International Journal of Computational Economics and Econometrics

ISSN online: 1757-1189 - ISSN print: 1757-1170
https://www.inderscience.com/ijcee

An optimised CNN-stacked LSTM neural network model for predicting stock market time-series data

Kalva Sudhakar, Satuluri Naganjaneyulu

DOI: 10.1504/IJCEE.2025.10069875

Article History:

Received:	06 December 2023
Last revised:	25 September 2024
Accepted:	25 November 2024
Published online:	17 March 2025

An optimised CNN-stacked LSTM neural network model for predicting stock market time-series data

Kalva Sudhakar*

Department of Computer Science and Engineering,
Jawaharlal Nehru Technological University Kakinada,
Kakinada, India
and
CSE Department,
G Pulla Reddy Engineering College (Autonomous),
Kurnool, A.P, India
Email: sudhakarkalva484@gmail.com
*Corresponding author

Satuluri Naganjaneyulu

Department of Information Technology,
LakiReddy Bali Reddy College of Engineering (Autonomous),
NTR District, Mylavaram AP-521230, India
Email: naganjaneyulu.s@lbrce.ac.in

Abstract: Stock market analysis and prediction are crucial for understanding business ownership and financial performance, this study proposes an optimised CNN-stacked LSTM neural network model for accurate stock market trend prediction. The initial challenge lies in designing a customised CNN-stacked LSTM model for stock data prediction due to the abundance of non-optimised algorithms. To address this, we conducted training and testing using diverse datasets, including NYSE, NASDAQ, and NIFTY-50, observing variations in model accuracy based on the dataset. Remarkably, our model demonstrated exceptional performance with the NIFTY-50 dataset, accurately predicting up to 99% of stocks even during the testing phase. Throughout training and validation, we measured mean squared error (MSE) values ranging from 0.001 to 0.05 and 0.002 to 0.1, depending on the dataset. Our proposed CNN-stacked LSTM model presents a promising solution for accurate prediction of stock market trends, addressing the limitations of previous methods.

Keywords: stock market prediction; CNN-stacked LSTM model; time-series data; NYSE; NASDAQ; NIFTY; mean squared error; MSE; mean absolute error; MAE.

Reference to this paper should be made as follows: Sudhakar, K. and Naganjaneyulu, S. (2025) 'An optimised CNN-stacked LSTM neural network model for predicting stock market time-series data', *Int. J. Computational Economics and Econometrics*, Vol. 15, Nos. 1/2, pp.196–224.

Biographical notes: Kalva Sudhakar is currently pursuing his PhD degree as a research scholar in the Department of Computer Science and Engineering, Jawaharlal Nehru Technological University, Kakinada, Andhra Pradesh, India. He is working as an Assistant Professor in Department of Computer Science

and Engineering, at G Pulla Reddy Engineering College (Autonomous) Kurnool, India. He received a research grant from University Grants Commission of India. His research interests include big data analytics, machine learning, and data mining.

Satuluri Naganjaneyulu is working as a Professor in the Department of Information Technology at Lakireddy Bali Reddy College of Engineering, Krishna, Andhra Pradesh, India. He received his PhD in Computer Science and Engineering from Acharya Nagarjuna University, Guntur, Andhra Pradesh, India, in 2014. He received International Paper Presenter Award from CSI in the CSI2020 Annual Convention during 16 through 18 January 2020 at Kalinga Institute of Industrial Technology, Deemed to be University, Bhubaneswar, Odisha, India. He is a life member of Computer Society of India (CSI) – L1501249. His research interests include data mining, big data, machine learning, etc.

1 Introduction

The term 'stock market' refers to a monetary exchange where investors buy and sell shares of ownership in businesses. Since stock market investments can potentially provide substantial profits, studying how the market will behave is intriguing. However, due to its random walk characteristic (Yoo et al., 2022), forecasting the stock market is notoriously difficult. The stock market is a common area for corporate and business people to examine their 'shares' or other forms of ownership claims in their own companies. Since the stock market determines a nation's wealth, it is a crucial part of the economy. Stock market forecasting is challenging because market data is dynamic.

Financial markets are a captivating invention with a profound impact on various industries, including business and technology. Investors have increasingly shown interest in these markets as advancements in communication technology have facilitated participation and potential profits. The stock market provides a straightforward avenue for trading stocks and commodities, allowing traders to make substantial gains through intra-day or inter-day trading. However, it is important to recognise that the potential for high returns is accompanied by proportionate risks. While the stock market presents an enticing opportunity for low-income investors, it also exposes them to significant risk. It serves as an attractive market for both short and long-term investments, to generate profits. Nevertheless, it is crucial to acknowledge that the stock market offers substantial financial gains but also carries inherent risks.

Many people in the beginning tried to foretell stock prices using traditional methods, but they mostly failed. Therefore, the likelihood of making accurate stock market predictions is low. Machine learning (ML) methods are being used to forecast stock price movements and advise investors on how to best contribute to a company's success. However, they are false. This research examines previous efforts at stock market forecasting and introduces a novel method based on the CNN-stacked LSTM Neural Network model approach to forecasting time series data. Time series forecasting frequently makes use of the autoregressive integrated moving average (ARIMA) model, a type of statistical model. However, the linearity of the true process is assumed in ARIMA, which is not necessarily the case (Liu et al., 2019). When applied to the stock

market, artificial neural networks (ANN) and deep learning in general have been found to improve accuracy (Adebiyi et al., 2014). Time series forecasting is only one area where deep learning's many algorithms can be put to use. Long short-term memory (LSTM) networks outperform more traditional deep neural networks (DNNs) at predicting the stock market (Shah et al., 2018).

