Machine Learning with Applications 9 (2022) 100320

Contents lists available at ScienceDirect

Machine Learning with Applications

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

Predicting stock market index using LSTM

Hum Nath Bhandari [a], Binod Rimal [b,*], Nawa Raj Pokhrel [c], Ramchandra Rimal [d], Keshab R. Dahal [e], Rajendra K.C. Khatri [f]

[a] Department of Mathematics, Roger Williams University, Bristol, RI, USA
[b] Department of Mathematical Sciences, Florida Atlantic University, Boca Raton, FL, USA
[c] Department of Physics and Computer Science, Xavier University of Louisiana, New Orleans, LA, USA
[d] Department of Mathematical Sciences, Middle Tennessee State University, Murfreesboro, TN, USA
[e] Department of Statistics, Truman State University, Kirksville, MO, USA
[f] Department of Applied Mathematics, Philander Smith College, Little Rock, AR, USA

ARTICLE INFO

Keywords:
Stock market index
LSTM
Prediction
Machine learning
Deep learning
Denoising

ABSTRACT

The rapid advancement in artificial intelligence and machine learning techniques, availability of large-scale data, and increased computational capabilities of the machine opens the door to develop sophisticated methods in predicting stock price. In the meantime, easy access to investment opportunities has made the stock market more complex and volatile than ever. The world is looking for an accurate and reliable predictive model which can capture the market's highly volatile and nonlinear behavior in a holistic framework. This study uses a long short-term memory (LSTM), a particular neural network architecture, to predict the next-day closing price of the S&P 500 index. A well-balanced combination of nine predictors is carefully constructed under the umbrella of the fundamental market data, macroeconomic data, and technical indicators to capture the behavior of the stock market in a broader sense. Single layer and multilayer LSTM models are developed using the chosen input variables, and their performances are compared using standard assessment metrics–Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Correlation Coefficient (R). The experimental results show that the single layer LSTM model provides a superior fit and high prediction accuracy compared to multilayer LSTM models.

1. Introduction

The stock price fluctuations are uncertain, and there are many interconnected reasons behind the scene for such behavior. The possible cause could be the global economic data, changes in the unemployment rate, monetary policies of influencing countries, immigration policies, natural disasters, public health conditions, and several others. All the stock market stakeholders aim to make higher profits and reduce the risks from the thorough market evaluation. The major challenge is gathering the multifaceted information, putting them together into one basket, and constructing a reliable model for accurate predictions.

Stock price prediction is a complex and challenging task for companies, investors, and equity traders to predict future returns. Stock markets are naturally noisy, non-parametric, non-linear, and deterministic chaotic systems (Ahangar, Yahyazadehfar, & Pournaghshband, 2010). It creates a challenge to effectively and efficiently predict the future price. Feature selection from the financial data is another difficult task in the stock prediction for which many approaches have been suggested (Hoseinzade & Haratizadeh, 2019). There has been a trend in which some researchers use only technical indicators, whereas others use historical data (Di Persio & Honchar, 2016; Kara, Acar Boyacioglu, & Ömer Kaan Baykan, 2011; Nelson, Pereira, & de Oliveira, 2017; Patel, Shah, Thakkar, & Kotecha, 2015; Qiu & Song, 2016; Wang & Kim, 2018). The performance of the predictive model may not be top-notch due to the use of limited features. On the flip side, if all the available features from the financial market are included, the model could be complex and difficult to interpret. In addition, the model performance may be worse due to collinearity among multiple variables.

A proper model developed with an optimal set of attributes can predict stock price reasonably well and better inform the market situation. A plethora of research has been published to study how certain variables correlate with stock price behavior. A varying degree of success is seen concerning the accuracy and robustness of the models. One possible reason for not achieving the expected outcome could be in the variable selection process. There is a greater chance that the

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: hbhandari@rwu.edu (H.N. Bhandari), brimal2014@fau.edu (B. Rimal), npokhrel@xula.edu (N.R. Pokhrel), ramchandra.rimal@mtsu.edu (R. Rimal), kdahal@truman.edu (K.R. Dahal), rkhatri@philander.edu (R.K.C. Khatri).

https://doi.org/10.1016/j.mlwa.2022.100320
Received 8 February 2022; Received in revised form 2 May 2022; Accepted 2 May 2022
Available online 13 May 2022

developed model performs reasonably better if a good combination of features is considered. One of the contributions of this study is selecting the variables by looking meticulously at multiple aspects of the economy and their potential impact in the broader markets. Moreover, a detailed justification is supplied why the specific explanatory variables are chosen in the present context in Section 4.

The field of quantitative analysis in finance has a long history. Several models ranging from naive to complex have been developed so far to find the solution to financial problems. However, not all quantitative analyses or models are fully accepted or widely used. One of the first attempts was made in the seventies by two British statisticians, Box and Jenkins, using mainframe computers (Hansen, McDonald, & Nelson, 1999). They developed the Auto-Regressive Integrated Moving Average (ARIMA) model utilizing only the historical data of price and volume. The ARIMA is used to handle only stationary time series data by default. Performance can be abysmal if it is used for non-stationary data. Therefore, it is essential to convert non-stationary time series data to stationary before implementation, which may lose the original structure and interpretability of the feature. With very few exceptions, almost all classical models assume that data has a linear relationship. This assumption vividly raises the questions about the robustness of the classical time series models as the real-world time series data are often nonlinear.

Things were getting more interesting from the eighties because of the development in data analysis tools and techniques. For instance, the spreadsheet was invented to model financial performance, automated data collection became a reality, and improvements in computing power helped predictive models to analyze the data quickly and efficiently. Because of the availability of large-scale data, advancement in technology, and inherent problem associated with the classical time series models, researchers started to build models by unlocking the power of artificial neural networks and deep learning techniques in the area of sequential data modeling and forecasting. These methods are capable of learning complex and non-linear relationships compared to traditional methods. They are more efficient in extracting the most important information from the given input variables.

Several deep learning architectures have been developed to deal with various problems and the intrinsic structure of datasets. Information flows only in the forward direction in a basic feedforward neural network architecture. Since each input is processed independently, it does not retain information from the previous step. Thus, these models are ineffective in dealing with sequential data where series of prior events are essential in predicting future events. Recurrent neural networks (RNN) are designed to perform such tasks. The RNN architecture consists of loops, allowing relevant information to persist over time. Information is being passed from one timestep to the next internally within the network. Therefore, the RNN is more suitable for sequential data modeling and time series applications such as stock market predictions, language translations, auto-completion in messages/emails, and signal processing. During the training process of the RNN, the cost or error is calculated between the predicted values and the actual values from a labeled training dataset. The error is minimized by repeatedly updating the networks' parameters (weights and biases) until the lowest possible value is obtained. The training process utilizes a gradient, the rate at which cost changes with respect to each parameter. The gradient provides a direction to move in the error surface by adjusting the parameters iteratively. This strategy is called backpropagation, where the error is propagated backward from the output layer all the way up to the input layer. One of the challenges of this technique is that parameters can be anywhere in the networks, and finding a gradient involves calculations of partial derivatives with respect to all the parameters. This process sometimes needs a long chain rule, especially for the parameters in earlier layers of the networks. As a result, gradients could ultimately vanish or decay back through the networks, known as the vanishing gradient problem, a common issue in neural networks training. Unfortunately, this problem also persists in the RNN architecture. LSTM, a typical recurrent neural network architecture, is designed to overcome the vanishing gradient problem (Hochreiter, 1998). Memorizing information for a longer period of time is the default behavior of the LSTM model.

This study considers the computational framework to predict the stock index price using the LSTM model, the improved version of neural networks architecture for time series data. The bird's-eye view of the proposed research framework via the schematic diagram is expressed in Fig. 1. As outlined in the diagram, the proposed study utilizes the carefully selected features from fundamental, macroeconomic, and technical data to build the model. After that, the collected data has been normalized using the min–max normalization technique. Then input sequence for the LSTM model is created using a specific time step. The hyperparameters such as number of neurons, epochs, learning rate, batch size, and time step have been incorporated in the model. The regularization techniques have been utilized to overcome the overfitting problems. Once the hyperparameters are tuned, the input data is fed into the LSTM model to predict the closing price of the stock market index. The quality of the proposed model is assessed through RMSE, MAPE, and R.

In a nutshell, plenty of research has been done in predicting the stock market. Some research focuses on complex statistical or machine learning techniques without focusing on the type of attributable variables. Others use only the fundamental data without exploring additional factors that could influence the stock market prediction. There is a need to develop a model with a good combination of features of the stock market variables and simplicity in model architecture. Thus, our contribution is to create a model without adding any complexity in model architecture and maintaining well-balanced set of variables to capture the behavior of the stock market from multiple dimensions.

The rest of the paper is organized as follows. Section 2 explains the related work in this field. Section 3 explores the implementation of the LSTM model in the S&P 500 data. The data collection and feature selection procedure are explained in Section 4. Model outcomes are discussed in Section 5. It also explains the predictive capability of the model after tuning the hyperparameters. Finally, Section 6 presents the conclusion and future work, followed by acknowledgments, list of references, and appendix.

2. Related work

Chen et al. used the LSTM model to predict China stock returns (Chen, Zhou, & Dai, 2015). The historical data was transformed into 30-days long sequences with ten learning features and 3-day learning rate labeling. The authors claimed that the model improved the accuracy from 14.3% to 27.2% compared to the random prediction method. Bao et al. applied the Haar wavelet transformation to denoise the financial time series data and implemented the stacked autoencoders to learn the deep features of the data and then used LSTM to predict the closing price of stock indices (Bao, Yue, & Rao, 2017). Their average R score was below 88% on the LSTM model for S&P 500. Roondiwala et al. used the LSTM model for the NIFTY 50 data ranges from 2011 to 2016 (Roondiwala, Patel, & Varma, 2017). The authors used fundamental data (open, close, low, and high) without incorporating macroeconomic and technical indicators to predict the closing price.

