction [11,13,15,16]. For example, Hoseinzade and Haratizadeh [11] used principal component analysis (PCA), and Bao et al. [13] and Qiu et al. [16] used the wavelet transform. Nonetheless, dimension reduction can cause information loss, which is fatal to DL models that learn high-level features from input data [17]. Some studies have used data augmentation techniques [17,18], but even these are customized to their methods; thus, it is difficult to apply them in general. Therefore, for cases where real financial data are inherently lacking, a new DL method is required that can learn high-level temporal features without overfitting and self-identify the beneficial indicators and their relationships from raw data.

Another critical factor in improving performance is to determine the appropriate tasks for the DL model. Most existing DL models and algorithmic trading systems have been studied as a single task problem. These models define the problem either as a regression task that predicts stock returns or prices [13,15,16] or as a classification task that predicts return direction [6,19,20]. They are trained to minimize a loss function. Stock return prediction commonly uses the mean squared error (MSE) as a loss function. Predicting stock return directions usually utilizes the cross-entropy loss function. Predictions with the same distance to the actual return have the same MSE regardless of whether the return is negative or positive. Therefore, the prediction of the return sign in the regression can be less accurate than that in the classification. Furthermore, when classifying the return direction, the magnitude of the return can be overlooked. Consequently, precisely predicting both the sign and the magnitude of the return will improve stock market predictions and the profitability of trading systems. This can be achieved through multi-task learning (MTL) by jointly optimizing the losses of multiple tasks [21]. There are several MTL studies in stock market forecasting. MTL has been used primarily in tasks such as predictions of movement and volatility [22,23], correlated stock prices [24], and technical indicators [25]. However, thus far, there has been no study using both stock return prediction and stock movements prediction as MTL tasks.

Accordingly, the primary objective of this study is to enhance predictability and profitability in stock markets, and the motivations are summarized as follows. First, a novel DL framework is needed that can use the time-series of numerous technical indicators without overfitting. Second, we examine the effects of an MTL model using both the stock return-regression task and the stock return direction-classification task.

In this study, we propose a new LSTM–Forest framework. It integrates LSTM and a random forest (RF) by designing the classifiers or regressors with LSTMs instead of decision trees in the RF. An RF can handle thousands of input variables while avoiding overfitting [26]. In contrast, LSTM is superior to a decision tree for temporal patterns. Given the advantages of both methods, the overfitting problem can be reduced because each LSTM in the proposed model uses far fewer variables than the total. Furthermore, prediction performance can be improved because the model can learn high-level features that consider all the necessary technical indicators without information loss. Our model is explainable because it adopts an RF's ability to analyze variable importance. It identifies the principal technical indicators in stock market forecasting.

LSTM–Forest is an ensemble of multiple LSTMs that outputs the average of the predicted values of these LSTMs. The input variable set and the training data of each LSTM are randomly selected from the entire dataset and randomly sampled out of the entire training period, respectively. LSTM–Forest consists of low-correlated LSTMs. Thus, the proposed model is not biased to outliers or specific variables because it reflects numerous combinations of variables and data when compared with a single LSTM using equally many variables. Our model differs from previous studies that merely coupled RF and LSTM or compared the two approaches [20,27].

We also propose an LSTM–Forest with multi-task (LFM) model which consists of a stock return-regression task and a stock return direction-classification task. LFM replaces the single-task LSTMs of LSTM–Forest with multi-task LSTMs because the member model is responsible for training in the proposed approach. LSTM–Forest for a single task is called LFS to distinguish it from LFM. Furthermore, we develop an LFM-based trading system to validate the performance of our model during real trading.

We evaluate our framework by conducting empirical analyses on three real stock indices: the Standard & Poor's Index (S&P500), Shanghai Stock Exchange Composite Index (SSE), and Korean Composite Stock Price Index (KOSPI200). We consider 43 popular technical indicators used in previous studies [28–34]. Our contributions are summarized as follows:

- We propose a novel framework that integrates an RF framework and LSTM to prevent overfitting while using many technical indicators. Then, we tackle the issue of information loss due to dimension reduction methods.
- We develop an MTL model (LFM) with stock return prediction and direction classification as two separate tasks. Our model demonstrates superior predictability and profitability to those of the single-task model. LFM can learn features considering the magnitude as well as the sign of returns.
- We conduct extensive experiments on three global stock indices to evaluate the effectiveness of the proposed framework. Our model outperformed the baseline models and previous models.
- We identify vital indicators for return prediction and direction classification using the variable importance analysis of RF. The framework combining LSTM with an RF makes our model explainable.

The remainder of this study is organized as follows. Section 2 explains the background approaches of the proposed model, and Section 3 describes the stock data and methodology. Section 4 presents details of the experiments, followed by the verification and analysis of the results. Section 5 discusses the implications and limitations of our study. Finally, Section 6 presents concluding remarks and future research directions.

2. Background

2.1. Trading systems

Automated trading systems based on technical analysis can predict financial market directions by studying past market data (mostly stock prices and volume). Many traders use trading indicators and models for forecasting financial markets. Popular trading indicators include the simple moving average (SMA), moving average convergence divergence (MACD), relative strength index (RSI), and Bollinger band (BOLL). Their formulae are listed in Table 1. Earlier technical analyses were carried out by constructing trading systems that used these indicators [35–37]. Trading strategies were often developed by mixing several such indicators to increase trading profits [38,39].

Conversely, some studies have predicted the financial market using trading indicators directly as model inputs. Chen [29] proposed a method that used the MACD and RSI indicators as inputs into a neural network to classify the direction of the Taiwan 50 Index ETF and recommend transaction timings. Sezer and Ozbayoglu [10] proposed a CNN-based algorithm that classified stock directions. They used 15 technical indicators, including the RSI and Williams percentage range (Williams%R), and transformed them into 2D images to extract local and hierarchical patterns between indicators. Pair trading [40] and reinforcement learning models [41] are rule-based methods, whereas ML [42] and ensemble [43] models are feature-extraction methods.

2.2. Random Forest (RF)

RF is a combination of decision-tree classifiers (or regressors) where each tree depends on the independent random sample and has the same distribution for all trees in the forest [44]. In RF, not only the training data but also the input variables are randomly selected while generating each decision-tree classifier or decision-tree regressor [45]. Consequently, each decision tree of an RF becomes sufficiently robust to use thousands of variables without overfitting [26]. Furthermore, the variance and correlation of the trees can be reduced.

Mathematically, the RF can be explained in detail as follows. Three parameters must be determined in advance to construct the RF: number of trees (n), number of variables (K), and maximum depth (J) of the decision trees. The training sets (D_i, $i = 1, 2, \ldots, n$) and variable sets (V_i, $i = 1, 2, \ldots, n$) for the decision tree are created through random sampling with replacement, known as bootstrapping. The cardinality of each V_i is K. Each decision tree with depth J generates a weak learner τ_i from each D_i and V_i. Next, n weak learners are applied to the prediction of the test data and an ensemble of n trees $\{\tau_i\}_{i=1}^{n}$ is created. For a new sample x, the RF regressor can be described as $\hat{R}(x) = \frac{1}{n}\sum_{i=1}^{n}\hat{r}_i(x)$, where $\hat{r}_i(x)$ is a prediction of τ_i. The RF classifier model can be described as $\hat{C}(x) = \upsilon(\{\hat{c}_i(x)\}_{i=1}^{n})$, where $\hat{c}_i(x)$ is a class prediction of τ_i, and $\upsilon(\cdot)$ is a majority voting function.

2.3. Long Short-Term Memory (LSTM)

A recurrent neural network (RNN) has a feedback loop, ideal for learning temporal patterns and conducting a time-series analysis [45]. LSTM is a type of RNN, more effective than the basic RNN in analyzing time-series data because it can overcome vanishing gradient problems and effectively learn long-term dependence through memory cells and gates [46]. LSTM incorporates gating mechanisms to regulate the flow of information. The amount of incoming information that will be retained is systemically determined at each time step. Thus, LSTM can memorize temporal patterns over a longer period.

Fig. 1 illustrates the cell and three gate structures of a vanilla LSTM memory block. In LSTM, the cell state is responsible for the memory, which is controlled through gates. At time t, a forget gate (f_t), an input gate (i_t), an output gate (o_t), an input candidate (g_t), a cell state (c_t), and a hidden state (h_t) are calculated using the following formulae:

$$f_t = \sigma\left(W_{xh\_f}x_t + W_{hh\_f}h_{t-1} + b_{h\_f}\right) \quad (1)$$

$$i_t = \sigma\left(W_{xh\_i}x_t + W_{hh\_i}h_{t-1} + b_{h\_i}\right) \quad (2)$$

$$o_t = \sigma\left(W_{xh\_o}x_t + W_{hh\_o}h_{t-1} + b_{h\_o}\right) \quad (3)$$

$$g_t = \tanh\left(W_{xh\_g}x_t + W_{hh\_g}h_{t-1} + b_{h\_g}\right) \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (5)$$

$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

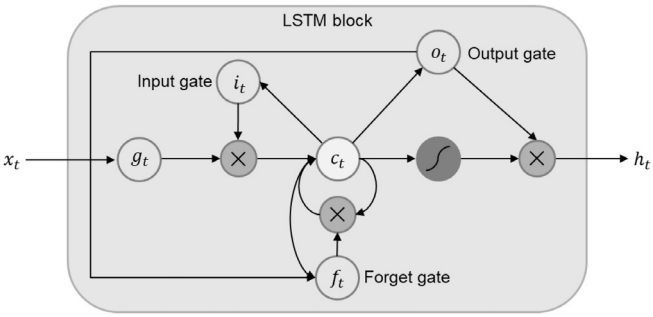


Fig. 1. Vanilla LSTM memory block architecture.

In these formulae, $\sigma(\cdot)$, $\tanh(\cdot)$, and $\odot$ denote the sigmoid function, hyperbolic tangent function, and element-wise multiplication, respectively. x_t is the input at time t. Ws are weight matrices and bs denotes the bias terms. First, f_t of Eq. (1) is calculated as the weighted sum of x_t, h_{t-1}, and the bias. It then has a value in the range of 0–1 through the sigmoid function, where 1 indicates that all information in the input passed, and 0 indicates the opposite. Thus, the forget gate controls the amount of information in the past cell state at time $t-1$, i.e., c_{t-1}, when updating the cell state at time t.

Similar to f_t, i_t and o_t are obtained by applying the sigmoid function to the weighted sum of x_t, h_{t-1}, and each bias, as depicted in Eqs. (2) and (3). Furthermore, i_t determines how much new information is stored in the cell state, and o_t determines how much the cell state value is to be released at time t. g_t calculates the new information to be stored in c_t and is obtained by calculating tanh on the weighted sum of x_t, h_{t-1} and bias as defined by Eq. (4). c_t is updated based on the sum of c_{t-1} and g_t controlled by f_t and i_t, respectively in Eq. (5). Finally, the output value h_t is determined by calculating the element-wise sum of o_t and tanh of c_t.

2.4. Multi-Task Learning (MTL)

MTL is a subfield of ML in which multiple tasks are simultaneously solved when exploiting commonalities and differences across tasks. The formal definition of MTL is as follows [21]. Given N related or partially related tasks $\{T_j\}_{j=1}^{N}$, MTL seeks to improve the training of task T_j using knowledge from all or some of the N tasks. The multi-task loss (L_{Multi}) is obtained from the combination of L_js, where L_j is the loss function corresponding to T_j. The combination method is usually a simple average or a weighted average. Then, the multi-task loss can be represented as $L_{multi} = J\{L_1(y,\hat{y}), L_2(y,\hat{y}), \ldots, L_N(y,\hat{y})\}$, where y and $\hat{y}$ are the target and prediction, respectively.