This study proposes a CNN-stacked LSTM solution to a major problem encountered when employing LSTM to analyse stock market data. In this research, we use a CNN-stacked LSTM model to forecast stock prices for client recommendations. This model is part of a larger framework that also includes preprocessing, feature selection, and a carefully crafted architecture. In order to evaluate the effectiveness of the suggested model, the loss function, mean absolute error (MAE), mean squared error (MSE) and root mean squared error (RMSE) were used to both the training and testing data. These measures give a thorough analysis of the model's performance in capturing the dynamics of stock prices. The effectiveness of the proposed CNN-stacked LSTM model is verified via a comparison to both conventional LSTM and CNN-LSTM models. By contrasting the suggested model to other methods, its superiority in terms of accuracy and performance can be assessed.

The research also explores different hyper parameters during the model design and training process. This research work introduces a novel CNN-stacked LSTM model to overcome the limitations of LSTM in stock market data analysis. By incorporating preprocessing, feature selection, and a carefully designed architecture, the proposed model aims to provide accurate stock price predictions for customer recommendations. The study rigorously evaluates the model's performance using various evaluation metrics and compares it with LSTM and CNN-LSTM models. The exploration of different hyper parameters further contributes to the optimisation of the model for effective stock price prediction

The goal of this paper is to use a deep learning approach to predict the behaviour of a stock market. Experimental results are put forward to show the benefits of our approach. The paper is structured as follows: Section 2 contains the basic preliminaries and related works; Section 3 describes the optimised CNN-Stacked LSTM model implementation on the stock market dataset. Section 4 showcases the results and discussion, and Section 5 concludes respectively.

2 Basic preliminaries and related research work

In this section, we explain the theory behind the introduced concepts that will be applied and investigated. Further, we describe how the concepts are related to the application.

A stock exchange serves as a marketplace where buyers and sellers come together to trade stocks, representing ownership in corporations. The prices of stocks are determined through the interaction of supply and demand on these exchanges. While stock trades are transactions between individuals and do not directly impact the issuing corporations, they hold significance in various ways. Major stock exchanges, such as the New York Stock Exchange (NYSE), NASDAQ, BSE, and NIFTY-50, play a crucial role in the global market. Stock market prediction is of great interest and importance due to several reasons. Firstly, it aids in investment decision-making by providing insights into price movements, trends, and market behaviour, enabling investors to optimise their strategies and maximise returns (Vasista, 2022). Additionally, it assists in risk management by

identifying potential market downturns and helping investors mitigate losses. Moreover, stock market predictions contribute to financial planning for individuals and businesses, allowing for long-term plans, investment goal-setting and informed decisions regarding financial commitments. These predictions also aid in market analysis, providing a deeper understanding of economic conditions and informing policymaking, analysis, and research. Furthermore, stock market prediction is essential in algorithmic trading, where predictive models and algorithms automate trading decisions based on real-time market data and predicted price movements. However, it's important to acknowledge the complexity and challenges involved in stock market prediction, as it is influenced by various factors and subject to inherent uncertainty. Nevertheless, ongoing research and analysis strive to improve decision-making and risk management in the financial industry.

2.1 Stock market prediction system

A stock market prediction system goes through several phases: data collection, data preprocessing, feature selection/extraction, model selection, model training, prediction and evaluation, and deployment and monitoring. Data is collected from various sources, preprocessed to remove errors and inconsistencies, and relevant features are selected or extracted. Different prediction models are evaluated and chosen, and the selected model is trained using the available data. Predictions are made and evaluated against actual market movements. The system is then deployed in a real-world setting and continuously monitored for performance and updates.

2.2 Basic preliminaries

2.2.1 Artificial neural networks

An ANN is an algorithm that will identify patterns in adapted data. The simplest ANN consists of one neuron. Each element in a vector input, $\mathbf{x}$, is assigned a weight. The summation of the elements and weights is passed to the neuron. It is common for an activation function to be applied to adjust the outcome of the summation. Provided that the threshold is achieved, a signal will be sent from the neuron. The learning in the neuron develops when the weights of each element are adapted to the threshold. Furthermore, the complexity of the ANN will become more advanced when the number of neurons is greater (Olah, 2015). ANNs have trouble managing sequential data; hence a different tool is needed, such as an RNN (Goodfellow et al., 2016).

2.2.2 Recurrent neural networks

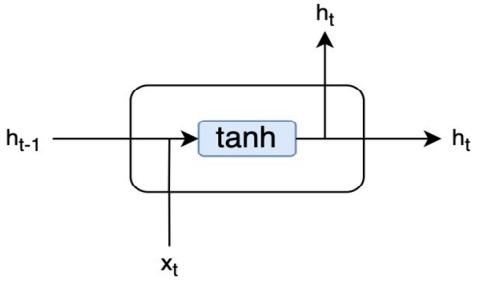
Recurrent neural network (RNN) differs from an ANN in a few ways. One major difference is that an RNN has as input at time t its output from time $t - 1$. Hence, the input to the RNN consists not only of the input data but the outcome of the last prediction. This allows the RNN to handle sequential data better when compared to an ANN.

As seen in Figure 1, the input vector at time t, $\mathbf{x}_t$, concatenates with the previous output vector, $\mathbf{h}_{t-1}$, also known as the hidden state. The concatenation, $[\mathbf{h}_{t-1}, \mathbf{x}_t]$, proceeds as the input vector for the ANN with the tanh activation function

$$\mathbf{h}_t = \tanh\left(W\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}\right), \tag{1}$$

where the nonlinear function is applied element-wise and the usage of the hyperbolic tangent function is such that the outcome vector is constrained between the values –1 and 1. The matrix **W** stores the weights and the bias vector, **b**, adds an offset independently of $\mathbf{x}_t$ to adjust the outcome. The new hidden state $\mathbf{h}_t$ is the output for the input $\mathbf{x}_t$. The hidden state, $\mathbf{h}_t$, will be passed as an input together with $\mathbf{x}_{t+1}$ for the next state.

Figure 1 RNN flow chart (see online version for colours)



Note: The tanh block performs the operations shown in equation (1).