Fischer and Krauss used LSTM networks for the classification problem of predicting directional movements for the constituent stocks of S&P 500 from 1992 until 2015 (Fischer & Krauss, 2018). The authors concluded that the LSTM network could effectively extract meaningful information from the financial time series data. Based on prediction accuracy and daily returns after transaction costs, LSTM outperforms random forests, standard deep networks, and logistic regression. Qiu et al. implemented LSTM based model on historical data of S&P 500, Dow Jones Industrial Average (DJIA), and Hang Seng Index dataset (Qiu, Wang, & Zhou, 2020). They applied an attention mechanism extracting the information in the news to evaluate the price

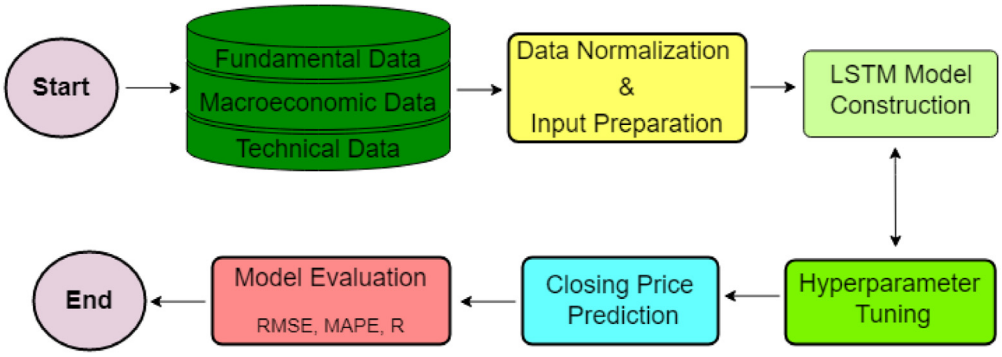

Fig. 1. Schematic diagram of the proposed research framework.

fluctuation. The authors denoised the data using wavelet transformation and then implemented their attention-based LSTM framework to predict the opening index price using only fundamental market data. Lanbouri and Achchab used the LSTM model for the high-frequency trading perspective in which their goal was to use the S&P 500 stock trading data to predict the stock price in the next 1, 5, and 10 minutes (Lanbouri & Achchab, 2020). Yadav et al. implemented LSTM model with various hidden layers to Indian stock market data removing the trend and seasonality components to predict the closing price (Yadav, Jha, & Sharan, 2020). They concluded that the single layer LSTM model had better prediction accuracy.

Kara et al. used support vector machine (SVM) and artificial neural network to predict movement in the daily Istanbul Stock Exchange National 100 Index from 1997 to 2007 (Kara et al., 2011). The authors selected ten technical indicators as input for their model. Experimental results showed that the average performance of the artificial neural network model was significantly better than that of the SVM model. Karmiani et al. compared LSTM, Backpropagation, SVM, and Kalman filter to predict stock price (Karmiani, Kazi, Nambisan, Shah, & Kamble, 2019). The stock price of selected nine companies were considered for the prediction. LSTM was the best choice in terms of prediction accuracy with low variance. Yu and Yan combined phase-space reconstruction method for time series analysis and LSTM model to predict the stock price (Yu & Yan, 2019). Various market environments such as the S&P 500, DJIA, Nikkei 225, Hang Seng Index, China Securities Index 300, and ChiNext index were considered for the analysis. This prediction model was built by taking the historical price only. The outcomes of LSTM with Multilayer Perception, Support Vector Regressor, and ARIMA were compared. Authors claimed that the LSTM model outperformed other models for S&P 500 data. Gao et al. conducted a comparative study of four machine learning algorithms —Multilayer Perceptron, LSTM, Convolutional Neural Network, and Uncertainty-Aware Attention —to predict the next day's stock price (Gao, Zhang, & Yang, 2020). The S&P 500 index, CSI 300 index, and Nikkei 225 index were taken to represent the most developed market, the less developed market, and the developing market. Open price, close price, trading volume, Moving Average Convergence Divergence, Average True Range, exchange rate, and interest rate were considered predictors. The outcome of the study suggested that Uncertainty-Aware Attention's performance was slightly better than other models. Moreover, additional predictors such as the volatility index and the unemployment rate could improve the model's performance.

3. Modeling approach

3.1. A brief overview of LSTM

LSTM is a popular deep learning technique in RNN for time series prediction. For example, LSTM is used for both classification and regression problems not only for the stock market prediction but the rainfall runoff modeling (Kratzert, Klotz, Brenner, Schulz, & Herrnegger, 2018), fMRI data analysis (Rahman et al., 2020), anomaly detection (Lindemann, Maschler, Sahlab, & Weyrich, 2021), mobile traffic prediction (Trinh, Giupponi, & Dini, 2018) to name a few. Although standard RNN is superior to traditional networks in preserving the information, it is not effective in learning long-term dependencies due to the vanishing gradient problem (Hochreiter, 1998). LSTM uses memory cells to overcome the issue of vanishing gradients. It consists of an input layer, a hidden layer, a cell state, and an output layer (Gers, Schmidhuber, & Cummins, 2000; Gers, Schraudolph, & Schmidhuber, 2003; Hochreiter & Schmidhuber, 1997). The key component of LSTM architecture is the cell state which runs through the chain, with only linear interaction, keeping information flow unchanged. The gate mechanism of LSTM deletes or modifies the information of the cell state. It is a way to pass the information selectively that consists of the sigmoid layer, hyperbolic tangent layer, and the point-wise multiplication operation.

Fig. 2 illustrates the architecture of LSTM at time t which is designed to model sequential input. In particular, four gates —output, change, input, and forget —are shown with their operations at time t.

For a given input sequence $\{x_1, x_2, \ldots, x_n\}$, $x_t \in \mathbb{R}^{k\times 1}$ is the input sequence at time t. The memory cell c_t updates the information using three gates: input gate i_t, forget gate f_t, and change gate $\tilde{c_t}$. The hidden state h_t is updated using output gate o_t and the memory cell c_t. At time t, the respective gates and layers compute the following functions:

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + W_{hi} h_{t-1} + b_i),\\
f_t &= \sigma(W_f x_t + W_{hf} h_{t-1} + b_f),\\
o_t &= \sigma(W_o x_t + W_{ho} h_{t-1} + b_o),\\
\tilde{c_t} &= \tanh(W_c x_t + W_{hc} h_{t-1} + b_c),\\
c_t &= f_t \otimes c_{t-1} + i_t \otimes \tilde{c_t},\\
h_t &= o_t \otimes \tanh(c_t)
\end{aligned}
$$

where, σ and tanh represent the sigmoid and hyperbolic tangent functions respectively, the operator $\otimes$ is the element-wise product, $W \in \mathbb{R}^{d\times k}, W_h \in \mathbb{R}^{d\times d}$ are weight matrices, and $b \in \mathbb{R}^{d\times 1}$ are bias vectors. Moreover, n, k, d are sequence length, the number of features, and the hidden size respectively (Greff, Srivastava, Koutník, Steunebrink, & Schmidhuber, 2017; Lei, Liu, & Jiang, 2019; Qiu et al., 2020).

LSTM cell takes 3 different pieces of information: the current input sequence x_t, the short-term memory from the previous cell h_{t-1}, and the long-term memory from the previous cell state c_{t-1} at time t. The forget gate takes the information from x_t and h_{t-1} and produces the output between 0 and 1 through the sigmoid layer and then it identifies which information to discard from the previous cell state c_{t-1}. When the value is 1, it stores all the information into the cell while with a value of 0, it forgets all the information from the previous cell state. Similarly, the input gate identifies which information to be updated from the change gate. The output gate decides which information to be taken as an output from the present cell state.

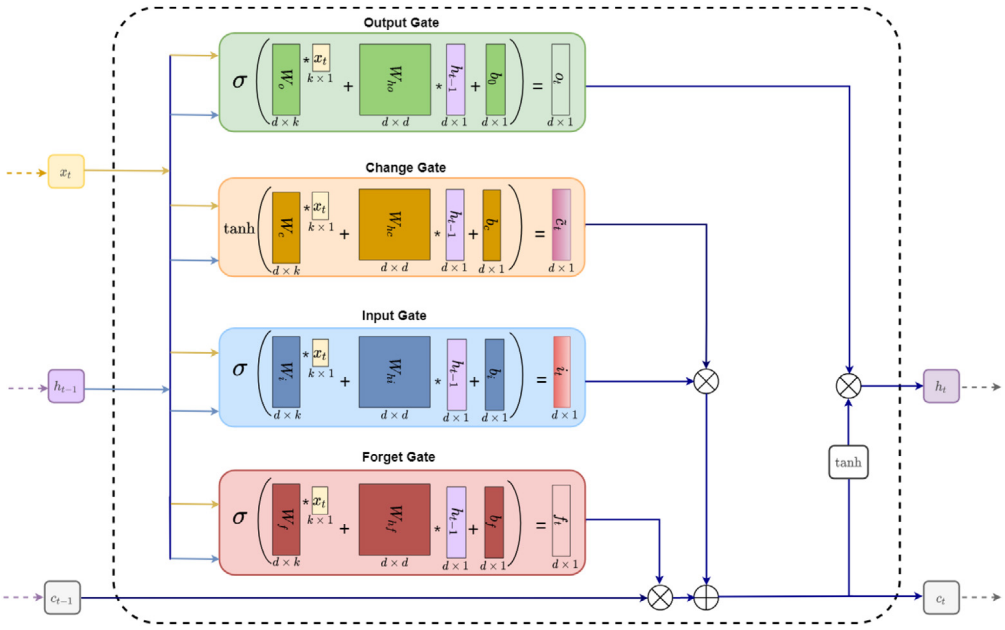

Fig. 2. Long short-term memory (LSTM) architecture.

3.2. Proposed model framework

For a given multivariate financial time series data collected from different sources, the goal of the proposed model is to predict the next day closing price using a multivariate sequence of input features. The following LSTM implementation procedures are considered to accomplish this task. From the original dataset $X = (x_1, x_2, \ldots, x_n)$ of size $k \times n$, the sequences $\{x_1, x_2, \ldots, x_{n-1}\}$ and $\{y_1, y_2, \ldots, y_{n-1}\}$ are created, where $x_t \in \mathbb{R}^{k\times 1}$ is the input sequence and $y_t \in \mathbb{R}$ is the next day closing price at time t. Here k and n are number of input features and the total number of observations respectively.

Furthermore, to incorporate the required dimension of LSTM architecture, input sequence X_t is created by taking m continuous sequence $x_t : x_{t+m-1}$ which is a matrix of shape $k \times m$ for $t \in \{1, 2, \ldots, n-m-1\}$. The output h_t of LSTM is a feature representation for the input sequence X_t at time t. Mathematically, h_t can be expressed as follows:

$$h_t = LSTM(X_t, h_{t-1}, c_{t-1}, w)$$

where w denotes all learnable parameters. Since the final hidden state h_f encodes the most information from the input sequence, it is converted to a vector using a dense layer.