As depicted in Fig. 2, DL-based MTL is typically composed of shared and task-specific layers. The shared layers learn common features beneficial for all tasks, whereas the task-specific layers receive common features as inputs and learn task-specific features. Given that features are trained considering all tasks in shared layers, a regularization effect occurs, reducing overfitting and extracting more robust features. Consequently, MTL can improve learning efficiency and prediction accuracy over single-task learning [21], similar to human learning.

3. Methodology

3.1. Data description

We selected three global stock indices to analyze the proposed approach – S&P500, SSE, and KOSPI200 – representing three major stock markets with diverse patterns. We used 43 technical indicators, including five primary indicators (open, high, low, close

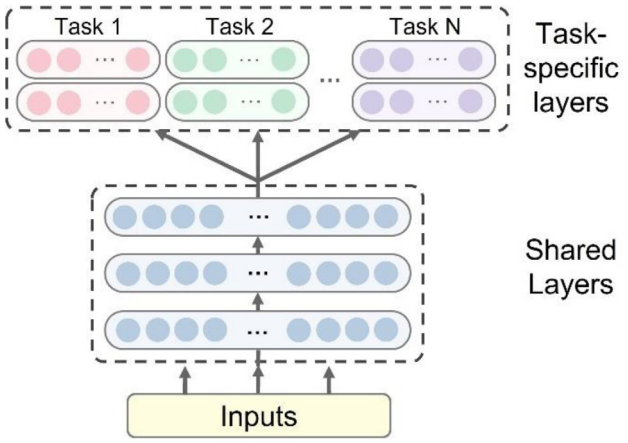


Fig. 2. Overview of MTL in a deep neural network.

prices, and total volume [OHLCV]) and 38 auxiliary indicators. Table 1 presents the descriptions and formulae of the indicators. These technical indicators are commonly included across previous published papers [28,30–34,47–50] when forecasting financial markets. We adopted the same indicator parameters used in previous studies.

Some of the technical indicators used are well-known trend indicators, including the SMA, exponential moving average (EMA) [48], and MACD [34]. RSI [28] is one of the most popular indicators used by traders to analyze stock markets, trading momentum, and on-balance volume (OBV) [31]. Developed by Joseph Granville, RSI is a trading-momentum indicator for measuring the positive and negative flow of volume. The commodity channel index (CCI) [34] and price rate of change (ROC) [33] are also helpful momentum indicators. Our indicator pool also contains volatility indicators such as the degree of variation in a trading price series, BOLL, and average true rate (ATR) [28], and market-strength indicators, such as the money flow index (MFI) [34], average ease of movement, and OBV. In addition, MACD, RSI, and OBV can capture short-term reversal.

Table 2 presents the training, validation, and test data of the stocks. All data were downloaded from https://finance.yahoo.com. The start dates differ for all indices to ensure that the maximum possible number of data points are used when modeling each index because this affects the model's representation capacity. We split our dataset into training, validation, and testing periods at a ratio of 7:1:2.

Fig. 3 presents the closing prices of three indices over the entire period considered, including the global financial crisis (2008–2009). Although the overall trends of the three indices differ considerably, common patterns emerge. All three indices, which had plunged after the dot-com bubble burst in 2001, rose again in 2007, just prior to the global financial crisis. They subsequently declined during the global financial crisis and increased until 2010. Since 2010, the three indices have exhibited different patterns. The S&P500 declined in 2011 when the US credit rating was demoted and then increased until 2015. It slightly declined in 2015 owing to China's devaluation of the Yuan and subsequently rose again. SSE declined after 2010 and increased starting in 2015; thereafter, it decreased again. KOSPI200 fluctuated after 2010 and increased to its highest level in 2017.

We performed min–max normalization to set variables within a range of 0–1 owing to the different scales of the 43 technical indicators. When forecasting financial markets, problems can be defined in various ways. In this study, we consider both return directions and returns of financial markets. Furthermore, logarithmic returns are used instead of returns to obtain stationary time-series data. The daily logarithmic return, r_t, is given by Eq. (7), where p_t and p_{t-1} denote stock closing prices at times t and $t-1$, respectively.

$$r_t = 100 * \ln \frac{p_t}{p_{t-1}} \tag{7}$$

The determination of daily return directions corresponds to the classification problem. The target value is $d_t \in \{0, 1\}$, where 0 indicates down and 1 indicates up. If the return at time t is greater than 0, it is coded as 1; otherwise, it is coded as 0.

Table 3 presents descriptive statistics for the logarithmic returns of each index, where N is the number of data points. The average logarithmic returns for the training and testing periods of all three indices are approximately 0. Only the SSE has a negative average return during the test period. The standard deviation of the logarithmic returns of the SSE is larger than those of the remaining indices during the test period. The augmented Dickey–Fuller (ADF) test in Table 3 is a test statistic to verify stationarity. The null hypothesis that the time-series is nonstationary is rejected because the ADF test statistic for logarithmic returns is less than the threshold of −3.46 (the critical value is 1%) for all the indices. Table 3 also summarizes the distribution of the direction classes. In the training and test periods, the rate of the Up class is slightly higher than that of the Down class.

3.2. Proposed framework: LSTM–Forest

This study aims to create a high-performance daily stock-market-forecasting model using many technical indicators and develop a more profitable trading system based on the prediction results. Most approaches designed for forecasting financial time-series are based on a CNN, LSTM, or an attention mechanism. However, CNN models are weak at learning long-range temporal patterns. Attention-based models are prone to overfitting due to the lack of data. Therefore, LSTM can still be the most suitable model for time-series modeling. Nonetheless, generic LSTM-based modeling is challenging to use because the daily number of available data points can be as small as 3000, and we consider a high-dimensional time-series of 43 technical indicators.

The primary advantage of the DL models is that high-level patterns are automatically extracted based on training data. More beneficial features can be learned with DL models than hand-crafted features using domain knowledge. The input of the DL model is crucial. The performance of DL models can improve with increasing usage of specific task-related variables. However, with insufficient data to learn neural networks using numerous variables, overfitting will occur because increasing the number of inputs raises the number of parameters.

Given that the DL model is a feature extractor, it is more optimal to adopt a method that can prevent overfitting when using all available variables rather than reducing the input dimensions and losing information. RF can prevent overfitting even if many variables are used. It is also an ensemble model, which is advantageous in terms of performance (Section 2.3). Hence, adopting an RF framework can be a suitable approach.

Accordingly, we propose a novel stock-market-prediction framework, LSTM–Forest, that uses LSTM instead of a decision tree in RF. The proposed model generates multiple LSTM models through random data sampling and random selection of variables. LSTM–Forest is an ensemble model with all trained LSTMs. Each weak LSTM model in this framework uses a small number of variables; however, all variables are used in LSTM–Forest.

Furthermore, the proposed method does not learn from a bias toward specific variables and data. Therefore, LSTM–Forest can avoid overfitting. Finally, this method learns correlated LSTM

Table 1
Descriptions of 43 technical indicators.

Name	Description	Formula	Study
O, H, L, C	Nominal daily open, highest, lowest, and closing price	O_t, H_t, L_t, C_t,	
Volume	Daily trading volume	V_t	
SD	Degree of variation in a trading-price series	$\sum_{t=1}^{n}\left(\frac{C_t-\overline{C_n}}{n}\right)^2$	[28]
UB & LB	Upper and lower Bollinger band	$\overline{C_{20}} \pm 2 \cdot SD_{20}$	[28]
FI	Force index	$(C_t - C_{t-1}) \cdot V_t$	[30]
FI5	Average force index	$\frac{1}{n} \cdot \sum_{t=1}^{n} FI_t$	[30]
C%, V%	Closing price and volume change	$\frac{C_t-C_{t-1}}{C_t}$, $\frac{V_t-V_{t-1}}{V_t}$	[34]
NVI	Cumulative negative volume index	$NVI_{t-1} + NV_t$	[28]
SEMV	Average ease of movement	$\frac{1}{n} \cdot \sum_{t=1}^{n} EMV_t$	[31,51]
TSI	True strength index	$\frac{DS_n}{DSA_n} \cdot 100$	[32]
MFR	Money-flow ratio	$\frac{\sum_{t=1}^{n} PMF_t}{\sum_{t=1}^{n} NMF_t}$	[34]
MFI	Money-flow index	$100 - \frac{100}{1+MFR_n}$	[34]
UVI & LVI	Upper and lower vortex indicator	$\frac{VM_+14}{ATR14}$ & $\frac{VM_-14}{ATR14}$	[34]
KST	Know sure thing	$\sum_{k=1}^{4} SROC \cdot \mathrm{k}$	[28,33]
KST9	Average know sure thing	$\frac{1}{n} \cdot \sum_{t=1}^{n} KST_t$	[28,33]
DPO20	Detrend price oscillator	$C_t - SMA_{t-11}$	[33]
DX	Directional index	$\frac{\lvert DM_+14-DM_-14\rvert}{DM_+14+DM_-14} \cdot 100$	[47]
ADX7, ADX14	Average directional index	$\frac{1}{n} \cdot \sum_{t=1}^{n} DX_t$	[47]
SMA(5,10,20)	Simple moving average	$\frac{1}{n} \cdot \sum_{t=1}^{n} C_t$	[31,32,47,51]
EMA(6,10,14)	Exponential moving average	$\frac{2}{n+1}\ (C_t - EMA_{t-1}) + EMA_{t-1}$	[31,32,51]
MACD(6,12)	Moving average convergence divergence	$EMA_\alpha - EMA_\beta$	[28,34]
RSI(10,14)	Relative strength index	$\frac{100 \cdot Gain_n}{Gain_n+Loss_n}$	[28,34,47]
CCI(20)	Commodity channel index	$\frac{TP-TP_n}{0.015 \cdot M.D_n}$	[34]
H–L	True range index (high–low)	$H_t - L_t$	[28]
H–Cp	True range index (high–previous close)	$\lvert H_t - C_{t-1}\rvert$	[28]
L–Cp	True range index (low–previous close)	$\lvert L_t - C_{t-1}\rvert$	[28]
TR	True range	$\max_i (TR_i)$	[28,32,34]
ATR14	Average true range	$\frac{1}{n} \cdot \sum_{t=1}^{n} TR_t$	[28,32,34]
ROC12	Price rate of change	$\frac{C_t-C_{t-n}}{C_{t-n}} \cdot 100$	[28,31,33]
Williams%R	Williams percentage range	$\frac{\max_t(H_t)}{\max_t(H_t)-\min_t(L_t)} \cdot -100$	[31]
OBV	On-balance volume	$OBV_{t-1} + (C_t - C_{t-1}) \cdot V_t$	[28,31,51]

Table 2
Data period.

Stock index	Train data		Validation		Test data	
	Period	N	Period	N	Period	N
S&P500	Aug 1, 2002–Apr 23, 2015	3204	1/8 of train data	400	Apr 24, 2015–Jun 28, 2018	802
SSE	Oct 13, 2003–Jul 21, 2015	2860	1/8 of train data	357	Jul 22, 2015–Jun 28, 2018	715
KOSPI200	Mar 4, 2002–May 18, 2015	3275	1/8 of train data	409	May 19, 2015–Sept 13, 2018	819

Table 3
Descriptive statistics and ADF tests for logarithmic returns.

Stock index		Logarithmic returns				Return directions				
		N	Mean	Standard deviation	ADF	Percentage (%)		Count		Mode
						Down	Up	Down	Up	Class
Train	S&P500	3204	0.027	1.25	−13.6*	45.3	54.7	1450	1754	Up
	SSE	2860	0.037	1.69	−11.5*	46.6	53.4	1334	1526	Up
	KOSPI200	3275	0.028	1.52	−10.8*	46.6	53.4	1525	1750	Up
Test	S&P500	802	0.031	0.82	−22.5*	46.8	53.2	375	427	Up
	SSE	715	−0.048	1.40	−7.4*	45.5	54.5	325	390	Up
	KOSPI200	819	0.015	0.84	−14.4*	48.2	51.8	395	424	Up

*Indicates that the null hypothesis is rejected at the 1% significance level.