Furthermore, the adjustable stored weights in the ANN can be adapted to learn patterns. This is done by a commonly applied technique named backpropagation (Goodfellow et al., 2016). The backpropagation algorithm calculates the gradient of the loss function, a function that measures the error; and adapts the weights to minimise the error of the outcome that the network outputs. However, when back propagating through the RNN elements of the weight matrices may become too small (Pascanu et al., 2013). This is due to the multiple matrix multiplications in the backpropagation algorithm combined with the elements within the weights matrices being less than one. The occurrence is defined as the vanishing gradient problem, leading to the performance of the RNN suffering.

2.2.3 Long short-term memory

The LSTM is proposed to solve the vanishing gradient problem (Olah, 2015). The architecture of the LSTM can be found in Figure 2. The LSTM is composed of four states: forget, store, update, and output. The forgotten state takes in an input vector $\mathbf{x}_t$, that concatenates with the previous output vector, $\mathbf{h}_{t-1}$. At the initial propagation of the network, the hidden state contains arbitrary scalars that will adapt for each iteration. The concatenated vector is passed through an ANN with a sigmoid activation function

$$\sigma = \frac{1}{1+e-x} \tag{2}$$

$$f_t = \sigma\left(W_f \otimes \left[h_{t-1}, X_t\right] = b_f\right) \tag{3}$$

The subscripts on W_f and b_f denote the forgotten state, as each state has individual weights and biases that are adapted through the data (the subscript should not be confused with the subscript of time, t). The ANN will pair the elements of the concatenated vector with adjustable weights and pass them through a sigmoid activation function. The result

of the ANN is a vector f_t that contains numbers in an interval between zero and one. Zero will indicate discard and one will indicate retaining the information. Thus, the name forget state, as it decides whether the information is remembered or forgotten. The store state will identically pass the concatenated vector through an ANN with a sigmoid activation function with weights and bias that will differ from the forget state, that is,

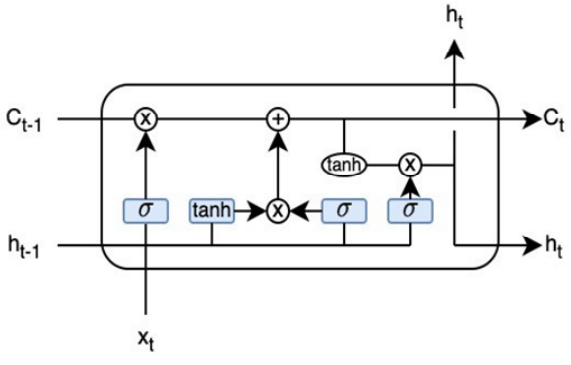
$$i_t = \sigma\left(\mathbf{W}_i \otimes \left[\mathbf{W}_{t.1}, \mathbf{x}_t\right] + \mathbf{b}_i\right) \tag{4}$$

Furthermore, the concatenated vector will also be passed through an ANN with the hyperbolic tangent activation function,

$$\tilde{\mathbf{C}}_t = \tanh\left(\mathbf{W}_C \otimes \left[\mathbf{h}_{t.1}, \mathbf{x}_t\right] + \mathbf{b}_c\right) \tag{5}$$

The final product of the state process is the vectors i_t which contain the information to be stored, and $\tilde{\mathbf{C}}_t$ which contains new possible values for the initial state. $i_t \odot \tilde{\mathbf{C}}_t$ consists of the new values scaled by the outcome of the ANN with a sigmoid activation function.

Figure 2 LSTM organisational structure (see online version for colours)



The update state consists of the cell state vector, $\mathbf{C}_{t-1}$, which at the initial propagation contains arbitrary scalars analogous to the hidden state. Moreover, the cell state is updated according to

$$C_t = f_t \otimes C_{t.1} + i_t \otimes \hat{C}_t \tag{6}$$

The first term updates the previous cell state and discards information that the algorithm wants to forget, and the second term adds new information that is scaled based on what information shall be remembered. This is due to the cell state being passed through an ANN with a sigmoid activation function, choosing what information to output, and then through a tanh activation function

$$o_t = \sigma\left(W_o \otimes \left[h_{t.1}, X_t\right] + b_o\right) \tag{7}$$

$$h_t = o_t \otimes \tanh\left(c_t\right)$$

Then the hidden state is passed as an output for the input as well as the cell state and hidden state are passed to the next input $\mathbf{x}_{t+1}$.

The LSTM reduces the vanishing gradient problem since it does not involve matrix multiplication. In place of matrix multiplication, the LSTM applies element wise multiplication about the cell state, therefore reducing the occurrence of vanishing gradient problems when back propagating (Song et al., 2022).

2.2.4 Stateful and stateless LSTM

In theory, all RNN or LSTM models are stateful, meaning they are designed to remember the entire input sequence. However, as the length of the input sequence increases, the complexity of the network also grows. To address this issue, batches are introduced, allowing the model to update its weights using backpropagation through mini sequences within each batch. It is important to note that networks do not back propagate through a set of batches (Liu et al., 2019).

In the case of a stateful LSTM, the model learns from the batch of data that is fed into the network. After backpropagation and weight updates, the network is then fed to the next batch. The weights set in each layer from the previous backpropagation serve as initial states for the subsequent batch. On the other hand, a stateless LSTM operates differently by resetting the weights to their initial states for every batch (Gers et al., 1999).

In financial models like price prediction, where the time-series data in different batches are dependent on each other, a stateful model is preferable. This allows the model to retain information from previous batches and capture the temporal dependencies within the data, improving its ability to predict prices accurately.

2.2.5 Feature learning

The intention of feature learning is data reduction and denoising. Stock data contains noise and as a result, interferes with learning (Song et al., 2022). Using feature learning we can effectively reduce the input space but also retain most of the information of the full dataset. Formally, feature learning estimates the correlation structure of the variables in a way that retains most of the information of the data. This is useful when the input dimension is high but several axes are redundant, that is, information on these axes is mostly irrelevant to the structure of the data (Wold et al., 1987).