Fig. 3 represents the proposed model framework. The input sequence is created using $k = 11$ features with time step m, as shown in top part of the figure. Then at time $t-1$, the input X_{t-1}, a matrix of size $11 \times m$, together with h_{t-2} and c_{t-2} is fed into the LSTM. For the next step, the output h_{t-1} of the previous step, together with input sequence X_t and the cell memory c_{t-1} become input for LSTM. This process continues until the final input sequence X_f with corresponding output h_f, a vector of length equal with given number of neurons of the last LSTM layer. Finally, h_f is transmitted to a fully connected layer where linear activation function is used to predict the closing price as shown in the last part of Fig. 3.

3.3. Algorithms and pseudo code

Algorithm 1 (*Pseudo Code for Hyperparameter Tuning Procedure*). *Input Preparation: Split train and validation data sets and create input of the form [#observations, time step, #features]*
Input: [#observations, time step, #features]; choices of optimizers, learning rates, and batch sizes.
Initialize: Set number of epochs sufficiently large and patience = 5

For *"choice of optimizers"*, **Do**
  For *"choice of learning rates"*, **Do**
    For *"choice of batch sizes"*, **Do**
      For *"range of number of replicates"*, **Do**
        Train the model, monitor validation loss;
        Continue *Until training loss at epoch $n \le$ training loss at epoch $n+1 \le \cdots \le$ training loss at epoch $n+4$, Or maximum epochs are reached.*
        Evaluate model on the validation data.
        Calculate RMSE scores.
      End Do.
      Calculate average RMSE scores.
    End Do.
  End Do.
End Do.
Output *Set of best hyperparameters, average RMSEs, best average RMSE.*

Algorithm 2 (*Pseudo Code for LSTM Model after Hyperparameter Tuning*). *Input Preparation: Split train and test data sets and create input of the form (#observations, time step, #features)*
Input: [#observations, time step, #features]; chosen hyperparameters (optimizer, learning rate, batch size) obtained from Algorithm 1 for each model.
Initialize: Set number of epochs sufficiently large and patience = 5

For *"choice of layers and neurons"*, **Do**
  For *"range of number of replicates"*, **Do**
    Train the model, monitor training loss;
    Continue *Until training loss at epoch $n \le$ training loss at epoch $n+1 \le \cdots \le$ training loss at epoch $n+4$, Or maximum epochs are reached.*
    Evaluate model on the test data.
    Calculate RMSE, MAPE and R scores.
  End Do.
  Calculate minimum, maximum, average and standard deviation of RMSE, MAPE and R scores.
  Save key results in respective files.
End Do.

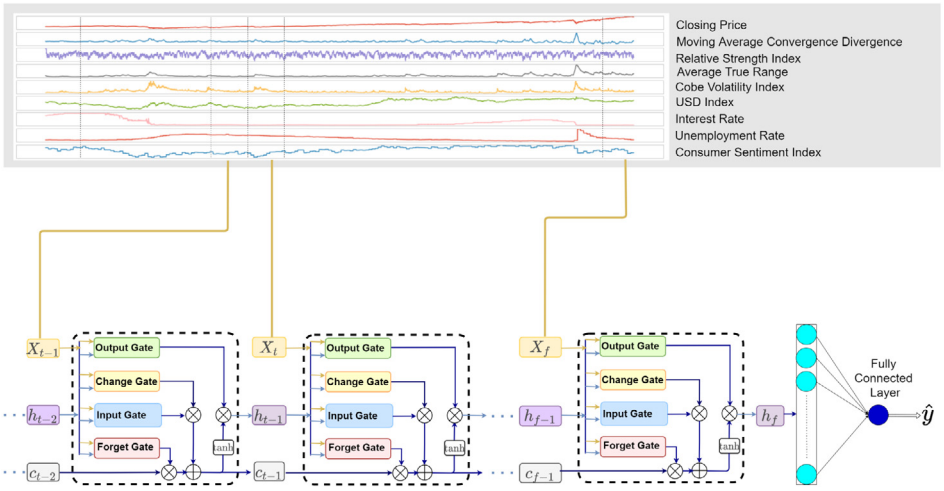

Fig. 3. The proposed model architecture for index price prediction.

Table 1
List of potential features for the model.

Data	Source	Frequency	Abbreviation
Fundamental			
Open price	Yahoo	Daily	···
Close price	Yahoo	Daily	···
Macroeconomic			
Cboe volatility index	Yahoo	Daily	VIX
Interest rate	FRED	Daily	EFFR
Civilian unemployment rate	FRED	Monthly	UNRATE
Consumer sentiment index	FRED	Monthly	UMCSENT
US dollar index	Yahoo	Daily	USDX
Technical indicator			
Moving average convergence divergence	···	Daily	MACD
Average true range	···	Daily	ATR
Relative strength index	···	Daily	RSI

4. Dataset preparation

In this study, S&P 500, a popular US stock market index, is used for the model prediction. The process of feature selection includes identifying the core factors that contribute to the index value fluctuations. Some features, such as fundamental data and technical indicators, are directly extracted from the underlying index. Other factors, namely macroeconomic variables, are selected based on their potential impact on the overall economy and broader markets. A complete 15 years of data have been collected from 2006 to 2020. The time frame selection incorporates two major bear markets, the financial crisis in 2008 and the COVID-19 pandemic in 2020. Thus, the construction of the model, including both bear and bull market, resembles the overall market scenario and may lead to a better prediction.

We start with a brief description of the features used in the proposed model. The closing price is predicted based on the fundamental trading data, macroeconomic data, and technical indicators of the underlying index. A combination of all the features from three different categories presented in Table 1 is input features. All the available features except Civilian Unemployment Rate and Consumer Sentiment Index are by default daily data. We have converted the monthly data to daily through the forward filling method to have uniformity among the variables.

4.1. Fundamental data

The first set of variables presented in Table 1 are fundamental data or historical data which provides basic information required for stock trading. It consists of open price, and the close price. Open price is the first transaction price upon the opening of a market on a trading day, whereas the closing price is the last price at which the stock is traded during that day. All the historical trading data accessed from Yahoo are daily data.

4.2. Macroeconomic data

The second set of variables demonstrated in Table 1 are macroeconomic variables that significantly influence stock market performance, explaining more potential information in stock price prediction. We have chosen Cboe Volatility Index (VIX), Interest Rate (EFFR), Civilian Unemployment Rate (UNRATE), Consumer Sentiment Index (UMCSENT), and US dollar index (USDX) under the macroeconomic factor. These variables are the representative features that explain the status of the economy as a whole in the proposed model.

VIX, sometimes called an investor's fear index, measures the 30-day expected volatility of the stock's market based on the S&P 500 index options. It is maintained by the Chicago Board Options Exchange, the largest derivatives (options) exchange in the US. There is an asymmetric relationship between VIX and stock market returns (Chandra & Thenmozhi, 2015; Sarwar, 2012). VIX indicates how investors and equity traders are thinking about the overall health of the current economy and near-term market fluctuations based on the options market activities. Therefore, it can provide a significant contribution to the proposed predictive model (Ruan, 2018). The real-time VIX index data is publicly available on different platforms, but this study uses the data from Yahoo Finance.

The federal funds rate is the interest rate at which financial institutions, also known as depository institutions, trade federal funds with each other overnight. These federal funds are balances held at Federal Reserve Banks. If a depository institution has a surplus in balances in its reserve account, it will lend to other institutions that need a larger sum of balances with the interest rate negotiated between them. The weighted average of all these types of interest rates is called EFFR. The 3-month Treasury bill rate and federal fund rate are two key US money market interest rates (Sarno & Thornton, 2003). Thus, EFFR affects the various US economic conditions; including inflation, growth, and employment. Bernanke and Kuttner (2005) studied the effects of the changes in monetary policy on equity prices. The authors concluded that the decrease in federal fund interest rate increase the broad stock price. The interest rate has a significant impact on corporate profitability and margin transactions, thus, it influences the stock price. The EFFR is calculated daily by the Federal Reserve Bank of New York. We accessed the data from the Federal Reserve Bank of St. Louis. The published rates are the volume-weighted median transacted rate, rounded to the nearest basis point.

UNRATE measures the percentage of the US labor force unemployed in the economy. An upward trend in UNRATE reflects the downtrend in the economy, whereas its downward trend signifies the healthy and growing economy. It has been used for the stock price prediction and is considered a significant predictor of stock price (Farsio & Fazel, 2013; Loungani, Rush, & Tave, 1990; Pan, 2018). The authors of the article (Bock, 2018) assert that UNRATE strongly affects the stock market and further investigate the possibility to construct a profitable investment strategy before it is officially published. The monthly data of UNRATE is accessed from the FRED.

UMCSENT is the monthly data accessed from the FRED. It is calculated based on a household survey of consumers' opinions on current economic conditions and the index of the future expectations. It is used to evaluate short-term views of customers on the economy, their income, and spending. Thus, it provides the consumer's level of optimism or pessimism that helps to predict the short-term and long-term economy. The UMCSENT with momentum factors plays a vital role in the stock price prediction (Benhabib, Wang, & Wen, 2015; Lansing & Tubbs, 2018; Otoo, 1999). It has direct (Baker & Wurgler, 2007) or indirect (Stambaugh, Yu, & Yuan, 2012) effect on stock returns.

The US Dollar Index (USDX) is a geometrically averaged calculation of six major world currencies weighted against the US dollar (Samanta & Zadeh, 2012). International Currency Exchange compiles, maintains, and weights the components of the index. It measures the strength of the US dollar relative to the world's major currencies. The index value is affected by economic conditions in the US and abroad. Since the US currency is a new gold standard for the inter-continental trades and currency exchanges, the index value also fluctuates with the interest rate policy, foreign investment policy, and volatility of the overall markets. Generally, the value of US equities tends to increase along with the demand for the US dollar; therefore, they have a positive correlation (Novianti, 2016). The world economy is directly or indirectly interconnected with the US economy. For instance, the exposures of the US-listed companies globally for their supply-chain targeted consumers and investment options, the fluctuation of dollar index value can significantly contribute to the proposed model. There are several tickers available for the index with a slightly different basis point. For this study, the index with ticker DX-Y.NYB is obtained from Yahoo, which is available daily.

4.3. Technical indicators

The last set of variables demonstrated in Table 1 are the technical indicators, including Moving Average Convergence Divergence (MACD), Average True Range (ATR), and Relative Strength Index (RSI). A technical indicator is a mathematical calculation performed on variables like price or even another technical indicator. Active traders extensively use them in the market as they are primarily designed to analyze short-term price movements.