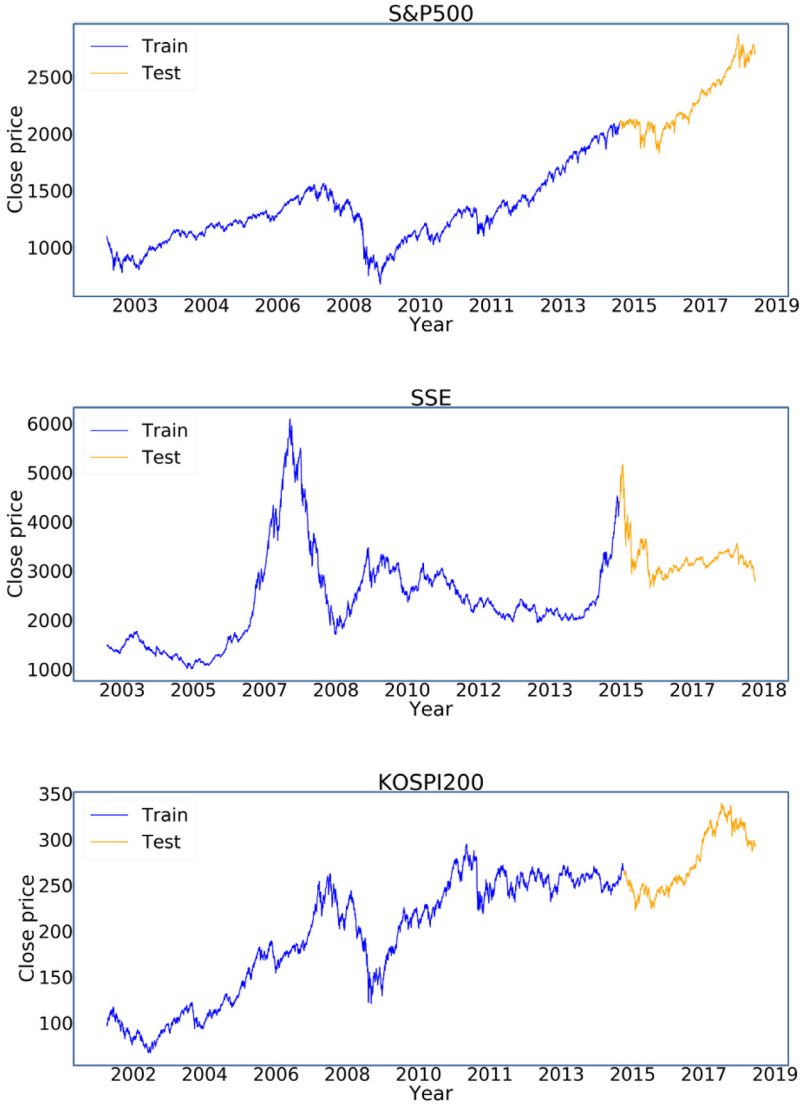


Fig. 3. Daily closing prices of three stock indices.

models through random sampling and variable random sampling, improving the performance of the ensemble results. Fig. 4 presents the LSTM–Forest framework.

The process of generating the proposed model is as follows. First, to construct the LSTM models that constitute the ensemble, training sets $(D_i, i = 1, 2, \ldots, n)$ for each LSTM are created through random sampling with replacement and K randomly selected technical indicator sets $(V_i, i = 1, 2, \ldots, n)$, similar to the RF. This results in low correlation of each LSTM through random sampling of data and variables. The variables differ for each LSTM, and the average number of times a variable is used during the entire learning period is $K \cdot n/v$ (where v = total number of variables).

Second, an LSTM model (C_i) is generated using D_i and V_i. The LSTM layer is connected to a fully connected (FC) layer, as depicted in Figs. 4 and 5. Features are extracted to predict the target value in an FC layer. Finally, the ensemble of n LSTM models is considered the final output of LSTM–Forest. The final predicted return is a simple average of predicted returns of the LSTM models. The final movement prediction, Up or Down binary classification based on a scale of 0.5, is also a simple average of predicted movements of LSTM models.

To the best of our knowledge, this is the first study combining LSTM and RF for stock-market prediction. Our proposed method differs from the frameworks in previous studies that merely join the prediction results or use LSTM and RF independently. Ma et al. [27] used the variable importance of RF to select important features from the Shanghai Composite Index as inputs for the LSTM model to predict stock trends. Ghosh et al. [20] independently generated LSTM and RF models for index return prediction and compared their performance. Both models in their studies differed from ours because they were not structurally combined.

3.3. LSTM–Forest: Single-task versus multi-task

MTL helps improve the learning of all tasks by exchanging knowledge from multiple related tasks (Section 2.4). Recent attempts have been made to apply MTL to predict the stock market. Li et al. [22] proposed a hierarchical multi-task RNN with two modules for stock price movement prediction. The first module predicted trends and volatility from market price data. The second module predicted stock movement by reflecting lead–lag relationships between stocks and the features extracted from the first module. Sawhney et al. [23] presented a multi-task ensemble model using multimodal data to forecast realized volatility

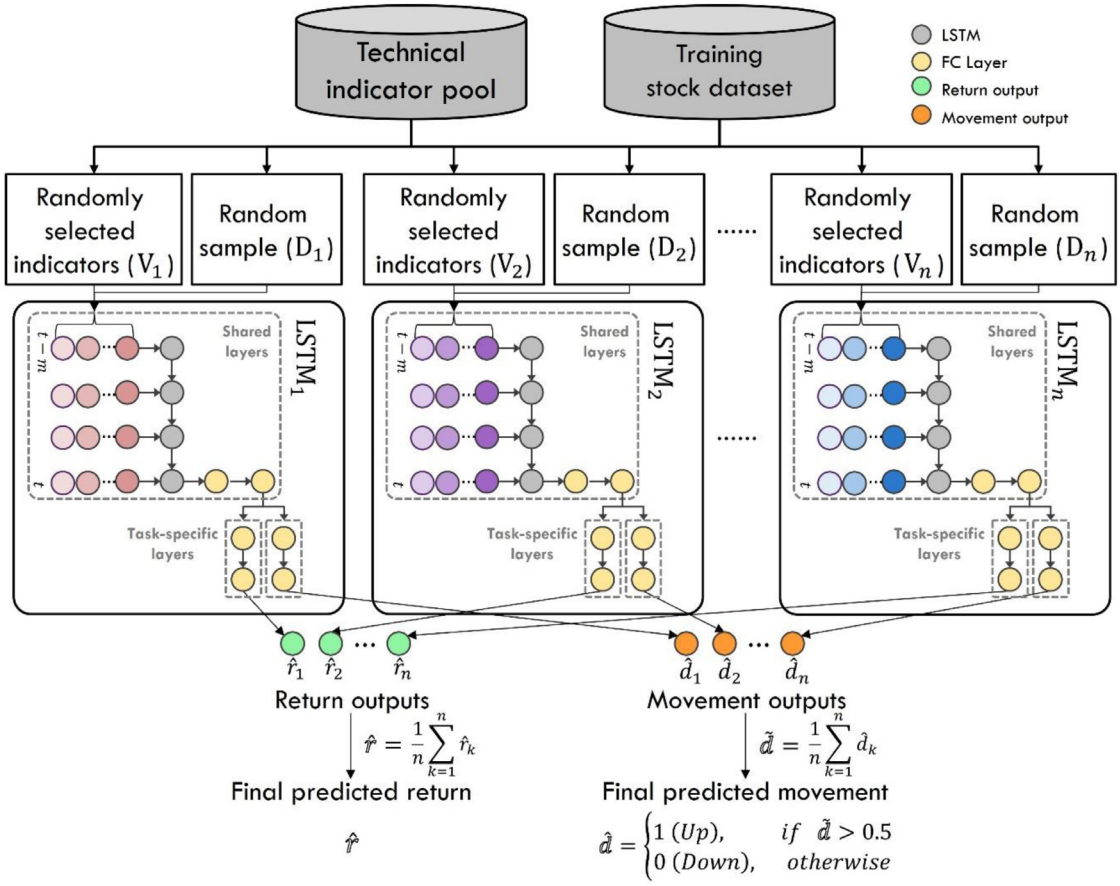


Fig. 4. Framework for LSTM–Forest with multi-task.

and stock price movement. The tasks of their study – volatility prediction and movement classification – are compared with our primary MTL model in Section 4.5.2.

Mootha et al. [25] developed a bidirectional LSTM-based sequence-to-sequence model using MTL to enable their model to learn the interrelationships among the open, high, close, and low prices of a stock. Ma and Tan [24] defined the price predictions of multiple stocks as MTL tasks to learn their inter-correlation and improve the prediction performance of each stock. The approaches of Mootha et al. and Ma and Tan are vulnerable to direction classification because they predict only the prices of stocks and can be inefficient for trading.

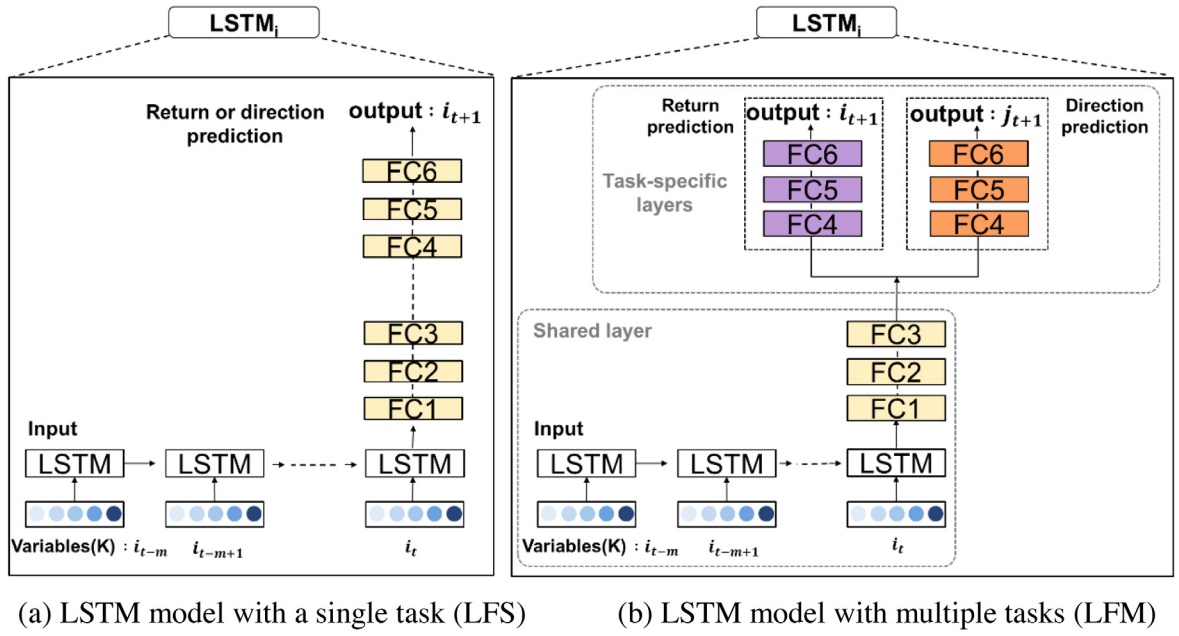
We extend LSTM–Forest to a multi-task approach – LFM – to improve the stock-market predictability and profitability of the trading system. We define stock-return predictions and stock market return-direction forecasts in this study as tasks in the proposed multi-task model because both have been actively researched in finance [2,6,52,53]. Furthermore, as explained earlier, these two related tasks are jointly trained to ensure that the model can learn considerably robust features due to its regularization effects—improving both return prediction and direction classification because shared patterns can be learned differently.