The goal of feature selection is to reduce a set of data from n dimensions down to a linear subspace of dimension d smaller than n, where all of the data points are contained. Feature components, which are d orthogonal vectors, define the subspace. This is done like this. Take $\mathbf{X}$ where the t rows of $\mathbf{X}$ denotes each sample and the n columns denote the n features of each sample. $\mathbf{X}$ is a $t \times n$ matrix. Let us call the primary component with the biggest variance $\mathbf{U}_1$. First, we define a linear combination of $\mathbf{X}$ with coefficients (or weights) $w = [w_1 \ldots w_n]$:

$$\mathbf{U}_1 = \mathbf{w}^{\mathbf{T}}\mathbf{X} \tag{8}$$

$$\mathrm{var}(\mathbf{U}_1) = \mathrm{var}(\mathbf{w}^{\mathbf{T}}\mathbf{X}) = \mathbf{w}^{\mathbf{T}}\mathbf{S}\mathbf{w} \tag{9}$$

where $\mathbf{S}$ is the $\mathbf{X}$ covariance matrix over n independent samples. We chose to maximise $\mathbf{w}^{\mathbf{T}}\mathbf{S}\mathbf{w}$ while limiting $\mathbf{w}$ to have unit norm max $\mathbf{w}^{\mathbf{T}}\mathbf{S}\mathbf{w}$, subject to $\mathbf{w}^{\mathbf{T}}\mathbf{w} = 1$ because increasing the size of $\mathbf{w}$ allows var($\mathbf{U}_1$) to be arbitrarily big.

The weights that are initialised at the start of the LSTM are not adapted for the training data. However, for each iteration, the weights are adjusted to minimise the error. Transfer learning involves training the LSTM on correlated data before training on the original dataset. As a result, the weights will be fitted before the original dataset is applied (Kraus and Feuerriegel, 2017).

2.3 Literature review

Before the time of writing this paper, many have proposed and implemented various algorithms to predict stock market data. We did a literature survey to find some of the algorithms proposed and found some of the advantages, and disadvantages present in those algorithms. Table 1 shows the summary of the literature review.

In their study, Kompella and Chakravarthy Chilukuri (2020) analysed different machine-learning methods for stock market data prediction. They discovered that random forest outperformed linear regression and other algorithms in terms of performance. However, they observed that the error percentage increased in the model when the input data was not pre-processed and smoothed beforehand.

Pang et al. (2020) conducted an experimental analysis comparing RNN and LSTM models for stock market data prediction. They found that the LSTM model with an auto-encoder module (AELSTM) achieved better predictions compared to RNN. However, the implementation of the model was based on older libraries and when tested with real-time stock market data, the accuracy of the predictions was low. This highlighted the importance of considering the challenges and limitations when training models with real-time data.

Gurav and Kotrappa (2020) proposed a new method where LSTM with a log bilinear layer on top of it. The model predicted most of the stock market data and turned out with high accuracy but it was proposed and not tested with real-time data, also it was meant to predict data only during the time of COVID-19 and not beyond that.

Kimoto et al. (1990) presented a comprehensive discussion on a prediction system for buying and selling timing in the stock market. Their approach utilised a modular neural network that transformed technical and economic indexes into a spatial pattern, which was then fed into the neural networks for analysis. The results showed that the neural network model achieved a higher correlation coefficient compared to multiple regression. This indicates that the modular neural network approach outperformed traditional regression methods in predicting the optimal timing for buying and selling stocks. The study highlighted the effectiveness of neural networks in capturing complex patterns and relationships in stock market data, leading to improved prediction accuracy for investment decision-making.

In their study, Guresen et al. (2011) experimented to assess the effectiveness of dynamic artificial neural networks (DAN2), multi-layer perceptron (MLP), and hybrid neural networks in time series forecasting. The results indicated that the classic ANN model, MLP, consistently provided the most reliable and accurate predictions. On the other hand, the hybrid methods tested in the study did not yield improved forecast results compared to the MLP model. These findings highlight the superiority of the MLP model for time series forecasting tasks and suggest that the inclusion of additional components or techniques in hybrid neural networks may not necessarily lead to enhanced performance in this context.

Table 1 Summary of the literature review on stock market prediction using time-series data

Reference	*Method*	*Stock market/dataset used*	*Metrics used*	*Key outcomes*
Kompella and Chakravarthy Chilukuri (2020)	Random forest, linear regression analysed various ML methods for stock market data prediction	Customised dataset	Variance, MSE, MAE	Random forest outperforms linear regression, but input data smoothing is necessary for improved prediction accuracy.
Pang et al. (2020)	LSTM, ELSTM, AELSTM compared RNN and LSTM models for stock market data prediction	Shanghai A-Share composite index, Sinopec	MSE, accuracy	LSTM with auto-encoder module (AELSTM) performs well, but accuracy is affected by old libraries and real-time data.
Gurav and Kotrappa (2020)	LBL-LSTM proposed LSTM with log bilinear layer for COVID-19 prediction	Mixed	MSE, accuracy	High accuracy was achieved, but not tested with real-time data
Kimoto et al. (1990)	Developed modular neural network for buying/selling timing	TOPIX (Japan)	Correlation coefficient	The neural network model shows a higher correlation coefficient than multiple regression for predicting market timing.
Guresen et al. (2011)	ANN, DAN2, MLP, Hybrid NN Evaluated DAN2, MLP, and hybrid neural networks	NASDAQ	MSE, MAD, coefficient score	MLP model provides the most reliable results for time series forecasting, while hybrid methods do not improve results.
Nelson et al. (2017)	LSTM proposed LSTM network for short-term stock price prediction	BOVA11, BBDC4, ITUB4, CIEL3, PETR4	Accuracy, F1, precision, recall	Achieved an average of 55.9% accuracy in predicting price trends
Selvin et al. (2017)	CNN, RNN, and LSTM with the sliding-window approach compared CNN, RNN, and LSTM models with sliding window	NIFTY-50	Error percentage	CNN outperformed RNN and LSTM by considering the current window
Hiransha et al. (2018)	ANN, MLP, LSTM, RNN compared ANN, MLP, LSTM, and RNN models for stock prediction	NIFTY-50	MAPE	CNN showed better performance overall
Bansal et al. (2019)	LSTM with smart contracts (DAG-based) Proposed intelligent decentralised stock market model	NYSE	MSE	LSTM-based ML model achieves 99.71% accuracy in stock prediction using specific feature vectors and training settings.