MACD, developed by Gerald Appel, is one of the widely applied technical indicators (Wang & Kim, 2018). It is a momentum oscillator calculated by subtracting the 26-day exponential moving average (EMA) from the 12-day EMA (Murphy, 1999). EMA differs from a simple moving average as it assigns more weight to the later data values. Traders use the MACD for determining the trend, direction, momentum, and potential reversals of the stock price. For example, if the MACD curve crosses the zero line from below, it signals the buying opportunity. Similarly, when it crosses the zero line from above, it signals a sell. Research results indicate significant possibilities of MACD indicator for making optimal investment decisions (Chong & Ng, 2008; Chong, Ng, & Liew, 2014; Eric, Andjelic, & Redzepagic, 2009). Anghel (2015) used data from different countries to evaluate market efficiency worldwide using MACD.

J. Welles Wilder Jr. has been credited for developing many technical indicators of the stock market. We consider the most popular volatility-based technical indicators, ATR and RSI (Wilder, 1978). ATR describes how volatile a stock has been on average over a particular period of time. Stock traders use this indicator as a risk management strategy to set the exit level. It can also be useful to sense how strong price moves are, which is beneficial to identify the start of the trend. ATR is based on the concept of true range, which is a way of measuring a stock's trading range that elucidates for gap openings. The gap indicates the stock's opening higher or lower relative to the previous day's closing price. Averaging out a stock's daily true range over a specified period provides ATR. Commonly used ATR periods are 14, 20, and 22 days. This study considers the 14 days period, which is also considered the default period in many platforms. Incorporating ATR to gauge the stock moves certainly adds some level of contribution in predicting the closing price.

RSI is a recognized technical indicator in stock price prediction (Rodríguez-González, García-Crespo, Colomo-Palacios, Iglesias, & Gómez-Berbís, 2011). It is an oscillating indicator constructed to measure the stock's momentum to provide bullish and bearish price signals. Investors use this indicator to identify security overbought or oversold. RSI may be helpful to determine potential entry and exit trading signals as well. In addition, RSI plays an essential role in predicting the stock price trend for a shorter time, as it compares the magnitude of recent gains to recent losses.

4.4. Feature selection strategy

All the input variables explained above have some level of contribution in predicting the closing price. A correlation heatmap of all input features explained above is presented in Fig. 4. The numerical value in the heatmap represents the correlation between the variables on the horizontal and the vertical axes. For instance, the diagonal value of the matrix is 1, the correlation between the variable to itself. Thus, the values on the diagonal are unimportant in our analysis. The entries on the off-diagonal are used for the feature selection process. These values are presented based on the intensity of the color, which is also the indicator of the level of relationship between the given variables. The vertical bar next to the graph shows the intensity of the color on the scale from 0 to 1. The correlation between the closing price and the remaining variables may be high or low. It indicates the intensity of the relationship. For example, the correlation between closing price to consumer sentiment index is 0.67, which is moderately positive. Thus, the consumer sentiment index may play a vital role in the closing price prediction. The same explanation applies to the rest of the entries presented in the graph. Most importantly, the high correlation (positive or negative) between the predictors can behave as a duplicate feature. In such case, either one of the highly correlated variables can be dropped from the analysis. For instance, open price and closing price have a strong correlation, indicating duplicate features. One of the features can be discarded because it will not provide significant additional information for the prediction. In this study, the open price has been dropped from further analysis. The correlation coefficient of 0.80 is considered as a threshold for removing the duplicate features. After removing the open price, none of the pair-wise correlations between the remaining variables exceeds the threshold. The heatmap validates the statistical significance of the variables chosen for the prediction.

After careful analysis of heatmap in the feature selection process, the 11 variables are considered as input variables. The partial snapshot of the dataset used in the study is presented in Table 2.

4.5. Data denoising, normalization and input preparation

Stock price data are noisy and are in sequential discrete format. The discrete Wavelet transformation is common to denoise time-series data. Haar wavelets are the most suitable and popular in stock price data (Chaovalit, Gangopadhyay, Karabatis, & Chen, 2011; Ortega & Khashanah, 2014). We have applied the soft mode of the Haar wavelets using python library scikit-image to denoise the close price of the index. The values of input variables vary from one to another; thus, it leads

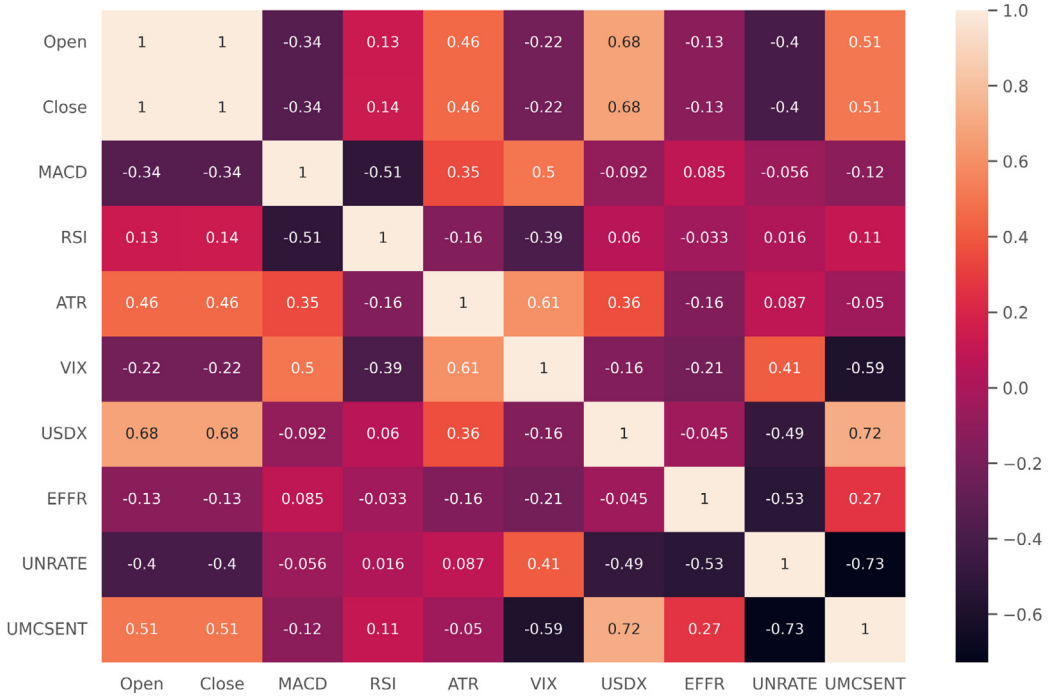

Fig. 4. Correlation heatmap among the attributable variables. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Snapshot of the dataset.

Date	Close	MACD	RSI	ATR	VIX	USDX	EFFR	UNRATE	UMCSENT
2006-01-24	1266.8599	−0.1018	38.3361	14.9114	13.31	88.0599	4.28	4.7	91.2
2006-01-25	1264.6801	0.7335	36.1706	14.7356	12.87	88.3300	4.36	4.7	91.2
2006-01-26	1273.8299	0.6497	49.8817	14.5230	12.42	88.6200	4.37	4.7	91.2
2006-01-27	1283.7199	−0.2123	60.4715	14.3821	11.97	89.3200	4.42	4.7	91.2
2006-01-30	1285.1899	−1.0026	61.8538	13.6712	12.39	89.4199	4.48	4.7	91.2
...	...	...	...	...	...	...	...	...	...
2021-09-28	4419.5400	14.4707	32.4778	47.4541	23.5000	93.7700	0.08	4.7	72.8
2021-09-29	4362.4102	18.7500	34.4857	46.4174	22.5600	94.3400	0.08	4.7	72.8

to a high level of variation. For instance, the stock index close price is much higher than the interest rate. More specifically, the standard deviation of the close price is 695.33, which is significantly higher than the standard deviation 1.664 of the interest rate. If the range of one feature varies more widely than the others, most ML algorithms might not perform well. We have implemented a min–max normalization technique for the feature scaling to address this concern. The min–max normalization technique is expressed in the following equation,

$$z = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where, z, x are scaled and the original input respectively. Similarly, x_{min} and x_{max} are the minimum and the maximum values of the input respectively.

The transformed data at this point is two-dimensional array (number of observations, number of features). LSTM model expects three-dimensional input data (number of observations, size of time step, number of input features). Therefore, the necessary steps are considered to make the input data compatible to the model. Due to the nature of time series data, first, we split the complete data into 80%–20% for the train-test datasets by maintaining the order of time series. We further divide the training data as 80%–20% of which the last 20% (accounting 16% of the total data) is used for validation purpose during hyperparameter tuning. Once we are done with the hyperparameters tuning, the validation set is included back in the training data. Then the final models are fitted on the complete training data with optimized hyperparameters. Finally, the performance scores are reported on the test data.

5. Experiment and results

As described in the previous section, normalized data has been constructed with the selected features. Also, necessary steps are taken to reshape and split the data into training and testing data sets. The goal is to predict the closing price of the S&P 500 index with high accuracy, which exhibits complex, noisy, and volatile behavior as shown in Fig. 5. The black curve represents the original time series of the closing price (vertical axis) within the interval of 01/03/2006–09/29/2021 (horizontal axis). Similarly, blue and green curves represent 50-day and 200-day moving averages to visualize the short-term and long-term trends of the closing price. The closing price's overall direction is upward despite having various irregularities.

5.1. Model performance metrics

We implement single layer and multilayer LSTM architecture to predict the closing price. Within each of these models, several options are considered with different number of neurons. Prediction accuracy and reliability of these models are assessed by calculating three different performance metrics —RMSE, MAPE, and R. The analytical form of these metrics are defined as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2},$$

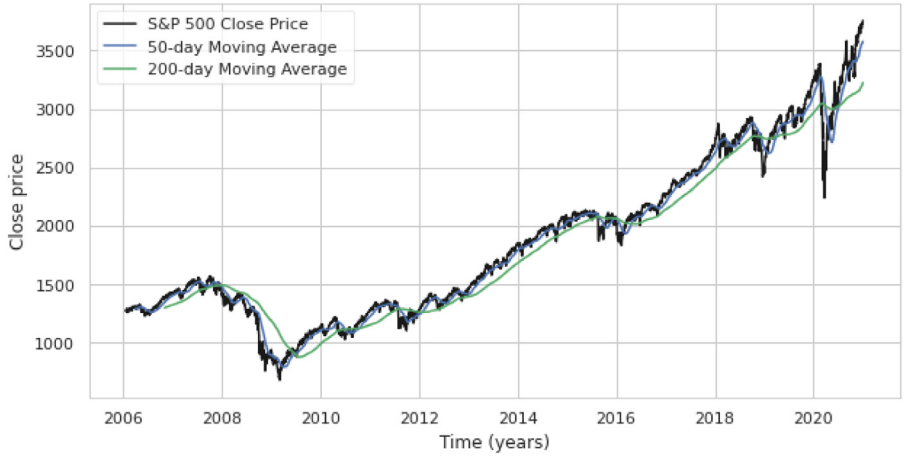

Fig. 5. S&P 500 closing price along with moving averages. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
Computing environmental condition.