Stock return-direction forecasting is highly correlated to market return-direction forecasting. In the financial context, it may be argued that there is an insignificant difference between them; however, both tasks differ in terms of DL-based modeling. In supervised learning, the performance of DL algorithms is heavily dependent on the loss function used for a particular problem. Let us consider an example.

Assume that the model only learns returns. If the target is 1, the model's predicted values of 3.5 and −1.5 have the same MSE. This problem is critical in situations (e.g., stock trading) where return directions become relevant. Therefore, if a positive return direction is provided, the model can be informed that the return value must be positive, naturally predicting the return as a positive number. Similarly, we assume that the model predicts only the return directions. When the label is the Up class, errors in training examples with a return of 0.5 and an example with a return of 3.5 are identical, despite different magnitudes. However, in trading, it is crucial to predict large returns accurately. If only return-direction classification is considered, the model cannot learn important examples effectively. However, this can be improved if the return-prediction loss is given. Therefore, in this study, we can make more precise predictions of the magnitude and the sign of the returns.

As depicted in Figs. 4 and 5, because the features of LSTM–Forest are learned through each member of the ensemble, we design the LFM as an ensemble of multi-task LSTM models. In the multi-task LSTM model, the shared layers consist of an LSTM layer and multiple FC layers, enabling it to learn common temporal features that benefit both tasks. Task-specific layers are composed of FC layers and learn features related to each task. Similarly, LFS is created as an ensemble of single-task LSTM models. LFS can consider two situations to predict returns or market-return directions. The LFS architecture is identical to LFM, with one of the task-specific layers removed.

We use MSE in the return-prediction task and cross-entropy in the return-direction classification task. LFM is trained using a multi-loss function by summing the MSE and cross-entropy. The multi-loss function enables LFM to learn features from the return-prediction and direction-classification tasks differently. Therefore, LFM can learn sign information more precisely in two ways,



(a) LSTM model with a single task (LFS) (b) LSTM model with multiple tasks (LFM)

Fig. 5. Structure of LSTM model with LSTM–Forest.

where sign information is included redundantly in both tasks. On a single task, two models must be used to predict the returns and return directions. However, learning a model using MTL is highly effective when solving similar tasks with the same data.

The detailed architecture of the LSTM model in the LFS and LFM is as follows. As depicted in Fig. 5(a), in LFS, the input (i.e., K technical indicators with 50 time-steps) is connected to the LSTM layer with 15 neurons. Next are five FC layers with 30 neurons. The FC5 layer is connected to the final output layer (FC6). A rectified linear unit is used as the activation function for all FC layers except FC6, which has one neuron, and the linear activation function is used for return prediction. In contrast, FC6 has two neurons, and the softmax activation function is used for direction prediction.

In Fig. 5(b), the architecture of LFM is a structure that adds a task-specific branch (task-specific layers)—with three FC layers of 30 neurons (i.e., the two task-specific branches have the same structure)—to that of LFS. A dropout (30%) is applied to the FC3 and FC5 layers in both LFS and LFM. The number of LSTM models is $N_{LSTM} = 100$. Of the total training data, 80% is selected when random sampling D_i.

The initial conditions that affect the forecasting results of our model are as follows: parameters of technical indicators ($\{\theta_i\}_{i=1}^{N_{TI}}$), number of input variables of each LSTM of LFM (K), steps of time-window (S), number of LSTMs of LFM (N_{LSTM}), set of initial weights of LSTMs (W), set of initial biases of LSTMs (B), mini-batch size (μ), and the parameters of the Adam optimizer: learning rate (L), momentum decay rate (β_1), and adaptive term decay rate (β_2).

We adopted the commonly applied technical indicator parameters from previous studies (Section 3.1). We define K for the random input variable selection, an integer value between 2 and 43 (Section 4.2). The other initial conditions, which are the hyperparameters for training, are selected through a grid search. Consequently, $S = 50$, $N_{LSTM} = 100$, and $\mu = 256$. Finally, we initialized the set of weights of LSTMs (W) using Xavier initialization [45] and the set of initial biases of LSTMs (B) to 0.

Furthermore, we used the default parameters of Keras [54] for the Adam optimizer [55]. The learning rate was reduced by 1/2 every 50 epochs for a total of 300 epochs. The LFM and LFS models both started to converge from 250 epochs, and overfitting tended to appear over 400–500 epochs. Therefore, all LSTM models were trained up to 300 epochs and tested. Our LFM method is described in detail in Algorithm 1. The corresponding code is available at https://www.github.com/parkhyunjun77/test_code.

Algorithm 1 is executed in two steps. In the first step, we generate commonly used technical indicators, 43 from five primary indicators (OHLCV) and parameters of technical indicators (θ_i). We then obtain normalized technical indicators I through min–max normalization. In the second step, for each LSTM of the LFM, we initialize W_i and B_i, randomly sampled training data D_i, and randomly selected technical indicators V_i using K. We trained $LSTM_i$ by updating W_i and B_i using the Adam optimizer. All LSTM models were trained up to the total epoch E, which is the stop criterion. The predictions of the last epoch for each LSTM are saved. Finally, the algorithm outputs an LFM in which all LSTMs are ensembled as described in Section 3.2.

4. Experiments and results

The proposed model predicts the next day's stock return and return direction using 43 technical indicators with 50 time-steps. In Section 4.1, we describe the metrics used to evaluate forecasting performance. In Section 4.2, we explore the number of variables (K) needed to optimize LSTM–Forest (LFS and LFM) and RF. In Section 4.3, we compare the LSTM–Forest models with the baseline models. In Section 4.4, LFM is compared with various existing stock-market-forecasting methods. In Section 4.5, we present the results of a trading test conducted to verify whether the LFM predictions can be used in real market trading.

Furthermore, we analyze an LFM variant with an absolute return prediction task and a return-direction prediction task to evaluate the suitability of the considered tasks for MTL. In Section 4.6, we conduct statistical tests to verify the robustness of our LFM model and identify the important technical indicators for forecasting returns and return directions. In Section 4.6.2, we generate LSTM models using only selected variables and compare them with LFM to test our hypothesis that using many available technical indicators rather than several selected technical indicators is more beneficial for DL-based prediction models.

Algorithm 1 Stock market forecasting using LSTM–Forest with multi-task (LFM).

Step 1. Generating training dataset by calculating technical indicators

Input: data X ▶ prices (open O, high H, low L, close C, volatility V)
target Y ▶ logarithmic return r, direction d

Output: train dataset D_{train}

$N_{TI} \leftarrow$ number of technical indicators
$\{\theta_i\}_{i=1}^{N_{TI}} \leftarrow$ each parameter of technical indicators
$I \leftarrow$ empty array for stacking N_{TI} technical indicators
for $i = 1, \cdots, N_{TI}$ **do**
  $Indicator_i \leftarrow Normalize\big(Formula_i(X, \theta_i)\big)$ ▶ as presented in **Table 1**
  $I \leftarrow Stack(Indicator_i)$
end for
$D_{train} \leftarrow Merge(I, Y)$

Step 2. Training LSTMs under the RF framework and generating the LFM model

Input: train dataset D_{train}

Output: LSTM-Forest with MTL model LFM

$N_{LSTM} \leftarrow$ number of LSTMs in LFM
$I_{list} \leftarrow$ list of total technical indicators
$K \leftarrow$ number of technical indicators of each LSTM
$S \leftarrow$ time-window size
$E \leftarrow$ number of total epochs ▶ stop criterion
$\mu \leftarrow$ mini-batch size
$L, \beta_1, \beta_2 \leftarrow$ learning rate, momentum decay rate, and adaptive term decay rate
for $i = 1, \cdots, N_{LSTM}$ **do**
  $W_i, B_i \leftarrow$ initialized weight and bias
  $V_i \leftarrow Sample(I_{list}, K)$ ▶ randomly selected technical indicators
  $D_i \leftarrow Sample(D_{train}, S)$ ▶ randomly selected train input data
  for $e = 1, \cdots, E$ **do**
    $\hat{r}_e, \hat{d}_e \leftarrow LSTM_i(V_i, D_i, \mu)$ ▶ predicted return and direction of each LSTM
    $Loss_e^r \leftarrow MSE(r, \hat{r}_e)$ ▶ r: target logarithmic return in D_i
    $Loss_e^d \leftarrow Crossentropy(d, \hat{d}_e)$ ▶ d: target direction in D_i
    $W_i, B_i \leftarrow Update\big(Optimizer(W_i, B_i, Loss_e^r, Loss_e^d, L, \beta_1, \beta_2)\big)$
  end for
  $LSTM_i$ ▶ trained i^{th} $LSTM$ model
end for
$LFM \leftarrow Ensemble\big(LSTM_1, LSTM_2, \cdots, LSTM_{N_{LSTM}}\big)$ ▶ ensemble $LSTM$ models

4.1. Metrics

The model was evaluated using several metrics to verify its robustness. Root mean squared error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE) are used to evaluate the regression problem. Accuracy (ACC) and balanced accuracy (BACC) are used to evaluate the classification problem. There are two primary classification accuracy measures to manage skewed data: BACC and F1-score. The BACC considers both positive and negative predictions, in contrast to the F1-score that only focuses on positive predictions [56]. Therefore, we used BACC because our dataset is slightly imbalanced. The formulae of the metrics are as follows:

$$RMSE = \sqrt{\left(\frac{1}{N}\sum_{t=1}^{N}(A_t - P_t)^2\right)}, \tag{8}$$

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|A_t - P_t|, \tag{9}$$

$$MAPE = \frac{1}{N}\sum_{t=1}^{N}\left|1 - \frac{P_t}{A_t}\right|, \tag{10}$$

$$ACC = \frac{correct\ number\ of\ predictions}{total\ number\ of\ predictions}, \tag{11}$$

$$BACC = \frac{1}{2}(sensitivity + specificity), \tag{12}$$

where A_t and P_t are the actual and predicted values, respectively, at time t, and N is the number of observed data points. RMSE is sensitive to outlier data. The index value becomes large if the outlier is not predicted. MAE is less sensitive to outliers than RMSE because MAE considers only the differences. MAPE can be objectively compared when there is a difference in the data scale. ACC is the rate of correct classifications. BACC is calculated as the average of sensitivity and specificity.

4.2. Optimal number of variables for LSTM–Forest

Determining the optimal number of variables for each LSTM model in LSTM–Forest is crucial because the proposed model adopts an RF framework. Therefore, we evaluate our proposed models using various values of K {K = 2 (5%), 4 (10%), 9 (20%), 13 (30%), 22 (50%), and 43 (100%)}, where K is the number of input variables that are randomly selected with replacement for a component model of LSTM–Forest. The numbers in parentheses indicate the percentages of the number of variables in each case. The training data for member models are also randomly selected with replacement from 80% of the total training data, and the validation data are used to determine the hyperparameters.