To forecast stock price movements at 15-minute intervals using a combination of price history and technical analysis indicators, Nelson et al. (2017) presented an LSTM network. Predictions about whether a stock's price will rise in the near future were correct 55.9% of the time on average.

Selvin et al. (2017) tried out a sliding window technique with three distinct deep learning models (CNN, RNN, and LSTM). CNN's superior performance over the other two models can be attributed to its exclusive focus on the most recent data when making stock-price forecasts. This paves the way for CNN to comprehend the evolving patterns and shifts in the current frame. However, RNN and LSTM forecast future instances based on data from earlier lags.

Experiments comparing ANN, MLP, LSTM, and RNN were undertaken in Hiransha et al. (2018). Both ANN and RNN were able to recognise the pattern early on, but once the pattern had been there for a while, neither could. Similarly, LSTM demonstrated reduced accuracy for the projected values during some time periods, but CNN still tended to outperform the other three networks.

Intelligent decentralised stock market models based on ML and DAG-based crypto currency were introduced in Bansal et al. (2019). The proposed model, which made use of LSTM (a RNN), managed a very respectable 99.71% accuracy in its predictions. Each stock's feature vector had a feature vector with four parameters: open, close, low, and high. The model was trained for a total of 100 iterations with a batch size of 50 for optimal results. This research shows that LSTM, a ML technique, can be used to forecast future stock market patterns in a distributed, intelligent system.

These studies explore various aspects of stock market prediction, including the performance of different ML models, the importance of data preprocessing, the effectiveness of neural network models, and the potential of intelligent decentralised approaches.

2.4 Motivation and research issues identified

The implementation of an optimised CNN-Stacked LSTM neural network model for stock market prediction is driven by the significance of accurate predictions in the financial industry. The volatile and complex nature of stock market data, the need to incorporate relevant features, limitations of existing models, and the potential of the CNN-Stacked LSTM architecture all contribute to the motivation. Accurate predictions are crucial for investors and traders to make informed decisions, and the optimised model aims to overcome the challenges and improve prediction accuracy in this domain.

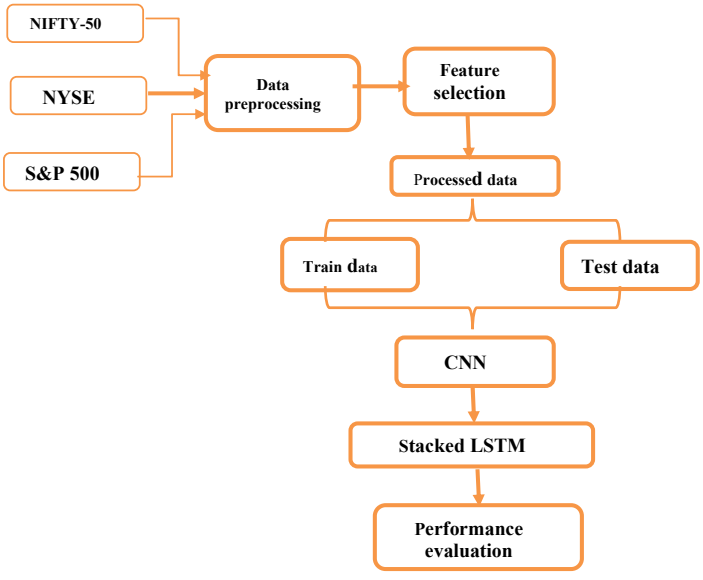
By addressing these research issues and implementing an optimised CNN-Stacked LSTM neural network model, this research aims to provide a more effective and accurate tool for predicting stock market time-series data, enabling investors and traders to make informed decisions and improve their financial outcomes.

3 An optimised CNN-stacked LSTM model for stock market predictions

Considering the research gaps, we focused on making the deep learning model for time series data. We decided to go on with the CNN-Stacked LSTM Neural Network approach because CNN helps in tracking the features of the dataset and LSTM helps in tracking the

patterns, allowing us to train on them. This approach is not the first time as some researchers already tried to implement the CNN-stacked LSTM method but we tweaked the parameters, kernel sizes (for CNN), and layers to experiment and test it on real-time data. Since this is a regression type of problem where we had to train with time-series data, we used MSE as the standard metric rather than accuracy. The architecture diagram for the neural network is shown in Figure 3.

Figure 3 Proposed methodology for optimised CNN-stacked LSTM neural network model for predicting stock market time-series data (see online version for colours)



The objective for using CNN-stacked LSTM in stock market prediction stems from the desire to leverage the unique strengths of these models. CNNs excel at capturing spatial patterns, while LSTMs are effective at modelling temporal dependencies. By combining these architectures, researchers aim to capture complex and multi-dimensional patterns in the stock market data, extract meaningful features from sequential data, model long-term dependencies, handle multivariate inputs, capture nonlinear relationships, and draw upon the success of these models in other domains. The ultimate goal is to develop robust prediction models that provide accurate insights into stock market trends, enabling investors and financial institutions to make informed decisions and improve risk management.

3.1 Dataset description

Data opening, closing, high, low, and volume are all examples of numerical historical data employed in these models. The information also includes technical indicators based on past performance. The rolling window approach is utilised for time series forecasting. A series of matrices, where each row represents a day and each column a feature, is created by rolling window.