Machine configuration	Google Colab with NVIDIA-SMI 495.44 GPU
Environment	Python 3.6.0, TensorFlow, and Keras APIs
Architecture	Single layer and multilayer LSTM

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right|,$$

$$R = \frac{\sum_{i=1}^{N}(y_i - \bar{y}_i)(\hat{y}_i - \bar{\hat{y}}_i)}{\sqrt{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2(\hat{y}_i - \bar{\hat{y}}_i)^2}}$$

where,

y_i : Original time series,

$\bar{y}_i$: The average value of the original time series,

$\hat{y}_i$: Predicted time series computed from the model,

$\bar{\hat{y}}_i$: Average value of the predicted time series,

N : Number of observations.

Among three prediction metrics, RMSE measures the square root of the mean square error of the actual values and estimated values, MAPE estimates the size of the error computed as the relative average of the error, and R determines the linear correlation between actual and predicted values. Smaller the values of RMSE and MAPE, better the performance of the model. On the contrary, larger value of R indicates the similarity between predicted and actual series. Moreover, performance scores are calculated after applying the inverse transformation in the predictions obtained from the normalized data. Each model are executed multiple times independently in order to address the stochastic behavior. Average RMSE score obtained from these multiple replicates is considered as a primary model selection criteria followed by average MAPE and R scores. A model with the smallest RMSE and MAPE along with the greatest possible R would be considered as the best model.

Table 3 summarizes the experimental environment for this study. The experiment uses python programming environment along with TensorFlow and Keras APIs. All the experiments are conducted in a machine configuration as stated in Table 3.

5.2. Hyperparameter tuning

The final optimal model architecture is selected by exploring a wide range of possibilities. The overall model selection procedure is divided into two broad categories —(a) single layer LSTM and (b) multilayer LSTM. The single layer LSTM model consists of a only one LSTM layer in the model whereas the multilayer LSTM model consists of more than one LSTM layer in the model. For each of these categories, the process is further divided into two phases —(i) tuning hyperparameters and (ii) training the model in full scale with the respective optimized hyperparameters. In the first phase, the values of hyperparameters —optimizer, initial learning rate, and batch size —are optimized using the validation data. For the single layer LSTM architecture, six different models with 10, 30, 50, 100, 150, and 200 neurons are experimented with optimizers Adam, Adadelta, and Nadam; learning rates 0.1, 0.01, and 0.001; and the batch sizes 4, 8, and 16; resulting 27 different combinations of hyperparameters for each model. Every model is executed 10 times for each combination of hyperparameters and the average RMSE score is calculated. The best possible combination of hyperparameters for the given model is chosen based on the lowest average RMSE score on the validation data. Table 4 provides the list of the best hyperparameters for all six single layer LSTM models. Furthermore, the complete validation results of all 27 different combinations for the 10 neurons model are provided in the Table 5. This table shows that the lowest average RMSE value is 28.65, which corresponds to the Adam optimizer with the learning rate of 0.001 and the batch size 8. Hence the Adam optimizer with the learning rate of 0.001 and batch size 8 is chosen to train the 10 neuron model in the full scale. The detailed validation results for remaining models are provided in the Tables A.10–A.14 in Appendix A.

Similarly, the best hyperparameters are identified for six different multilayer LSTM models with $(10, 5)$, $(20, 10)$, $(50, 20)$, $(100, 50)$, and $(100, 50, 20)$ neurons. Here, the first and second hidden layers is represented as (n_1, n_2). The same explanation is applicable as the number of hidden layers increases. The list of the best hyperparameters for each multilayer model can be found in Table 6 and the complete validation results of all 27 different combinations for the $(10, 5)$ neurons model are provided in the Table 7. The validation results for other models are presented in Tables B.15–B.19 in Appendix B.

5.3. Single layer LSTM results

In the previous section, we have discussed about the process of identifying the best hyperparameters for each model with an extensive, thorough, and data-driven approach. Using the best hyperparameters presented in Table 4, we further train all six single layer models with

Table 4
List of the best hyperparameters for single layer LSTM models.

No. of Neurons	Optimizer	Learning rate	Batch size
10	Adam	0.001	8
30	Adagrad	0.01	8
50	Adagrad	0.01	8
100	Adagrad	0.01	16
150	Adagrad	0.01	16
200	Adagrad	0.001	4

Table 5
Hyperparameter tuning for 10 neurons single layer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	33.54	33.32	32.94
	0.01	30.55	30.01	29.61
	0.001	28.85	**28.65**	29.10
Adagrad	0.1	28.82	28.69	28.91
	0.01	29.92	31.12	33.19
	0.001	38.95	45.28	51.36
Nadam	0.1	51.09	53.71	56.80
	0.01	55.15	54.53	54.35
	0.001	53.33	52.48	51.59

Table 6
List of the best hyperparameters for multilayer LSTM models.

No. of neurons	Optimizer	Learning rate	Batch size
(10, 5)	Adagrad	0.1	4
(20, 10)	Adagrad	0.01	16
(50, 20)	Adagrad	0.01	16
(100, 50)	Adagrad	0.01	16
(150, 100)	Adagrad	0.01	16
(100, 50, 20)	Adagrad	0.001	8

Table 7
Hyperparameter tuning for $(10, 5)$ neurons multilayer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	51.00	57.27	53.42
	0.01	46.77	42.67	39.88
	0.001	38.13	36.67	35.84
Adagrad	0.1	35.17	34.70	34.79
	0.01	34.52	**34.42**	36.11
	0.001	39.10	42.29	46.19
Nadam	0.1	50.46	54.28	57.43
	0.01	56.31	55.92	56.03
	0.001	54.75	53.43	52.72

the training data set (i.e. 80% of the total data set) and select the best model on the basis of average performance scores calculated on the test data set. Each experiment is replicated a sufficiently large number of times (30 times) to maximize model reliability. The results suggest that the model with 150 neurons outperforms its competitors in RMSE, MAPE, and R as shown in Table 8. Thus, a single layer LSTM model with 150 neurons can be considered as the winner among other candidates.

Fig. 6 illustrates the graphical representation of the data entries of average scores via line graph. The subplots (a) and (b) reveal the overall patterns of average RMSE and MAPE scores. From the figure, we observed that the average RMSE and MAPE scores increase until the neurons reaches 30 and then gradually decrease until 150 neurons. Average scores show increasing trend afterwards. The average R is going down while average RMSE and MAPE scores increases on the range of neurons from 10 to 30, then it increases steadily until the neurons reaches 150, there is a drop in the average R score afterwards as shown in subplot (c). Looking at the overall trend of all the performance metrics, a model with 150 neurons seems to be the best among the single layer models.

Table 8
The performance scores of the single layer LSTM models in the test data.

Metrics	Neurons →	10	30	50	100	150	200
RMSE	Min	34.7359	43.8253	38.5586	37.2795	37.9416	62.1324
	Average	49.9564	57.0731	47.1908	42.7093	**40.4574**	73.1992
	Max	77.4861	72.1660	60.7464	49.4979	43.4026	88.8964
	Std	9.7758	8.0805	4.9642	2.9514	1.3957	5.3066
MAPE	Min	0.7511	0.8959	0.7657	0.7287	0.7008	1.5401
	Average	1.1264	1.2375	0.9759	0.8691	**0.7989**	1.856
	Max	1.6124	1.5081	1.2409	1.1089	0.9768	2.3210
	Std	0.2513	0.1762	0.0995	0.0912	0.0584	0.1586
R	Min	0.9958	0.9946	0.9967	0.9969	0.9974	0.9903
	Average	0.9972	0.9962	0.9972	0.9974	**0.9976**	0.9937
	Max	0.9983	0.9973	0.9977	0.9978	0.9979	0.9953
	Std	0.0006	0.0007	0.0003	0.0002	0.0001	0.0010
No. of parameters		891	5071	12451	44901	97351	169801

Figs. 7 and 8 visualize the quality of the prediction obtained from the best model with 150 neurons. The scatter plot of the true values versus the predicted values of closing price for training and test data are plotted in Fig. 7(a) and (b) respectively. This plot is useful to gauge the goodness of fit of the model. The best-fit linear equation ($y = x$) is shown in Fig. 7 by the red dotted line. The performance of the best model is slightly better in the training set compared to the test set, which is as expected. In the test data, the predicted closing price is deviated a bit from the true closing price in the range of 2400 to 3200, possibly due to the unusual market circumstances created by the COVID-19 pandemic in 2020.

Fig. 8 represents the original closing price together with predictions obtained from the replication with the lowest RMSE score of the best single layer model. In subplot (a), the black curve represents the actual values of the closing price, whereas the blue and the green curves represent the predictions in the training and test data, respectively. Subplot (b) is the magnified portion of the subplot (a) that consists of the true and predicted closing price on the test data. The prediction curve of the closing price for training data almost overlaps with the curve of the true closing price, as shown in subplot (a). This suggests that the best model can learn both upward and downward movement of the original closing price almost exactly. The quality of the fit is also impressive in the test data, as demonstrated in the later portion of the subplot (a) or in a magnified subplot (b). Although the test prediction curve does not seem to overlap exactly with the actual closing price, the model captures the overall trend of the test data with minor errors. Moreover, Fig. 8(b) depicts that the model is well fitted even in the unusual market circumstances, where there is a sudden significant drop in the market and then a sharp V-shaped recovery. This particular section of the test data corresponds to the year 2020, when the COVID-19 pandemic badly impacted the stock markets, thus making it highly volatile. The ability to predict such complex and noisy market fluctuations further validates the robustness and resilience of the proposed model. This suggests that the model does not suffer from overfitting and is well-suited for predicting the out-of-sample data.

5.4. Multilayer LSTM results

The prediction results obtained from single layer LSTM architecture discussed in previous section suggest that a model with 150 neurons seems to produce impressive predictions. Even though single layer LSTM architecture provides compelling results, we want to explore the possibility of further improvements via multilayer LSTM architecture. The primary objective of this investigation is to improve the prediction accuracy in conjunction with maintaining model simplicity.