We then determine the optimal K by comparing the experimental results. We also compare our LFM with RF and LFS using various K to evaluate the model performance according to the number of variables. Fig. 6 presents the experimental results of RF, LFS, and LFM by varying the number of variables and the results of two tasks. The results of RF and LFS are obtained

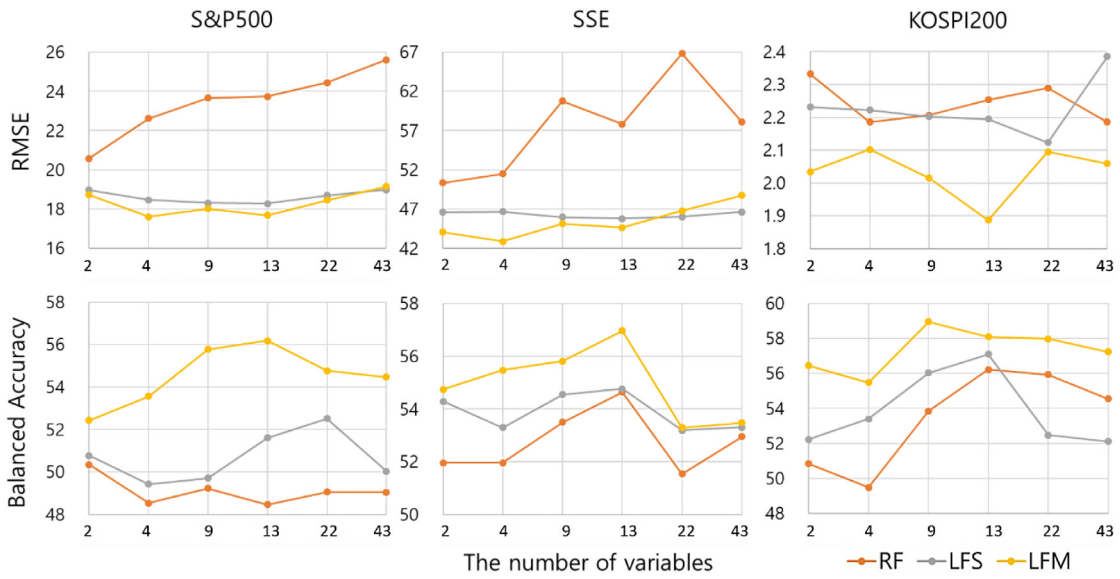


Fig. 6. Model performance comparison by number of variables.

through individual models for each task, whereas the results of LFM are acquired through a single model for both tasks.

The optimal number of variables for an RF is $\sqrt{N}$, $\log_2 N + 1$, or $N/3$ [44]. More than half of all variables should not be used because considerable correlations would be observed between the models, causing performance degradation [44]. Owing to the novelty of the proposed approach, which is not identical to the RF framework, we select an optimal value of K based on experiments within the recommended range. Our results indicate that K values between 4 and 13 exhibit superior performance, degrading when K exceeds 22. We select 13 as the optimal K value because many models deliver the best performance when $K = 13$.

The RMSE of LSTM–Forest (LFS and LFM) for all three indices and all K values (numbers of variables), except in two cases, is considerably smaller than that of the baseline RF model when comparing the stock index-return predictions in Fig. 6. The RMSE of the LFM is smaller than that of the LFS in all return prediction experiments, except in three cases. Furthermore, the classification results for the return direction of stock prices exhibit the same tendency as the results of the return-prediction. The BACC of LFS is 0.4–1.6 points higher than that of the RF model, except in two cases. The BACC of LFM is 2.2–5.4 points higher than that of the RF model. The BACC of LFM in all classification experiments is higher than that of LFS.

However, LFM spends more time than LFS on training and inference. On a GeForce GTX 1080 Ti GPU, the required training times to learn one LSTM of LFS and LFM (including data sampling, model training, and prediction) were 220 and 240 s, respectively. Complete learning for the entire framework required 6.2 h for LFS and 6.7 h for LFM. The inference times in one LSTM of the LFS and LFM were 48 and 49 ms, respectively.

4.3. Comparison with baseline models

We verify the robustness of the proposed LFM with the optimal K value. The return-prediction and return direction-prediction results are presented in Tables 4 and 5, respectively. We define RF and LSTM_{base} as the baseline models to analyze the return–forecasting performance. Long-only, RF, and LSTM_{base} are the baseline models for evaluating the return direction-prediction performance. LSTM_{base} is a model that uses all 43 variables as inputs without feature selection. Long-only predicts all classes as "+1" (i.e., the Up class).

Table 4
Comparison of return predictions for three stock indices.

Method	Metric	S&P500	SSE	KOSPI200
RF	MAE	16.06	46.65	1.72
	RMSE	23.74	57.81	2.25
	MAPE	0.70	1.45	0.63
LSTM_{base}	MAE	15.52	29.24	1.79
	RMSE	20.45	47.98	2.38
	MAPE	0.68	0.91	0.65
LFS	MAE	12.34	27.14	1.67
	RMSE	18.28	45.82	2.20
	MAPE	0.55	0.85	0.61
LFM	MAE	11.51	26.78	1.41
	RMSE	17.68	44.66	1.89
	MAPE	0.51	0.83	0.51

Table 5
Comparison of return-direction predictions for three stock indices.

Method	Metric	S&P500	SSE	KOSPI200
Long-only	ACC	53.24	54.54	51.77
	BACC	50	50	50
RF	ACC	48.37	53.84	56.04
	BACC	49.68	54.145	56.06
LSTM_{base}	ACC	45.76	47.55	49.08
	BACC	47.58	49.92	48.98
LFS	ACC	51.12	53.29	56.65
	BACC	51.48	50.15	57.82
LFM	ACC	55.74	55.52	59.83
	BACC	56.19	56.02	59.95

LFM delivers the best return-prediction performance for all three indices. The RMSEs of LFM are 21.52%, 13.75%, and 6.63% lower than those of RF, LSTM_{base}, and LFS, respectively. MAE and MAPE of LFM are also lower than those of other models. Similarly, in the return direction-prediction, LFM outperforms the baseline models and LFS. The ACCs (BACCs) of LFM are 3.84 (7.38), 4.20 (4.09), 9.56 (8.56), and 3.34 (4.23) points higher than those of long-only, RF, LSTM_{base}, and LFS, respectively.

The overall directions of the stock returns exhibit upward-sloping patterns. Our data indicate high rates in the Up class for all three indices during the training and test periods. Therefore, long-only strategies are advantageous in terms of classification.

Table 6
RMSE performance comparison between previous studies and proposed LFM model.

Method	Index	Data period	Previous studies	LFM
PCA–STNN [57]	S&P500	2006.08.04–2012.08.31	19.247	14.152
FWSVM-FWKNN [48]	SSE	2012.12.06–2014.12.31	0.0143	0.0130
EMD2FNN [58]	S&P500	2007.01.03–2011.12.30	17.659	16.079
WLSTM+Attention [16]	S&P500	2000.01.03–2019.07.01	0.2337	0.2128
CNN-LSTM [59]	SSE	2013.06.01–2020.04.30	0.0449	0.0415
Elman-DIOCs [60]	KOSPI	2005.12.13–2013.12.30	183.61	178.25

Only LFM outperforms long-only in terms of both ACC and BACC. Based on these results, the hybrid model of RF and LSTM outperformed the single model. Moreover, MTL using the tasks that we defined is more effective than single-task learning.

4.4. Comparison with previous studies

We evaluated the superiority of our proposed model by comparing LFM with the latest approaches listed in Table 6. Wang and Wang [57] proposed the PCA–stochastic time effective neural network (STNN) to predict stock prices. They used PCA to reduce the number of dimensions and subsequently as an input to the STNN. STNNs can reflect the characteristics of dynamic and nonstationary stock time-series more accurately than standard backpropagation using the stochastic time effective function, which weights the loss according to the past data time. FWSVM-FWKNN is a stock index forecasting framework combining a feature-weighted support vector machine and feature-weighted k-nearest neighbor based on information gain [48].

Empirical mode decomposition (EMD) and factorization machine (FM)-based neural network (EMD2FNN) is a two-step model [58]. First, the raw time-series is decomposed into intrinsic mode functions (IMFs) using the EMD. Then, the IMFs are fed to the FM-based neural network. WLSTM+Attention is an attention-mechanism-based LSTM model. It uses wavelet transformation for data denoising and assigns feature weights using a common attention mechanism [16]. Jing et al. [59] developed a hybrid model integrating a CNN sentiment classifier and an LSTM time-series prediction model. They used 26 technical indicators and text from an online social network as input data. Wang et al. [60] proposed a variant of the Elman neural network with direct input-to-output connections (Elman-DIOCs) that outperformed the existing Elman neural network.

The results are presented in Table 6. The values in the "Previous studies" column are adopted from the indicated papers. The values in the "LFM" column are the results of the LFM experiments for the same test period. LFM outperforms all previous models. When compared with PCA-STNN and FWSVM-FWKNN using the dimension reduction method, the RMSEs of LFM decrease by 26.47% and 9.09%, respectively. Furthermore, LFM surpasses the recent attention-based model WLSTM+Attention by 8.94%. Our proposed model even outperforms CNN-LSTM, which uses stock forum text and 26 technical indicators. Although it adopts a hybrid architecture and multimodal inputs, our proposed model achieves a 7.57% performance improvement.

4.5. Trading simulations

4.5.1. LFM versus baseline models

We analyze the classification performance of the proposed model and verify through a trading simulation that it can be applied in the actual stock market. The trading simulation is performed based on the RF, LSTM_{base}, LFS, and LFM results and compared based on cumulative profits. It is also compared with the traditional long-only strategy. The value of $\dot{c}_i(t)$ is estimated based on the model's classification output at time t. This value selects the class with the highest probability among C_{down} and C_{up}. The simulation process according to the trading strategy is presented in Table 7. If $\dot{c}_i(t)$ is C_{down}, a transaction is applied in which one unit of an asset is sold at time t and one unit purchased at time $t+1$. If $\dot{c}_i(t)$ is C_{up}, the opposite transaction occurs. We assume that assets can be bought and sold countless times in any quantity and that the short selling of assets is allowed.

Table 7
Trading strategy using predicted classification output.

Conditions	Transactions	Daily profits
$\dot{c}_i(t) = C_{down}$	Sell → Buy	$\text{Close}(t) - \text{Close}(t+1)$
$\dot{c}_i(t) = C_{up}$	Buy → Sell	$\text{Close}(t+1) - \text{Close}(t)$

The daily profit and loss are calculated based on the transactions listed in Table 7, while Table 8 summarizes the cumulative profits, Sharpe ratio, and turnover. Fig. 7 illustrates the cumulative profits over time for comparison with the other strategies. We assume that the transaction cost is fixed at 0.3% [61]. For all stock indices, LFM scores the highest cumulative profits. LFS scores the next-highest cumulative profits except for KOSPI200. The cumulative profits of LFM are higher than those of LFS because the former predicts more accurately the period in which the returns change significantly with MTL. LFM has 2.47-times higher cumulative profits on average than the long-only strategy for all indices. RF and LSTM_{base} have negative cumulative profits.

The long-only strategy is more profitable than the other baseline models (RF and LSTM_{base}) in the S&P500 index because this index exhibits a typical upward trend during the test period, as depicted in Fig. 3. In contrast, the trends of SSE and KOSPI200 in the test period exhibit mixed up and down patterns; thus, the long-only strategy has extremely low cumulative profits and SSE exhibits negative cumulative profits. The Sharpe ratio values are higher for LFS and LFM than those for the long-only strategy for all indices, and those for LFM are higher than those for LFS. The turnover of each method is between 48% and 55%.

4.5.2. LFM versus LSTM_v

In this study, we developed a multi-task model with stock-return regression tasks and stock return direction-classification tasks. However, it is possible to define tasks differently for stock-market forecasting. MTL typically determines tasks in four ways: low rank, task clustering, task relation learning, and decomposition [21]. Traditional approaches decompose the returns into $|\boldsymbol{r_t}|$ and sign $(\boldsymbol{r_t})$ to predict the volatility and return directions in financial markets [62,63]. Some studies used the volatility-prediction task in MTL [22,23]. Therefore, we also designed a variant LFM model using sign $(\boldsymbol{r_t})$ and $|\boldsymbol{r_t}|$ that predict volatility accurately and are not redundant. We refer to this variant of the proposed model as LFM_v.

The performance of LFM and LFM_v for S&P500, SSE, and KOSPI200 are compared in Fig. 8. The average RMSE of the return-prediction performance of the three indices is 11.1% smaller for LFM than for LFM_v. Furthermore, the return direction–classification performance (BACC) is 0.86 points higher on average for LFM than for LFM_v. After the trading exercise, the profits of LFM_v for S&P500, SSE, and KOSPI200 were 824.31, 780.46, and 157.61, respectively, and the LFM profits on the three indices were 20.3% higher on average than those when using LFM_v. We believe this is because LFM_v underestimated the importance of predicting the return direction compared with LFM.