The description of each dataset is

1 *NIFTY-50*: The NIFTY-50 is a stock market index of the National Stock Exchange (NSE) in India. It consists of the 50 largest and most actively traded stocks across various sectors of the Indian economy. Similar to the SENSEX dataset, the NIFTY-50 dataset may include historical price data, trading volumes, market capitalisation, sector information, and other relevant variables (Kaggle, 2017).

2 *NASDAQ*: The NASDAQ is a global electronic marketplace for buying and selling securities, with a focus on technology stocks. The NASDAQ Composite Index represents the performance of over 3,000 stocks listed on the NASDAQ stock exchange. Datasets related to NASDAQ may include historical price data, trading volumes, company information, sector classification, and other variables relevant to the listed stocks (Kaggle, 2020).

3 *NYSE*: The NYSE is the world's largest stock exchange by market capitalisation. It lists a wide range of stocks from various industries. Datasets related to the NYSE may include historical price data, trading volumes, company information, sector classification, market indices, and other variables associated with the stocks listed on the NYSE (Kaggle, 2022).

4 *S&P 500*: The S&P 500 stock dataset contains historical financial data for companies listed on the S&P 500 index. It includes information such as opening and closing prices, trading volume, and adjusted closing prices. The dataset is used for financial analysis, risk assessment, and stock market prediction. It is a valuable resource for investors and researchers studying the performance of S&P 500 companies (Kaggle, 2022).

3.2 Exploratory data analysis

Before making a model, the first step is to collect enough datasets such that the base analysis is made to study the stock market data. So, we gathered enough datasets from Kaggle (explained in later stages) but realised that they are sample ones and we had to search for real-time ones. Then, we came across several finance APIs like Yahoo Finance, and Alpha Vantage which help in gathering stock data for a specific period. So, we took Alpha Vantage API and used the 'TIME SERIES DAILY' option to obtain stock data of a company ranging from ten years. We used 'full' mode to collect enough data rather than using 'compact' mode in API (which fetches only 100 columns meant for rapid usage cases) and we were able to collect the data for any company with valid API keys. Some stock data is also gathered using Google Sheets via the 'GOOGLEFINANCE' function. Then we stored the data in CSV format for the testing phase. Then, we did an exploratory data analysis (EDA) on the dataset to know about the stock market data in depth. We also implemented Moving Average and Daily Return columns to know how a stock market works and analysed some of the features present in it. After that, we went to preprocessing phase.

In the preprocessing phase, we first cleansed the data by removing NULL values from the dataset and taking the mean of data, and replacing it if necessary using the Pandas library. Then we took the four columns of any stock market dataset, namely 'open', 'close', 'high', and 'low'. These are the columns which mainly involve in training the dataset especially the 'close' column (shown in Table 2). The graphs are plotted using the matplotlib and seaborn libraries in Python.

3.3 Data normalisation and feature data construction

Label transformation, duplicate elimination, and data normalisation are the three phases of data pre-processing for the proposed stock market prediction model. The initial step is to translate the symbolic class input columns into numerical labels depending on the prediction type. To minimise biased categorisation toward frequent data records, the second step is to remove duplicate information. The most essential stage is the third, which is primarily beneficial in stabilising the dataset by eliminating the biased features of greater values. It entails converting each element's values into a proportionate range. The range [0, 1] is specified in the proposed method, and the elements are standardised toward it utilising a generalised normalisation condition.

$$Y_{normalised} = \frac{Y - Y_{\min}}{Y_{\max} - Y_{\min}} \tag{10}$$

where $Y_{\min}$ and $Y_{\max}$ are the data feature's minimum and maximum values, respectively, and Y is the data feature's current value. The data characteristics are normalised to allow for linear data processing.

During the preprocessing stage, we realised that CNN always considers two-Dimensional and three-dimensional arrays to train and select required features. But here, the data we have is of one-dimensional arrays. This is one of the reasons why CNN is often seen in computer vision (CV)-based applications and not in NLP-based applications. So, for the CNN model to parse the dataset, we made a function where the 1D arrays are made to convert to 2D arrays. Table 2 shows the instance of the sample stock market dataset.

[100, 1] tensors (precisely, a vector). Tensors are a type of data structure that describes a multilinear relationship between a set of objects in a vector space. So, for converting a 1-D array to a tensor, every 100 rows are taken, and from that the mean of the values are calculated and made to store in a separate column. This process is done for the entire dataset. In our case, we did this on the 'close' column as it's the main column where we would decide the prediction of the stock data.

Table 2 Instance of stock market dataset

S. no.	*Date*	*Symbol*	*Open*	*High*	*Low*	*Close*	*VWAP*	*Volume*
1	2004-08-25	TCS	1,198.7	1,198.7	979	987.95	1008.32	17,116,372
2	2004-08-26	TCS	992	997	975.3	979	985.65	5,055,400
3	2004-08-27	TCS	982.4	982.4	958.55	962.65	969.94	3,830,750
4	2004-08-30	TCS	969.9	990	965	986.75	982.65	3,058,151
5	2004-08-31	TCS	986.5	990	976	988.1	982.18	2,649,332
6	2004-09-01	TCS	990	995	983.6	987.9	989.68	2,491,943

After this step, we would obtain tensors for the CNN side of the model to train. Then, we split 80% for training and 20% for testing. Finally, we reshaped the data and sent it to the training phase.

3.4 CNN-stacked LSTM for stock price prediction

The CNN is generally utilised in tasks based on recognition of objects and CNN can overcome the problems related to processing the objects. The performance of CNN architecture is analysed to predict the best stock according to the user's previous portfolio. The stacked LSTM is used as a numerical function to aid in the provision of the stock market. A sequence processing model called a stacked LSTM, also referred to as a stacked LSTM, is made up of two LSTMs, first LSTM is trained on the data, and the second is trained on the outcome of the first LSTM to access additional data, which improves the context of the algorithm. The CNN is composed of the convolutional layer which acts as a platform to project the CNN where the attributes of products are stored. In this paper, the stacked LSTM is combined with the CNN architecture to provide an effective and efficient prediction of time series data prediction on stocks. The primary objective of the proposed stacked-LSTM architecture is to develop an efficient system that can be accessible with less data. The study utilises standard sequential CNN architecture to predict the best stock according to the company and user portfolio.