Using the best hyperparameters presented in Table 6, we train all six multilayer LSTM models with the training data set and select the best model on the basis of average performance scores calculated on the test data set. Each experiment is replicated 30 times. The data entries presented in Tables 9 are the performance metrices of multilayer LSTM

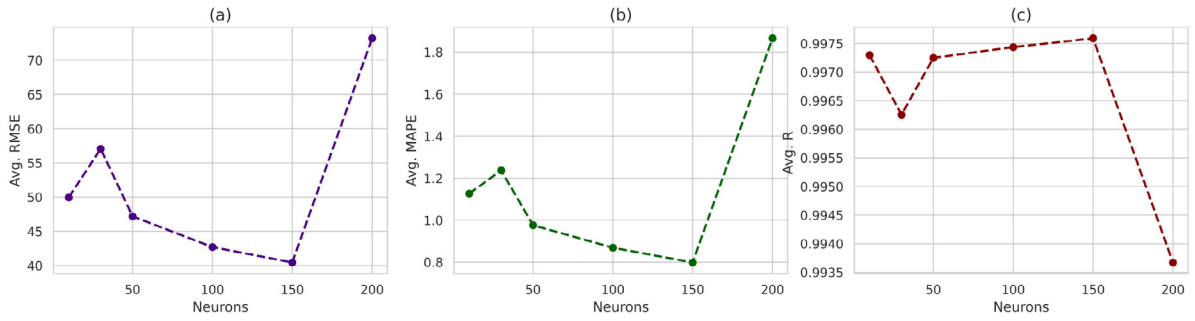

Fig. 6. The line plot of the performance metrics of the single layer LSTM models with different neurons.

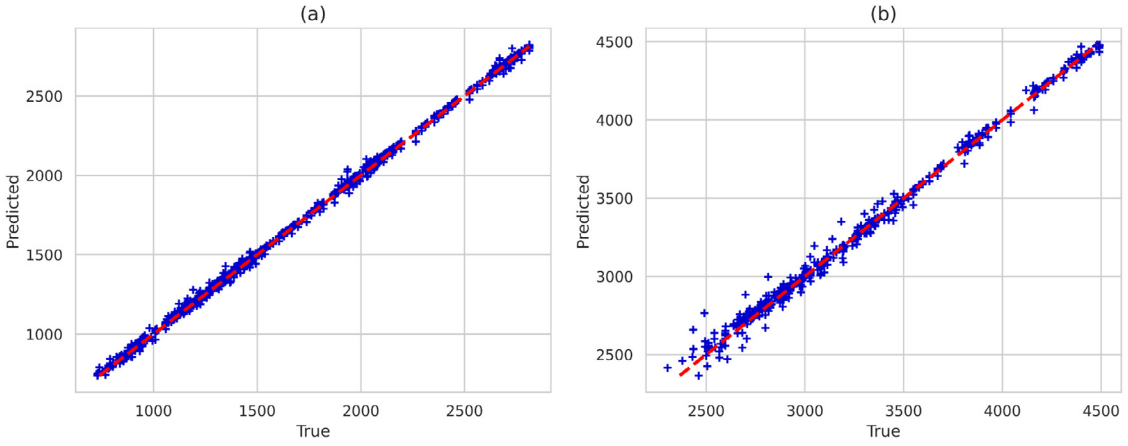

Fig. 7. Scatter plot of true vs predicted closing price of the best single layer LSTM model (with 150 neurons) on (a) training data and (b) test data.

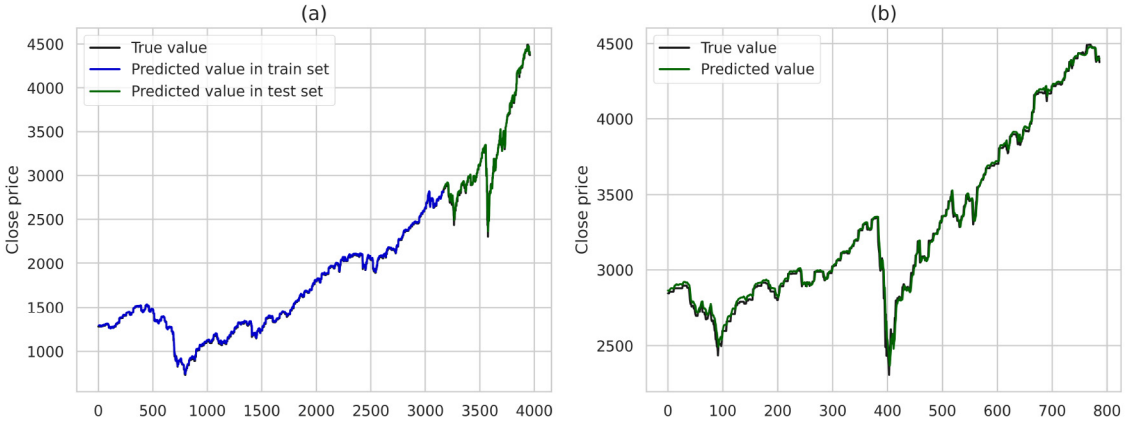

Fig. 8. Plots of the true closing price together with its predictions on the training and test data obtained from the replicate with lowest RMSE of the best single layer LSTM model. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

models. The table shows that the model with (150, 100) neurons is the best among candidates for the multilayer LSTM. Moreover, it reveals the fact that multilayer LSTM does not improve the performance as compared to single layer LSTM. The possible reason for not improving the result could be over-fitting or the complexity in the model architecture. To address the such concern, we also implemented a dropout strategy to improve further the model performance where 10% of the neurons are frozen after each hidden layer. However, the performance did not improve as compared to the original multilayer models.

5.5. Comparison of single and multilayer LSTM models

As we discussed in the previous sections, single and multilayer LSTM models were implemented to predict the closing price. In this section, we analyze the performance of these two LSTM architectures by using the comparative boxplots of performance metrices obtained from 30 replicates. Figs. 9 and 10 provide boxplots of the average performance scores obtained from the single and multilayer models respectively. As we see in the boxplots, evaluation metrics generally follow similar patterns in both single and multilayer LSTM models. The median scores of RMSE and MAPE are increasing but that of metric R is decreasing consistently until the number of neurons reaches to 30 in single layer and (20, 10) in multilayer models. Then, the median scores of RMSE and MAPE are decreasing but that of metric R is increasing consistently until the number of neurons reaches to 150 in single layer and (150, 100) in multilayer models. Afterwards, we noticed the significant jump in the RMSE and MAPE scores in both single layer and multilayer models and the large drop in R scores. In this case, complex models

Table 9
The performance scores of the multilayer LSTM models in the test data.

Metrics	Neurons →	(10, 5)	(20,10)	(50,20)	(100, 50)	(150, 100)	(100, 50, 20)
RMSE	Min	47.8386	58.8881	49.1501	47.6658	46.4954	167.5684
	Average	61.8187	86.9894	67.1374	53.9076	**49.8362**	197.2483
	Max	83.6141	114.2188	92.3948	59.1283	52.6221	228.9790
	Std	9.9735	14.7349	8.4727	2.8917	1.8095	13.9473
MAPE	Min	0.9549	1.2690	1.006	0.9668	0.9108	4.4809
	Average	1.2740	1.9661	1.4671	1.1430	**1.0269**	5.4067
	Max	1.9123	2.7574	1.9676	1.2971	1.1351	6.3168
	Std	0.2016	0.3929	0.1948	0.0822	0.0580	0.4039
R	Min	0.9935	0.9838	0.9925	0.9957	0.9959	0.9339
	Average	0.9956	0.9913	0.9949	0.9962	**0.9964**	0.9542
	Max	0.9967	0.9960	0.9966	0.9966	0.9967	0.9669
	Std	0.0008	0.0026	0.0009	0.0002	0.0001	0.0083
No. of parameters		1206	3811	18,101	75,051	197,701	80,701

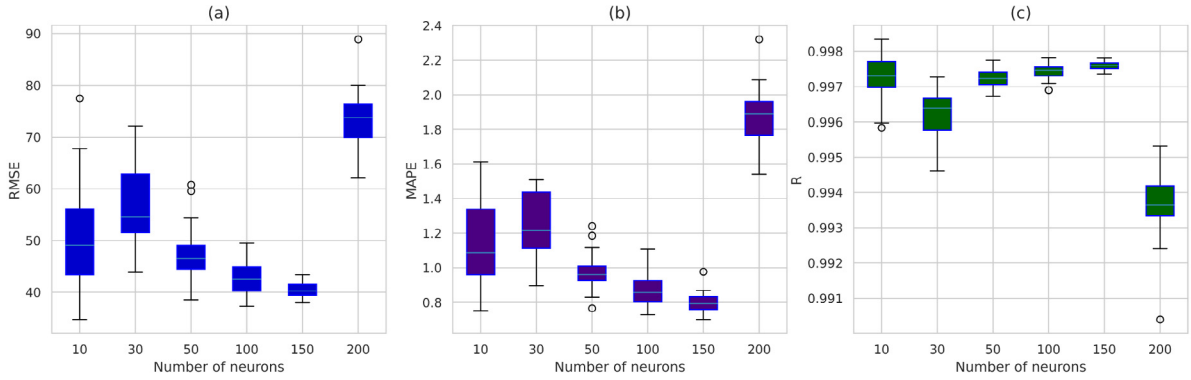

Fig. 9. Box plots of the performance scores obtained from single layer LSTM models with 30 replications.

provide even worse results in the out-of-sample data as compared to the corresponding simple versions. This shows that a single layer LSTM model with around 150 neurons and a multilayer LSTM model with around (150, 100) neurons can be considered as the best models in their respective categories.

More interestingly, the median scores of all evaluation metrics in the single layer LSTM models are better than that of the multilayer models. For instance, median score of RMSE obtained from the single layer models consistently fall within the range of 40–55 except for the last model which has median score above 70. On the other hand, almost all of these values obtained from the multilayer models are well above 50, suggesting inferior performance compared to the single layer models. In Fig. 11, we observe that the worst value among the 30 replicates of best single layer model is much smaller than the best RMSE scores among 30 replicates of the best multilayer model. In terms of RMSE metric, the single layer LSTM model is superior than multilayer LSTM model. Similar conclusions can be drawn from the boxplots of MAPE and R scores, and in each case, the single layer models outperform the multilayer competitors. In nutshell, a single layer LSTM model architecture having 150 neurons is more reliable to predict the closing price with high accuracy because it outperforms all other models.

5.6. Statistical analysis

The RMSE, MAPE, R, and the visualization through boxplots discussed above indicate that the performance of the LSTM model with a single hidden layer performs better than the LSTM model with multiple hidden layers. We further like to validate the fact using a statistical test. In other words, we would like to test statistically whether the average RMSE from the best single-layer LSTM model with 150 neurons is significantly better than the average RMSE of the best multilayer LSTM with (150,100) neurons. Since the RMSEs are independent and quantile quantile (QQ) plots in Fig. 12 show that they are normally distributed, we use Welch's two-sample t-test to test the hypothesis that two populations have equal means. The test statistic and the p-value of the Welch t-test are −22.2387, and 7.0847e−29 respectively. We reject the null hypothesis since the p-value of the test is almost zero. Hence, based on the Welch t-test, there is sufficient evidence to claim that the performance of the LSTM model with a single hidden layer with 150 neurons performs significantly better than the LSTM model with multiple hidden layers with (150, 100) neurons.