Furthermore, the predictability and profitability of LFM_v were significantly superior to those of long-only. However, the LFM results are superior to those of LFM_v for all measures. The return-prediction error is significant in LFM_v because multiplying the

Table 8
Comparison of trading results with transaction costs for each method.

Stock index	Metrics	Long-only	RF	$LSTM_{base}$	LFS	LFM
S&P500	Cumulative profits	601	−591	−1246	607	989
	Sharpe ratio	0.03	0.01	−0.06	0.04	0.07
	Turnover (%)	–	49	53	54	55
SSE	Cumulative profits	−1179	−1732	−312	172	1049
	Sharpe ratio	−0.03	−0.01	0.00	0.02	0.04
	Turnover (%)	–	49	53	53	51
KOPSI200	Cumulative profits	33	−80	−141	16	194
	Sharpe ratio	0.01	0.03	−0.00	0.07	0.16
	Turnover (%)	–	51	53	53	48

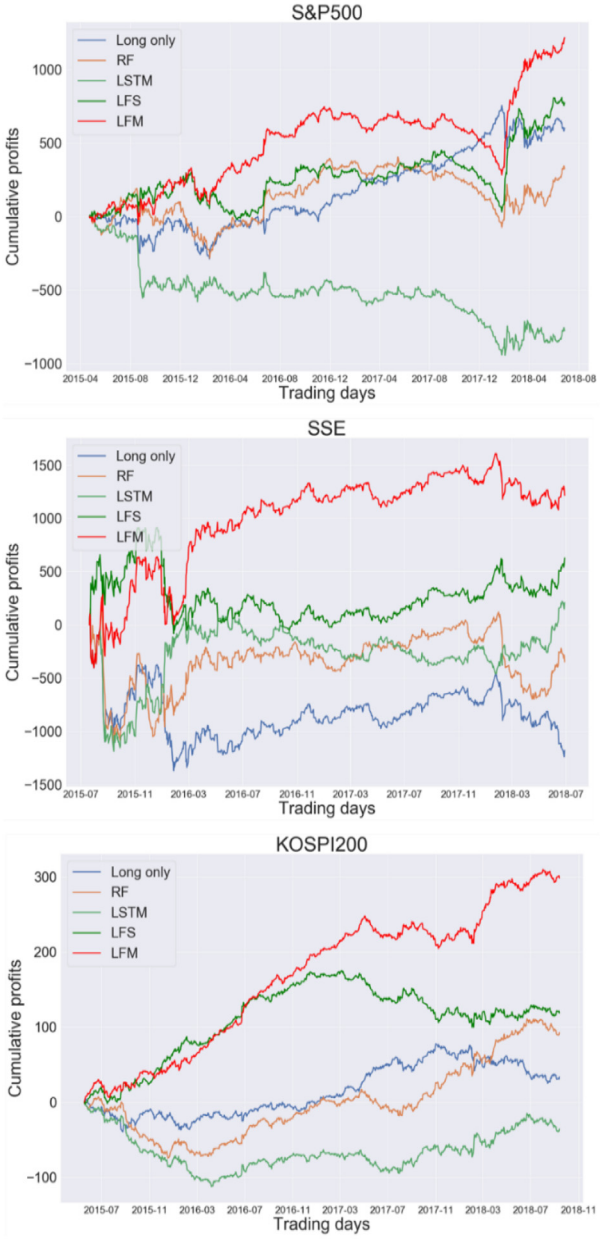


Fig. 7. Cumulative profits of stock indices without transaction costs during test period.

predicted values of the two tasks yields the return-prediction value (i.e., the error is amplified). These results suggest that LFM chose a more appropriate task than the models of previous studies using volatility as the MTL task [22,23].

4.6. Testing and analysis

4.6.1. Statistical test

A Diebold–Mariano (DM) test [64] is conducted to verify the prediction accuracy of the two models and assess statistically significant differences between their predictions. The DM test is based on a null hypothesis: there is no difference between the two forecast errors. The forecast errors are denoted as $e_{i,t}$, defined as $e_{i,t} = a_{i,t} - p_{i,t}$, where i is the index of the prediction models, $a_{i,t}$ is the actual value at time t, and $p_{i,t}$ is the predicted value at time t. The null hypothesis of equal forecasting accuracies for the two forecasts is $E\left[g\left(e_{i,t}\right)\right] = E\left[g\left(e_{j,t}\right)\right]$, where $g(\cdot)$ is a specific loss function. Thus, the "equal accuracy" null hypothesis is equivalent to the null hypothesis because the population mean of the loss-differential series is 0. The DM statistics are as follows:

$$DM = \frac{\bar{d}}{\sqrt{2\pi \dot{f}_d(0)/T}}, \tag{13}$$

where T is the total forecast period, $d_t = g\left(e_{i,t}\right) - g\left(e_{j,t}\right)$, $\bar{d} = \sum_{t=1}^{T}(d_t)$, and $\dot{f}_d(0)$ are the consistent estimates of $f_d(0)$.

If the p-value is less than 0.05, we reject the null hypothesis and conclude that the difference in predictions between the two models is significant. Table 9 lists the DM test results. Based on RMSE, MAE, and MAPE, most DM results are significant at the 0.01 level of p-values, and some are significant at the 0.05 level. Consequently, the results of the LFM, LFS, and baseline models demonstrate statistically significant evidence that they differ.

4.6.2. Variable-importance analysis

For real applications, it is important to identify critically relevant predictors rather than just predicting responses using "black box" models. Our proposed framework combining LSTM and RF addresses this issue. The primary technical indicators can be identified through an RF variable-importance analysis, making the proposed model explainable. The relative importance is calculated using Eqs. (14) and (15) based on variable importance [65]. Here, $\mathrm{VI}_{rmse}\left(v_j\right)$ is the importance of variable v_j based on RMSE. Eq. (14) is used to obtain $\mathrm{VI}_{rmse}\left(v_j\right)$.

$$\mathrm{VI}_{rmse}\left(v_j\right) = \frac{1}{n_{tree}}\sum_{t=1}^{n_{tree}}\frac{\Sigma_{i\in test}\, loss\left(y_i, f_t\left(x_i\right)\right)}{t \in st} - \frac{1}{n_{tree*}}\sum_{t=1}^{n_{tree*}}\frac{\Sigma_{i\in test}\, loss\left(y_i, f_t\left(x_i\right)\right)}{test} \tag{14}$$

In Eq. (14), n_{tree} is the number of all LSTM models used in the ensemble, and n_{tree*} is the number of LSTM models that do not contain v_j. $f_t(x)$ is the prediction result of the tth weak model,

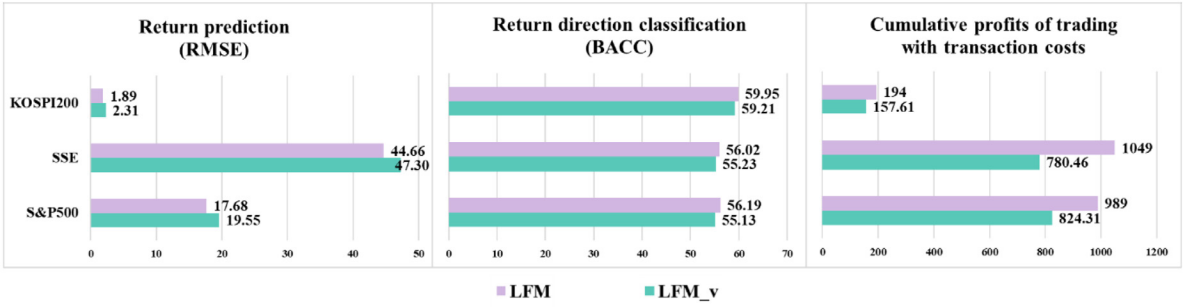


Fig. 8. Performance comparison of LFM and LFM_v.

Table 9
Diebold–Mariano (DM) test results.

	Metrics	RF & LSTM	RF & LFS	RF & LFM	LSTM & LFS	LSTM & LFM	LFS & LFM
S&P500	RMSE	7.38**	8.26**	19.51**	3.37**	13.48**	7.66**
	MAE	25.41**	25.81**	66.5**	2.8**	59.84**	14.04**
	MAPE	19.22**	20.35**	49.33**	3.15**	45.11**	11.6**
SSE	RMSE	26.85**	22.85**	21.13**	11.70**	37.00**	14.45**
	MAE	56.26**	54.04**	100.08**	22.41**	100.53**	13.73**
	MAPE	45.3**	43.96**	70.24**	20.25**	76.71**	14.33**
KOSPI200	RMSE	6.07**	7.52**	32.15**	5.70**	19.91**	2.39*
	MAE	25.35**	25.02**	81.21**	3.21**	86.05**	−7.38**
	MAPE	21.55**	19.89**	62.63**	5.41**	70.09**	4.67**

*Indicate that the p-values are less than 0.05.
**Indicate that the p-values are less than 0.01.

and y_i is the actual value. The first term is the average RMSE of n_{tree} weak models, whereas the last term is the average RMSE of the weak model without v_j variables. Therefore, a small value of $\mathrm{VI}_{rmse}\left(v_j\right)$ indicates that v_j is an important variable. Similar to the calculation of $\mathrm{VI}_{rmse}\left(v_j\right)$, the calculation of $\mathrm{VI}_{bacc}\left(v_j\right)$ is based on BACC. Eq. (15) calculates $\mathrm{VI}_{bacc}\left(v_j\right)$ as follows:

$$\mathrm{VI}_{bacc}\left(v_j\right) = \frac{1}{n_{tree}}\sum_{t=1}^{n_{tree}}\frac{\Sigma_{i\in test} I\left(y_i = f_t\left(x_i\right)\right)}{t \in st} - \frac{1}{n_{tree^*}}\sum_{t=1}^{n_{tree^*}}\frac{\Sigma_{i\in test} I\left(y_i = f_t\left(x_i\right)\right)}{test} \quad (15)$$

In Eq. (15), f_t (x) is the prediction result of the tth weak classifier, y is the actual value, and I is the indicator function. The first term is the average BACC of n_{tree} weak classifiers, whereas the last term is the average BACC of the weak classifier without v_j variables. Therefore, v_j becomes an important variable as the value of $\mathrm{VI}_{bacc}\left(v_j\right)$ increases.

In this study, we perform variable importance analysis using Eqs. (14) and (15) on S&P500, SSE, and KOSPI200 indices for the return and direction predictions. Based on the relative variable importance scores, top-10 variables are selected for all cases. The result demonstrates that top-10 variables vary, depending on the indices and tasks. For an integrated analysis, we assume that the considerably important variables are frequently included in the three stock indices and the two tasks.

Fig. 9(a) illustrates the detailed treemap [66] based on the top-10 frequency in the tasks and stock indices. The rectangular area indicates the number of times the variables are included in the top-10 in the tasks and stock indices, where the variables with the same frequency form the same group. For example, V.F. (5) represents the variables included five times in the top-10 variables. The color of each rectangle represents the average relative variable importance score.

In the treemap, the variables most influential to the stock returns and return-direction forecasts are Close, RSI10, ROC12, and SD20. Technical indicators such as CCI20, H–Cp, H–L, MACD6_20, and Volume are secondary influential variables. Furthermore, the darker the color, the higher the relative variable-importance score in Fig. 9(a). Volume, Close, CCI20, SD20, KST, TR, OBV, and NVI are critical variables in the models based on an analysis of the average relative variable-importance score. These technical indicators have also been considered important for predicting the stock market in previous studies [28,31–34].

The Venn diagram in Fig. 9(b) is used to analyze important variables for each task. The variables are chosen based on the top-10 appearance frequencies for each task and reduced to seven variables for each task under the condition that a variable is included at least twice for all three indices. In total, there are 11 important variables for the two tasks according to the frequency and three common variables in the intersection: SD20, Close, and RSI10.