3.4.1 CNN for extraction of feature attributes

CNN is utilised to extract the top-notch features of the product attribute from input data. Based on the performance of the model and to reduce computational complexity, the sequential CNN architecture is adopted on the different stock market datasets. For an input attribute that has an area of 64 × 64 in channels based on attributes. The output is obtained as feature maps based on the CNN (trained) and it is denoted as $F_{t_i} \in R^{D\times7\times7}$ and $F_{b_j} \in R^{D\times7\times7}$ in that the dimensional size of the output is denoted as D and the size of the feature map is denoted as 7 × 7. The visuals based on feature maps are compressed by F_{t_i}) and F_{b_j} to gather dimensional vectors. The vectors based on the dimensions are denoted as $v_{t_i} = \{V_1^{t_i}, V_2^{t_i}, ..., V_{49}^{t_i}\}$ and $v_{b_j} = \{V_1^{t_i}, V_2^{t_i}, ..., V_{49}^{t_i}\}$ where the number of features represent in to feature map is denoted as $V \in R^D$. The vectors t_i and b_j is obtained by pooling the respective v_{t_i} and v_{b_j} vectors in a pooling layer, mathematically represented by equation (1). This approach allows for the extraction and representation of relevant features from the input data for stock market prediction tasks.

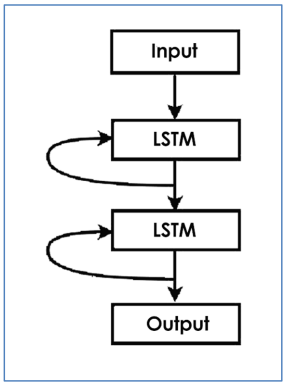
$$v_{t_i} = \frac{1}{49}\sum_{n=1}^{49} V_n^{t_i}, v_{b_j} = \frac{1}{49}\sum_{n=1}^{49} V_n^{b_j} \tag{11}$$

where the features utilised for embedding t_i and b_j is denoted as $v_{t_i}, v_{b_j} \in R^D$.

3.4.2 Stacked LSTM architecture

The initial model for forecasting stock prices, which all the proposed models are based upon, consists of 2 LSTM architectures, thus the name stacked LSTM. The architecture of a stacked LSTM is constructed by the first LSTM is trained on the data, and the second is trained on the outcome of the first LSTM. The result of a stacked LSTM is the ability to detect complex features in the data and improve the performance of forecasting (Song et al., 2022). Figure 4 represents the stacked LSTM model organisation.

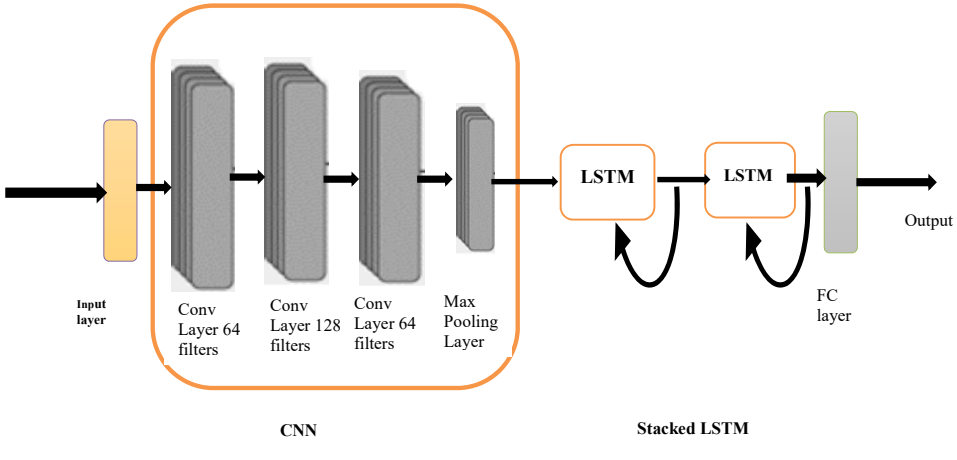
Figure 4 Stacked LSTM model organisation (see online version for colours)



3.4.3 Initial CNN-stacked LSTM model

Before the model is initialised the data is prepared. The data is partitioned into training, validation, and testing. The data does not only consist of the sequences of stock prices but also a vector based on the closing prices, and labels, as the model will adjust its weights according to these values. As the model is a stacked LSTM, two sets of models which are based on the equations of the LSTM background will be constructed. More specifications on how and which hyper parameters were chosen for the equations can be found in Section 4. Note the initial value of the h_0 and c_0 will consist of random elements, as these vectors are needed to be passed into the model at the initial time. To obtain an optimal model it is relevant to address the problem of overfitting, which has an impact on the stacked LSTM. Figure 5 represents the proposed CNN-stacked LSTM model for stock market time series data prediction.

Figure 5 Proposed CNN-stacked LSTM model for stock market time series data prediction (see online version for colours)



3.4.4 Training phase

After the dataset is processed, the NN model has to be made. In our case, it is the CNN-stacked LSTM Neural Network model. For our model, we considered dividing the model into two parts, CNN and stacked LSTM.

- *CNN*: For the CNN section of the model, we followed a custom way instead of ascending kind of way in the size of layers. So, we made three layers of neuron size 6,412,864 with kernel size = 3 along with max pooling layers in between. Finally, we added a flatten layer at the end of the CNN section to convert the tensors back to a 1D array. All CNN layers are added with the time distributed function to train every temporal slice of input, as we're approaching a time-series problem in this case. Then, the processed data is sent to LSTM layers.
- *LSTM*: For the LSTM, we made two bidirectional LSTM layers to detect the features and train them forward and backward. For each layer, the neuron size is 100. Additionally, dropout layers are added in between with a value of 0.5 in drop some features for stability. Last, we added a dense layer with a linear activation function, and 'Adam' optimiser, MSE as the loss function, and 'MSE' and 'MAE' as metrics.