6. Ethics and implications

In this study, we have used publicly available data, and it is not modified except as explained in the paper. The results can be used as additional information to make an investing decision. However, the investment decision should not be solely based on this research. Investors are encouraged to perform their due diligence and consider their risk tolerance in various market conditions. The rational forecast depends not only on the outcome of the specific model but also on the volatile nature of the stock market, especially during geopolitical tension, the global supply chain disturbance, war, pandemic, and various other situations. If the market's current behavior is appropriately analyzed and amalgamated with the model's outcome, there is a good possibility of making profit.

The ability to precisely predict the stock price and consequently project the estimated return is the "dream" of equity traders, individual investors, and portfolio managers. This research shows the promising possibility of the LSTM neural network to delineate the cone of uncertainty in stock price prediction. As a result, stakeholders can use the results as additional information to make investment decisions. Moreover, academic researchers can use the model to enhance their knowledge in sequential data modeling.

7. Conclusion and future work

Stock price prediction is the area of high interest for equity traders, individual investors, and portfolio managers. However, precise and

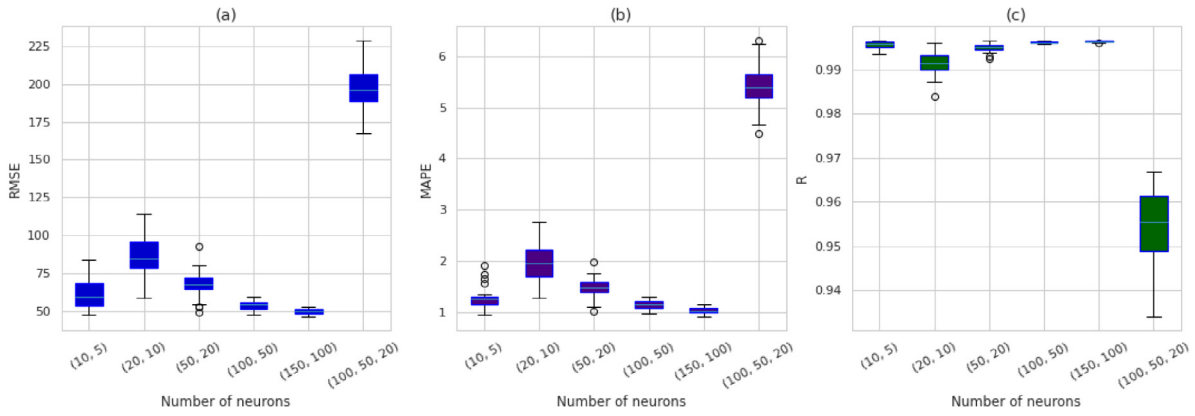

Fig. 10. Box plots of the performance scores obtained from multilayer LSTM models with 30 replications.

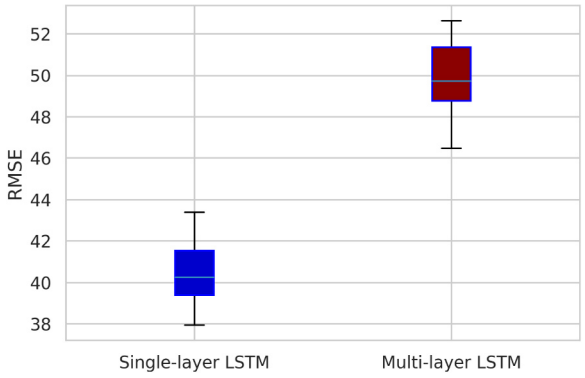

Fig. 11. Box plots of the RMSE scores of the best model obtained from single layer and multilayer LSTM models with 30 replications.

Table A.10
Hyperparameter tuning for 30 neurons single layer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	46.51	39.82	37.13
	0.01	33.37	31.89	31.84
	0.001	31.27	30.50	29.72
Adagrad	0.1	29.09	28.33	27.87
	0.01	27.40	**27.22**	27.93
	0.001	31.16	35.04	39.94
Nadam	0.1	42.76	45.09	47.70
	0.01	46.50	46.15	46.70
	0.001	45.84	45.05	44.33

consistent stock price prediction is a difficult task due to its noisy and nonlinear behavior. There are several factors that can impact the prediction such as fundamental market data, macroeconomic data, technical indicators, and others. This study focuses on developing LSTM based models to predict S&P 500 index's closing price by extracting a well-balanced combination of input variables capturing the multiple aspects of the economy and broader markets. Both single and multilayer LSTM architectures have been implemented and their performances are analyzed by using various evaluation metrics to identify the best model. The experimental results show that single layer LSTM model with around 150 hidden neurons can provide a superior fit and high prediction accuracy compared to multilayer LSTM. The proposed model can be easily customized to apply in other broad market indexes where the data exhibits a similar behavior. Interested stakeholders can use the proposed model to better inform the market situation before making their investment decisions.

In the near future, we plan to explore the possibility of incorporating unstructured textual information in the model such as investor's sentiment from social media, earning reports of underlying companies, the immediate policy-related news, and research reports from market analysts. Another potential direction of the future work can be developing hybrid predictive models by combining the LSTM with some other neural networks architectures. To improve the prediction accuracy even further, we also plan to implement hybrid optimization algorithms to train the model parameters by combining the existing local optimizers with the global optimizers such as genetic algorithms and particle swarm optimization algorithm.

CRediT authorship contribution statement

Hum Nath Bhandari: Conceptualization, Methodology, Software, Supervision, Writing – original draft, Writing – review & editing. **Binod Rimal:** Conceptualization, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Nawa Raj Pokhrel:** Methodology, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing. **Ramchandra Rimal:** Methodology, Validation, Visualization, Investigation, Writing – original draft, Writing – review & editing. **Keshab R. Dahal:** Methodology, Validation, Visualization, Investigation, Writing – original draft, Writing – review & editing. **Rajendra K.C. Khatri:** Methodology, Validation, Visualization, Investigation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

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

Acknowledgments

The authors would like to thank the Association of Nepalese Mathematicians in America for creating a collaborative research opportunity that resulted in this work. We are also thankful to Google LLC for providing GPU supported open source cloud computing platform, Google Colab.

Appendix A. Single layer hyperparameter tuning results

See Tables A.10–A.14.

Appendix B. Multilayer hyperparameter tuning results

See Tables B.15–B.18.

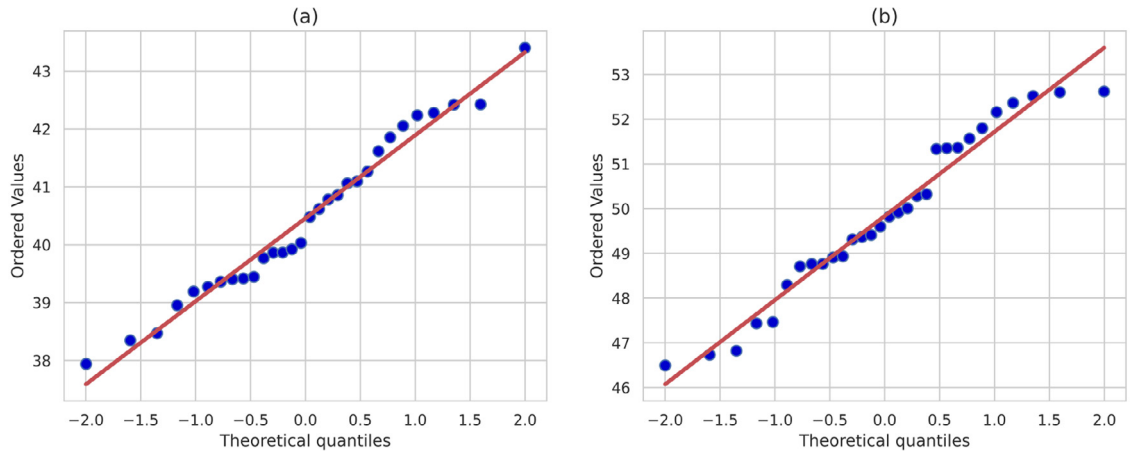

Fig. 12. QQ Plot of RMSE scores: (a) Single layer LSTM with 150 neurons, (b) Multilayer LSTM with (150–100) neurons.

Table A.11
Hyperparameter tuning for 50 neurons single layer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	60.94	53.30	50.36
	0.01	43.21	40.76	38.42
	0.001	36.80	35.05	33.62
Adagrad	0.1	32.45	31.25	30.25
	0.01	29.43	**29.00**	29.40
	0.001	31.41	34.22	38.32
Nadam	0.1	43.80	45.06	46.70
	0.01	45.71	45.76	46.19
	0.001	45.28	44.53	43.90

Table A.12
Hyperparameter tuning for 100 neurons single layer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	65.68	64.53	60.38
	0.01	51.63	46.56	43.74
	0.001	41.01	38.84	36.85
Adagrad	0.1	34.78	33.04	31.76
	0.01	30.58	29.74	**29.41**
	0.001	30.93	35.56	35.27
Nadam	0.1	45.96	52.70	58.31
	0.01	56.95	56.49	56.51
	0.001	55.22	54.04	52.91

Table A.13
Hyperparameter tuning for 150 neurons single layer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	82.81	147.61	141.31
	0.01	112.13	94.86	84.71
	0.001	76.45	69.70	64.34
Adagrad	0.1	59.48	55.55	52.28
	0.01	49.47	47.16	**45.81**
	0.001	45.88	46.57	48.13
Nadam	0.1	57.52	64.49	74.04
	0.01	71.79	70.53	70.04
	0.001	68.09	66.43	65.09

Table A.14
Hyperparameter tuning for 200 neurons single layer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	158.66	191.35	161.94
	0.01	128.91	107.99	95.54
	0.001	85.87	78.07	71.74
Adagrad	0.1	66.18	61.75	57.91
	0.01	54.67	51.95	50.14
	0.001	**50.10**	50.39	51.43
Nadam	0.1	85.14	104.74	138.73
	0.01	133.28	129.17	126.27
	0.001	122.25	118.68	115.29