4.6.3. Comparison with LSTMs using selected variables

In this study, we hypothesize that using many available technical indicators rather than several selected technical indicators is beneficial for DL-based prediction models. LSTM models were created to test this hypothesis with only the variables (top k, where $k = 5$, 10, 20, 30, and 43) selected during the variable importance analysis (Section 4.6.2). The models were compared using LFM. Top k refers to the top-k indicators of the variable importance analysis. If k is 43, all indicators are used for LSTM learning; this is the same model as LSTM_{base} in Section 4.3.

We present the results in Table 10. The performance of the LSTM models using the top-k variables is inferior to those of the proposed LFM in terms of RMSE and BACC. Tables 6 and 10 indicate that LFM outperforms other models that use dimension reduction methods. Furthermore, we compared the LSTM model with the 43 variables, i.e., LSTM_{base}, with LFM to confirm our model's robustness against overfitting.

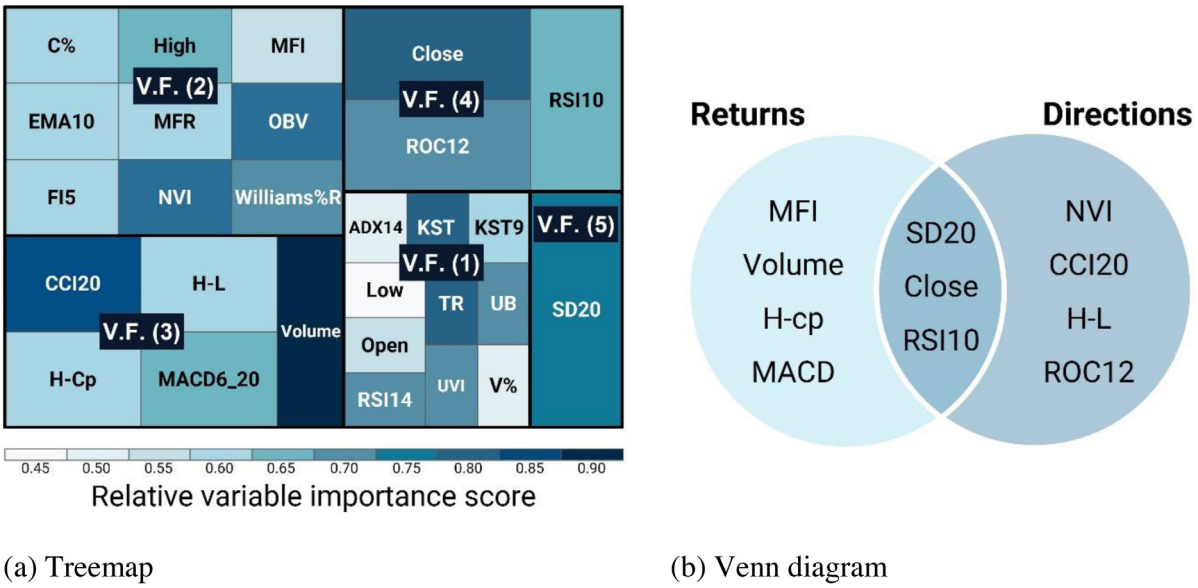


Fig. 9. Results of variable importance analysis based on top-10 frequency of appearance for two tasks and three stock indices.

Table 10
Return-prediction and direction-classification results of LSTM models using top-k indicators.

Metric	Method	S&P500	SSE	KOSPI200
RMSE	LSTM(5)	20.67	47.49	2.44
	LSTM(10)	21.40	48.20	2.29
	LSTM(20)	21.68	49.73	2.37
	LSTM(30)	20.14	49.93	2.40
	LSTM(43)	20.45	47.98	2.38
	LFM	17.67	44.66	1.89
BACC	LSTM(5)	50.06	48.23	52.61
	LSTM(10)	49.98	50.84	52.71
	LSTM(20)	50.73	50.30	51.03
	LSTM(30)	49.70	50.94	51.10
	LSTM(43)	47.58	49.92	48.97
	LFM	56.19	56.02	59.95

5. Discussion

Our proposed method demonstrated remarkable performance in stock market predictability and profitability for two reasons. First, our model addresses the overfitting problem. As presented in Table 10, LFM outperformed LSTM$_{base}$ (LSTM(43)) significantly in both return prediction and return-direction prediction for all three indices, although they used the same technical indicators. Furthermore, in Fig. 10, LFM illustrates little difference between the training and testing errors, whereas the LSTM(43) model presents a significant difference (typical when overfitting occurs). Although the technical indicators can contain helpful information for prediction, merely including them all in the input to the LSTM model does not guarantee improved performance.

Second, LFM uses MTL with appropriate tasks so that trading returns are high, especially when stock markets are highly volatile. In all trading tests described in Section 4.5.1, LFM is more profitable than LFS, particularly the long-only strategy. As presented in Table 11, LFM predicts the return directions particularly well during periods of large volatilities higher than 10%. It simultaneously learns returns and directions of returns; therefore, it can extract more distinct patterns during periods of high volatility. In Fig. 7, the profits of LFM were relatively high in early 2018 when the global financial markets declined. This period was highly volatile due to the monetary policy normalization in the US and Europe and concerns with the spread of trade conflicts.

The academic and practical implications of our proposed model are as follows. First, new technical indicators can be easily added to our model. In practical applications, further performance improvements are desirable because even tiny changes in performance can result in huge losses or profits. Therefore, technical indicators and their parameters must be modified often, and novel technical indicators should be supplemented. In our framework, new indicators are added to the pool of technical indicators; they do not increase the number of input variables of an LSTM model, a component of LSTM–Forest. Each LSTM model of LSTM–Forest updates itself with important financial market-forecasting variables.

Second, the proposed framework is readily applicable to other fields that study high-dimensionality problems. In our study, the proposed framework handles financial issues but does not use additional financial information besides technical indicators for inputs and indices for targets. Thus, the framework can be used in the time-series of various domains. Consequently, LSTM–Forest could be useful for many professionals who require predictive models in their respective fields. Furthermore, it could provide valuable insights for researchers who have difficulty modeling DL due to insufficient data.

Despite these implications, our proposed method has limitations. If unforeseen events occur or unseen patterns in the training data appear in the real-time market, the proposed model may not perform well. Online learning has been used in recent DL studies to address this issue [17,67]. Online learning is an ML technique used to train new patterns adaptively based on newly generated data over time. By applying online learning, our model can be updated incrementally based on new data and is robust to unexpected patterns instead of being re-trained over an entire dataset.

Second, the computational cost of LFM is higher than that of a single LSTM model. In our proposed method, the primary factor that affects the computational cost is the number of LSTMs (n), a parameter that functions as the number of trees in the RF. The steps of the time window (N_{time}) and the number of input variables (K) are also positively correlated with the computational cost. We reduce computational complexity by applying parallelization [68] or model compression [69], which enable each LSTM to be trained and executed independently during inference. A task-specific branch can be separated from the LFM, as required,

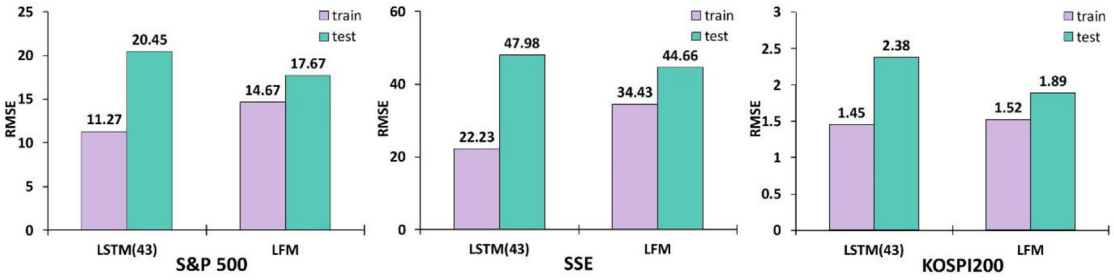


Fig. 10. Training and test errors of the LSTM (43) model and LFM.

Table 11
Stock index return-direction prediction accuracy (%) of Long-only and LFM according to volatility size during the test periods.

Period	Volatility (σ)	Long-only			LFM		
		S&P500	SSE	KOSPI200	S&P500	SSE	KOSPI200
Test	$0\% \le \sigma < 5\%$	54.1	53.1	52.4	43.5	53.9	59.1
	$5\% \le \sigma < 10\%$	51.2	50.4	50.6	52.4	50.0	57.9
	$10\% \le \sigma < 15\%$	57.2	51.3	56.2	60.9	54.6	65.8
	$15\% \le \sigma$	52.6	54.1	–	57.9	60.9	–

to reduce the cost further. However, even with these limitations, the advantages of our model, such as high performance, structural robustness, and expandability, make it suitable for application in the real world.

6. Conclusions

We proposed a novel DL-based LSTM–Forest framework that can use the time-series of many technical indicators while preventing overfitting to improve stock market forecasting. The framework of our proposed method combines an RF (which can have many variables without overfitting) and LSTM, which is excellent for learning temporal patterns. It is an ensemble model of LSTM models generated through numerous combinations of variables and training data. We also proposed the LFM multi-task model (with a stock return-prediction task and a stock direction-classification task), the LFS single-task model, and an LFM-based trading system.

The superiority and suitability of our proposed framework have been demonstrated in experiments. First, we compared our proposed models with the baseline models, RF and LSTM_{base}. The temporal pattern learning of our models was superior to RF. The proposed model also learns more effectively while preventing overfitting than the LSTM_{base} model. Furthermore, the DM test results of the proposed and baseline models differ significantly. Second, we compared the LFS and LFM results and demonstrated that the MTL model is superior in both return prediction and return-direction classification. When applied to trading, we found that LFM is more profitable than LFS. LFM with transaction costs achieved approximately 2.47-times higher profits than the long-only strategy. Third, our model is explainable because it uses the variable importance analysis of RF, although the DL model is typically a black-box model. Finally, the performance of the LSTM models using only selected variables from the variable importance analysis was compared with that of LFM. We demonstrated that using as many as 43 technical indicators rather than several selected technical indicators is beneficial for DL-based prediction models.

Future research could focus on improving the proposed model. First, the proposed method can be extended to use text data (e.g., financial news) as additional inputs to the model (along with the technical indicators). Furthermore, in LSTM–Forest, the architecture of the LSTM model can be advanced using an attention mechanism. Finally, we can extend LSTM–Forest to tasks related to volatility and other stock-market predictions for MTL.

CRediT authorship contribution statement

Hyun Jun Park: Writing – original Draft, Methodology, Formal Analysis, Software, Data curation, Investigation, Visualization. **Youngjun Kim:** Writing – review & editing, Methodology, Formal Analysis, Software, Investigation, Visualization. **Ha Young Kim:** Conceptualization, Methodology, Formal Analysis, Investigation, Writing – review & editing, Supervision, Project Administration, Resources, Funding Acquisition.

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.

Funding support

This research was supported by a grant from the National Research Foundation of Korea (NRF) [grant number NRF-2020R1F1A1071527] funded by the Korea government (MSIT; Ministry of Science and ICT). This research was also supported by the Yonsei University, Republic of Korea Research Fund of 2020 [grant number 2020–22–0523]. The funding sources played no role in the study design, data collection, and analysis; decision to publish; or preparation of the manuscript.