Table B.15
Hyperparameter tuning for [20, 10] neurons multilayer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	86.14	66.04	56.05
	0.01	48.81	44.02	41.41
	0.001	38.51	36.22	34.91
Adagrad	0.1	33.78	33.21	32.67
	0.01	32.13	**31.82**	32.13
	0.001	33.63	35.39	38.42
Nadam	0.1	42.63	46.54	49.46
	0.01	48.24	48.11	48.67
	0.001	47.58	46.75	45.90

Table B.16
Hyperparameter tuning for [50, 20] neurons multilayer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	136.80	146.34	118.73
	0.01	95.15	80.53	54.52
	0.001	64.71	59.04	54.52
Adagrad	0.1	50.93	48.29	45.94
	0.01	43.96	42.30	**41.17**
	0.001	41.27	41.69	42.36
Nadam	0.1	48.94	54.38	59.66
	0.01	57.79	57.19	57.62
	0.001	56.05	54.75	53.66

Table B.17
Hyperparameter tuning for [100, 50] neurons multilayer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	249.48	256.85	226.71
	0.01	176.80	147.48	129.50
	0.001	115.52	104.98	96.19
Adagrad	0.1	88.44	82.26	77.33
	0.01	72.86	69.10	66.14
	0.001	65.61	**65.45**	65.76
Nadam	0.1	77.25	87.64	96.58
	0.01	93.48	92.12	91.78
	0.001	89.24	87.14	85.05

Table B.18
Hyperparameter tuning for [150, 100] neurons multilayer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	334.19	316.71	306.57
	0.01	237.61	197.21	169.94
	0.001	149.58	134.34	122.28
Adagrad	0.1	111.92	103.56	96.50
	0.01	90.48	85.47	81.45
	0.001	79.80	78.83	**78.45**
Nadam	0.1	95.79	121.36	138.42
	0.01	134.05	130.94	128.93
	0.001	124.97	121.47	118.28

Table B.19
Hyperparameter tuning for [100, 50, 20] neurons multilayer LSTM.

Optimizer	Learning rate	Batch size		
		4	8	16
Adam	0.1	255.23	249.43	234.60
	0.01	184.22	156.23	137.46
	0.001	122.72	11.26	101.80
Adagrad	0.1	93.83	87.23	81.96
	0.01	77.19	73.29	70.67
	0.001	**70.67**	71.39	73.01
Nadam	0.1	82.97	91.19	98.93
	0.01	95.85	95.08	94.81
	0.001	92.14	90.09	88.09

References

Ahangar, R. G., Yahyazadehfar, M., & Pournaghshband, H. (2010). The comparison of methods artificial neural network with linear regression using specific variables for prediction stock price in tehran stock exchange. arXiv preprint arXiv:1003.1457.

Anghel, G. D. I. (2015). Stock market efficiency and the MACD. Evidence from countries around the world. *Procedia Economics and Finance*, *32*, 1414–1431.

Baker, M., & Wurgler, J. (2007). Investor sentiment in the stock market. *Journal of Economic Perspectives*, *21*(2), 129–152.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS One*, *12*(7), 1–24.

Benhabib, J., Wang, P., & Wen, Y. (2015). Sentiments and aggregate demand fluctuations. *Econometrica*, *83*(2), 549–585.

Bernanke, B., & Kuttner, K. (2005). What explains the stock market's reaction to federal reserve policy? *The Journal of Finance*, *60*(3), 1221–1257.

Bock, J. (2018). Quantifying macroeconomic expectations in stock markets using google trends. arXiv e-prints, arXiv:1805.00268.

Chandra, A., & Thenmozhi, M. (2015). On asymmetric relationship of India volatility index (India VIX) with stock market return and risk management. *Decision*, *42*(1), 33–55.

Chaovalit, P., Gangopadhyay, A., Karabatis, G., & Chen, Z. (2011). Discrete wavelet transform-based time series analysis and mining. *ACM Computing Surveys*, *43*(2).

Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of China stock market. In *2015 IEEE international conference on big data (Big Data)* (pp. 2823–2824).

Chong, T. T.-L., & Ng, W.-K. (2008). Technical analysis and the London stock exchange: testing the MACD and RSI rules using the FT30. *Applied Economics Letters*, *15*(14), 1111–1114.

Chong, T. T.-L., Ng, W.-K., & Liew, V. K.-S. (2014). Revisiting the performance of MACD and RSI oscillators. *Journal of Risk and Financial Management*, *7*(1), 1–12.

Di Persio, L., & Honchar, O. (2016). Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International Journal of Circuits, Systems and Signal Processing*, *10*(2016), 403–413.

Eric, D., Andjelic, G., & Redzepagic, S. (2009). Application of MACD and RVI indicators as functions of investment strategy optimization on the financial market. *Zbornik Radova Ekonomskog Fakulteta U Rijeci: ČAsopis Za Ekonomsku Teoriju I Praksu*, *27*(1), 171–196.

Farsio, F., & Fazel, S. (2013). The stock market/unemployment relationship in USA, China and Japan. *International Journal of Economics and Finance*, *5*(3), 24–29.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, *270*(2), 654–669.

Gao, P., Zhang, R., & Yang, X. (2020). The application of stock index price prediction with neural network. *Mathematical and Computational Applications*, *25*(3), 53.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, *12*(10), 2451–2471.

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2003). Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, *3*(null), 115–143.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232.

Hansen, J. V., McDonald, J. B., & Nelson, R. D. (1999). Time series prediction with genetic-algorithm designed neural networks: An empirical comparison with modern statistical models. *Computational Intelligence*, *15*(3), 171–184.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107–116.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hoseinzade, E., & Haratizadeh, S. (2019). CNNpred: CNN-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, *129*, 273–285.

Kara, Y., Acar Boyacioglu, M., & Ömer Kaan Baykan (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul stock exchange. *Expert Systems with Applications*, *38*(5), 5311–5319.

Karmiani, D., Kazi, R., Nambisan, A., Shah, A., & Kamble, V. (2019). Comparison of predictive algorithms: Backpropagation, SVM, LSTM and Kalman filter for stock market. In *2019 amity international conference on artificial intelligence (AICAI)* (pp. 228–234).

Kratzert, F., Klotz, D., Brenner, C., Schulz, K., & Herrnegger, M. (2018). Rainfall–runoff modelling using long short-term memory (LSTM) networks. *Hydrology and Earth System Sciences*, *22*(11), 6005–6022.

Lanbouri, Z., & Achchab, S. (2020). Stock market prediction on high frequency data using long-short term memory. *Procedia Computer Science*, *175*, 603–608, The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC), The 15th International Conference on Future Networks and Communications (FNC), The 10th International Conference on Sustainable Energy Information Technology.

Lansing, K. J., & Tubbs, M. (2018). Using sentiment and momentum to predict stock returns. *FRBSF Economic Letter*.

Lei, J., Liu, C., & Jiang, D. (2019). Fault diagnosis of wind turbine based on long short-term memory networks. *Renewable Energy*, *133*, 422–432.

Lindemann, B., Maschler, B., Sahlab, N., & Weyrich, M. (2021). A survey on anomaly detection for technical systems using LSTM networks. *Computers in Industry*, *131*, Article 103498.

Loungani, P., Rush, M., & Tave, W. (1990). Stock market dispersion and unemployment. *Journal of Monetary Economics*, *25*(3), 367–388.

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

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. In *2017 international joint conference on neural networks (IJCNN)* (pp. 1419–1426). IEEE.

Novianti, M. (2016). *Analysis on the influence of selected macroeconomic indicators (consumer price index, trade balance, non-farm payroll, housing starts, and S&P 500) towards US index (period 2010–2015)* (Ph.D. thesis), President University.

Ortega, L., & Khashanah, K. (2014). A neuro-wavelet model for the short-term forecasting of high-frequency time series of stock returns. *Journal of Forecasting*, *33*(2), 134–146.

Otoo, M. W. (1999). Consumer sentiment and the stock market. Available at SSRN 205028.

Pan, W.-F. (2018). Does the stock market really cause unemployment? A cross-country analysis. *The North American Journal of Economics and Finance*, *44*, 34–43.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, *42*(1), 259–268.

Qiu, M., & Song, Y. (2016). Predicting the direction of stock market index movement using an optimized artificial neural network model. *PLoS One*, *11*(5), 1–11.

Qiu, J., Wang, B., & Zhou, C. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PLoS One*, *15*(1), 1–15.

Rahman, M., Usman, O. L., Muniyandi, R. C., Sahran, S., Mohamed, S., Razak, R. A., et al. (2020). A review of machine learning methods of feature selection and classification for autism spectrum disorder. *Brain Sciences*, *10*(12), 949.

Rodríguez-González, A., García-Crespo, A., Colomo-Palacios, R., Iglesias, F. G., & Gómez-Berbís, J. M. (2011). CAST: Using neural networks to improve trading systems based on technical analysis by means of the RSI financial indicator. *Expert Systems with Applications*, *38*(9), 11489–11500.

Roondiwala, M., Patel, H., & Varma, S. (2017). Predicting stock prices using LSTM. *International Journal of Science and Research (IJSR)*, *6*(4), 1754–1756.

Ruan, L. (2018). Research on sustainable development of the stock market based on VIX index. *Sustainability*, *10*(11), 4113.

Samanta, S. K., & Zadeh, A. H. (2012). *Co-movements of oil, gold, the US dollar, and stocks*. Scientific Research Publishing.

Sarno, L., & Thornton, D. L. (2003). The dynamic relationship between the federal funds rate and the treasury bill rate: An empirical investigation. *Journal of Banking & Finance*, *27*(6), 1079–1110.

Sarwar, G. (2012). Is VIX an investor fear gauge in BRIC equity markets? *Journal of Multinational Financial Management*, *22*(3), 55–65.

Stambaugh, R. F., Yu, J., & Yuan, Y. (2012). The short of it: Investor sentiment and anomalies. *Journal of Financial Economics*, *104*(2), 288–302.

Trinh, H. D., Giupponi, L., & Dini, P. (2018). Mobile traffic prediction from raw data using LSTM networks. In *2018 IEEE 29th annual international symposium on personal, indoor and mobile radio communications (PIMRC)* (pp. 1827–1832). IEEE.

Wang, J., & Kim, J. (2018). Predicting stock price trend using MACD optimized by historical volatility. *Mathematical Problems in Engineering*, *2018*.

Wilder, J. W. (1978). *New concepts in technical trading systems*. Greensboro, N.C: Trend Research.

Yadav, A., Jha, C. K., & Sharan, A. (2020). Optimizing LSTM for time series prediction in Indian stock market. *Procedia Computer Science*, *167*, 2091–2100, International Conference on Computational Intelligence and Data Science.

Yu, P., & Yan, X. (2019). Stock price prediction based on deep neural networks. *Neural Computing and Applications*, *32*, 1609–1628.