References

[1] G.J. Deboeck, Trading on the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets, Wiley, New York, 1994.
[2] D. Enke, S. Thawornwong, The use of data mining and neural networks for forecasting stock market returns, Expert Syst. Appl. 29 (4) (2005) 927–940.
[3] J.-Z. Wang, J.-J. Wang, Z.-G. Zhang, S.-P. Guo, Forecasting stock indices with back propagation neural network, Expert Syst. Appl. 39 (2011) 14346–14355.
[4] N. Devpura, P.K. Narayan, S.S. Sharma, Is stock return predictability time-varying? J. Int. Final. Mark. Inst. Money 52 (2018) 152–172.
[5] Timmermann A., Elusive return predictability, Int. J. Forecast. 24 (1) (2008) 1–18.
[6] T.H. Nguyen, K. Shirai, J. Velcin, Sentiment analysis on social media for stock movement prediction, Expert Syst. Appl. 42 (24) (2015) 9603–9611.
[7] S. Karasu, A. Altan, S. Bekiros, W. Ahmad, A new forecasting model with wrapper-based feature selection approach using multi-objective optimization technique for chaotic crude oil time series, Energy 212 (2020) 118750.

[8] A. Altan, S. Karasu, The effect of kernel values in support vector machine to forecasting performance of financial time series, J. Cogn. Syst. 4 (1) (2019) 17–21.
[9] R.P. Schumaker, H. Chen, A quantitative stock prediction system based on financial news, Inf. Process. Manage. 45 (5) (2009) 571–583.
[10] O.B. Sezer, A.M. Ozbayoglu, Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach, Appl. Soft Comput. 70 (2018) 525–538.
[11] E. Hoseinzade, S. Haratizadeh, CNNpred: CNN-based stock market prediction using a diverse set of variables, Expert Syst. Appl. 129 (2019) 273–285.
[12] Y.C. Chen, W.C. Huang, Constructing a stock-price forecast CNN model with gold and crude oil indicators, Appl. Soft Comput. (2021) 107760.
[13] W. Bao, J. Yue, Y. Rao, A deep learning framework for financial time series using stacked autoencoders and long-short term memory, PLoS One 12 (7) (2017) e0180944.
[14] T. Fischer, C. Krauss, Deep learning with long short-term memory networks for financial market predictions, European J. Oper. Res. 270 (2) (2018) 654–669.
[15] J. Qiu, B. Wang, C. Zhou, Forecasting stock prices with long-short term memory neural network based on attention mechanism, PLoS One 15 (1) (2020) e0227222.
[16] Y. Huang, Y. Gao, Y. Gan, M. Ye, A new financial data forecasting model using genetic algorithm and long short-term memory network, Neurocomputing 425 (2021) 207–218.
[17] S.W. Lee, H.Y. Kim, Stock market forecasting with super-high dimensional time-series data using convlstm, trend sampling, and specialized data augmentation, Expert Syst. Appl. 161 (2020) 113704.
[18] Y. Baek, H.Y. Kim, ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module, Expert Syst. Appl. 113 (2018) 457–480.
[19] A.N. Kia, S. Haratizadeh, S.B. Shouraki, A hybrid supervised semi-supervised graph-based model to predict one-day ahead movement of global stock markets and commodity prices, Expert Syst. Appl. 105 (2018) 159–173.
[20] P. Ghosh, A. Neufeld, J.K. Sahoo, Forecasting directional movements of stock prices for intraday trading using LSTM and random forests, Finance Res. Lett. (2021) 102280.
[21] Y. Zhang, Q. Yang, A survey on multi-task learning, IEEE Trans. Knowl. Data Eng. (2021) 1.
[22] C. Li, D. Song, D. Tao, Multi-task recurrent neural networks and higher-order Markov random fields for stock price movement prediction: Multi-task RNN and higher-order MRFs for stock price classification, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 1141–1151.
[23] R. Sawhney, P. Mathur, A. Mangal, P. Khanna, R.R. Shah, R. Zimmermann, Multimodal multi-task financial risk forecasting, in: Proceedings of the 28th ACM international conference on multimedia, 2020, pp. 456–465.
[24] T. Ma, Y. Tan, Multiple stock time series jointly forecasting with multi-task learning, in: 2020 International Joint Conference on Neural Networks, IJCNN, IEEE, 2020, pp. 1–8.
[25] S. Mootha, S. Sridhar, R. Seetharaman, S. Chitrakala, Stock price prediction using bi-directional LSTM based sequence to sequence modeling and multitask learning, in: 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference, UEMCON, IEEE, 2020, pp. 0078–0086.
[26] T.F. Cootes, M.C. Ionita, C. Lindner, P. Sauer, Robust and accurate shape model fitting using random forest regression voting, in: European Conference on Computer Vision, Springer, Berlin, Heidelberg, 2012, pp. 278–291.
[27] Y. Ma, R. Han, X. Fu, Stock prediction based on random forest and LSTM neural network in: 2019 19th international conference on control, automation and systems, ICCAS, Jeju, Korea (South), 2019(10), 2019, pp. 126–130.
[28] J. Badge, Forecasting of Indian stock market by effective macro-economic factors and stochastic model, J. Stat. Econom. Methods 1 (2) (2012) 39–51.
[29] M.H. Chen, An ETF trading decision support system by using neural network and technical indicators, in: The 2006 IEEE International Joint Conference on Neural Network Proceedings, IEEE, 2006, pp. 2394–2401.
[30] P. Ładyzyński, K. Żbikowski, P. Grzegorzewski, Stock trading with random forests, trend detection tests and force index volume indicators, in: International Conference on Artificial Intelligence and Soft Computing, Springer, Berlin, Heidelberg, 2013, pp. 441–452.
[31] R. Majhi, G. Panda, G. Sahoo, Development and performance evaluation of FLANN based model for forecasting of stock markets, Expert Syst. Appl. 36 (3) (2009) 6800–6808.
[32] R.K. Nayak, D. Mishra, A.K. Rath, A Naïve SVM-KNN based stock market trend reversal analysis for Indian benchmark indices, Appl. Soft Comput. 35 (2015) 670–680.
[33] D.O. Oyewola, E.G.G. Dada, O.E. Olaoluwa, K.A. Al-Mustapha, Predicting Nigerian stock returns using technical analysis and machine learning, Eur. J. Electr. Eng. Comput. Sci. 3 (2) (2019).
[34] G.R. Weckman, S. Lakshminarayanan, J.H. Marvel, A. Snow, An integrated stock market forecasting model using neural networks, Int. J. Bus. Forecast. Market. Intell. 1 (1) (2008) 30–49.
[35] H. Dourra, P. Siy, Investment using technical analysis and fuzzy logic, Fuzzy Sets and Systems 127 (2) (2002) 221–240.
[36] R.D. Edwards, J. Magee, W.C. Bassetti, Technical analysis of stock trends, CRC Press, 2018.
[37] A.W. Lo, H. Mamaysky, J. Wang, Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation, J. Finance 55 (4) (2000) 1705–1765.
[38] Y. Hu, B. Feng, X. Zhang, E.W.T. Ngai, M. Liu, Stock trading rule discovery with an evolutionary trend following model, Expert Syst. Appl. 42 (1) (2015) 212–222.
[39] A. Kabasinskas, U. Macys, Calibration of bollinger bands parameters for trading strategy development in the baltic stock market, InžinerinĖ Ekonomika 21 (3) (2010) 244–254.
[40] E. Gatev, W.N. Goetzmann, K.G. Rouwenhorst, Pairs trading: Performance of a relative-value arbitrage rule, Rev. Financ. Stud. 19 (3) (2006) 797–827.
[41] M.A. Dempster, V. Leemans, An automated FX trading system using adaptive reinforcement learning, Expert Syst. Appl. 30 (3) (2006) 543–552.
[42] E.A. Gerlein, M. McGinnity, A. Belatreche, S. Coleman, Evaluating machine learning classification for financial trading: An empirical approach, Expert Syst. Appl. 54 (2016) 193–207.
[43] A. Booth, E. Gerding, F. Mcgroarty, Automated trading with performance weighted random forests and seasonality, Expert Syst. Appl. 41 (8) (2014) 3651–3661.
[44] Breiman L., Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[45] A Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques To Build Intelligent Systems, second ed., O'Reilly Media, Incorporated, 2019.
[46] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[47] F.A. Badawy, H.Y. Abdelazim, M.G. Darwish, Genetic algorithms for predicting the Egyptian stock market, in: 2005 International Conference on Information and Communication Technology, IEEE, 2005, pp. 109–122.
[48] Y. Chen, Y. Hao, A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction, Expert Syst. Appl. 80 (2017) 340–355.
[49] M. Göçken, M. Özçalıcı, A. Boru, A.T. Dosdoğru, Integrating metaheuristics and artificial neural networks for improved stock price prediction, Expert Syst. Appl. 44 (2016) 320–331.
[50] M.W. Hsu, S. Lessmann, M.C. Sung, T. Ma, J.E. Johnson, Bridging the divide in financial market forecasting: machine learners vs. financial economists, Expert Syst. Appl. 61 (2016) 215–234.
[51] M.Y. Chen, M.H. Fan, Y.L. Chen, H.M. Wei, Design of experiments on neural network's parameters optimization for time series forecasting in stock markets, Neural Netw. World 23 (4) (2013) 369.
[52] M. Qiu, Y. Song, F. Akagi, Application of artificial neural network for the prediction of stock market returns: The case of the Japanese stock market, Chaos Solitons Fractals 85 (2016) 1–7.
[53] M.T. Leung, H. Daouk, A.S. Chen, Forecasting stock indices: A comparison of classification and level estimation models, Int. J. Forecast. 16 (2000) 173–190.
[54] Chollet, et al., Keras, 2015, https://github.com/keras-team/keras (2015).
[55] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv:1412.6980.
[56] M. Bekkar, H.K. Djemaa, T.A. Alitouche, Evaluation measures for model assessment over imbalanced data sets, J. Inform. Eng. Appl. 3 (10) (2013).
[57] J. Wang, J. Wang, Forecasting stock market indexes using principle component analysis and stochastic time effective neural networks, Neurocomputing 156 (2015) 68–78.
[58] 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.
[59] N. Jing, Z. Wu, H. Wang, A hybrid model integrating deep learning with investor sentiment analysis for stock price prediction, Expert Syst. Appl. 178 (2021) 115019.
[60] Y. Wang, L. Wang, F. Yang, W. Di, Q. Chang, Advantages of direct input-to-output connections in neural networks: The Elman network for stock index forecasting, Inform. Sci. 547 (2021) 1066–1079.
[61] H.H. Huang, H.Y. Huang, J.J. Oxman, Stock liquidity and corporate bond yield spreads: Theory and evidence, J. Final. Res. 38 (1) (2015) 59–91.
[62] T.G. Andersen, T. Bollerslev, F.X. Diebold, P. Labys, Modeling and forecasting realized volatility, Econometrica 71 (2) (2003) 579–625.
[63] P.F. Christoffersen, F.X. Diebold, Financial asset returns, direction-of-change forecasting, and volatility dynamics, Manage. Sci. 52 (8) (2006) 1273–1287.
[64] F.X. Diebold, R.S. Mariano, Comparing predictive accuracy, J. Bus. Econom. Statist. 20 (1) (2002) 134–144.
[65] P. Wei, Z. Lu, J. Song, Variable importance analysis: a comprehensive review, Reliab. Eng. Syst. Saf. 142 (2015) 399–432.

[66] B. Johnson, B. Shneiderman, Tree-Maps: A Space-Filling Approach To the Visualization of Hierarchical Information Structures, IEEE, 1991, pp. 284–291.
[67] A. Sarabakha, E. Kayacan, Online deep learning for improved trajectory tracking of unmanned aerial vehicles using expert knowledge, in: 2019 international conference on robotics and automation, ICRA, Montreal, QC, Canada, 2019, pp. 7727–7733.
[68] A.S. Sambo, R.M.A. Azad, Y. Kovalchuk, V.P. Indramohan, H. Shah, Evolving simple and accurate symbolic regression models via asynchronous parallel computing, Appl. Soft Comput. 104 (2021) 107198.
[69] W. Zhang, G. Biswas, Q. Zhao, H. Zhao, W. Feng, Knowledge distilling based model compression and feature learning in fault diagnosis, Appl. Soft Comput. 88 (2020) 105958.