Overfitting poses a common challenge in ML, leading to decreased performance on test datasets. It arises when the algorithm overly fits the training data, usually due to model complexity, mismatched dataset representation, or excessive noise. To mitigate overfitting, dropout is employed as a preventive measure. Dropout temporarily deactivates a random set of neurons, allowing the remaining neurons to continue training. This prevents excessive adaptation to the training set (Srivastava et al., 2014). In the proposed model, dropout is applied to each model to address overfitting.

The model is learning during the training phase, the data from the sequence and labels are passed to the model in cycles called epochs. For each prediction, the value is applied to the loss function, which for this model is the mean square error, together with the matching label value. The value of the loss function is then applied to the back propagating algorithm Adam which is applied according to Kingma and Ba (2014). The Adam optimiser will adjust the weights of the stacked LSTMs to minimise the loss, thus for each epoch, the model will adjust its weights concerning the data.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(p_i - t_i\right)^2 \tag{12}$$

To find the most optimal model and also as a way to avoid overfitting, a validation dataset is constructed from the total dataset. Each time the model is adjusting the weights, it is tested on the validation dataset. Note that if the validation loss increases it implies the model is overfitting. Therefore, to obtain the most optimally performing model, the one model with the lowest validation error will be saved. When the training phase is completed, the saved model will be loaded and set to the evaluation model, turning dropout off, this also occurs in the validation dataset. Thereafter the testing dataset is applied, which has been unstandardised, and evaluated according to a given set of evaluation metrics.

3.4.5 Testing phase

After the model has been trained and the values are noted, we saved the model in HDF5 format using Keras API in the TensorFlow library. Then, we loaded the HDF5 file and tried training the model again but this time with a different dataset, we were able to train the model but the loss value varies accordingly. For example, if the loss is 0.055 during the training phase, the loss increases to 0.153 (estimated, not accurate). It is also found that this happens depending on the dataset we use, for the NIFTY sample dataset, the error did not occur whereas, in the NASDAQ dataset, it occurred while loading up the saved model.

3.5 Performance measures

For the evaluation of the model the common metrics are used: mean average percentage error (MAPE), MAE, RMSE, and the correlation coefficient (Powers and Ailab, 2011).

3.5.1 Mean average percentage error

MAPE describes the error in percentage in consideration of the true value, t_i, and the predicted value, p_i

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{p_i - t_i}{t_i}\right| \tag{13}$$

3.5.2 Mean absolute error

MAE represents the average absolute error

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|p_i - t_i\right| \tag{14}$$

3.5.3 Root mean squared error

RMSE is a measurement of the average Euclidean distance between t_i and p_i

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(p_i - t_i\right)^2} \tag{15}$$

3.5.4 The correlation coefficient (R)

The correlation coefficient (R) measures the correlation between the prediction and the true value. A greater value on R implies a greater correlation between the variables t and p.

$$R = \frac{\sum_{i=1}^{N}(t_i - t_{mean})(p_i - p_{mean})}{\sqrt{\sum_{i=1}^{N}(t_i - t_{mean})^2}\sqrt{\sum_{i=1}^{N}(p_i - p_{mean})^2}} \tag{16}$$

3.5.5 Regularisation

- *L1 regularisation*, also known as the goal of the ML technique known as Lasso regularisation is to include a penalty term whose magnitude is directly related to the absolute values of the model's coefficients (Tibshirani, 1996).
- *L2 regularisation* known as ridge regularisation is applied to models to combat overfitting. Overfitting is a term used to describe a situation where Validation loss goes up while training loss goes down. In other words, the model is well fitted on training data but it is not predicting accurately for validation data. The model is not able to generalise (Tikhonov, 1943).

$$minimise\left(Loss(Data \mid Model) + complexity(model)\right) \tag{17}$$

The complexity of the models used in the paper was minimised by using L2 regularisation. The formula of L2 regularisation is the sum of the square of all the weights,

$$L_2 \text{ regularisation term} = \| w \|^2 = w_1^2 + w_2^2 + ... + w_n^2 \tag{18}$$

In the models, two layers of L2 regularisation were used before the final output layer.

3.5.6 Evaluation of loss function

In the context of stock market prediction, the evaluation of the loss function is crucial. The dataset primarily consists of positive ratings representing successful stock predictions, while negative ratings indicating unsuccessful predictions are absent. To address this, a ranking loss function is employed to capture the relationship between the actual price and the predicted price. This loss function generates pairs of positive and negative for each stock, incorporating corrupted pairs (t_i, $b_{j'}$) and ($t_{i'}$, b_j) where positive and negative are exchanged. By considering these pairs, the ranking loss function facilitates a comprehensive assessment of stock market predictions. So, the noticed pair must be given priority at on higher rate than the unnoticed one, it is represented in equation (14) shown below,

$$L_{rl} = -\sum_{(i,j,j') \in D} \ln\left(\sigma\left(\hat{y}_{ij}^{compat} - \hat{y}_{ij'}^{compat}\right)\right) \tag{19}$$

where all training samples are represented as D and the sigmoid function is denoted as σ.

Finally, by using equations (2), (4), and (14), the objective function of the Attribute specific recommendation system for health products is formulated using equation (15).

$$L = L_{category} + L_{attribute} + L_{rl} \tag{20}$$

Assessing the model's prognostic accuracy during times of extreme index volatility, such as large out-of-the-ordinary swings in either direction, is also part of the evaluation process.

3.6 CNN-stacked LSTM for stock market price prediction

Algorithm: CNN-Stacked LSTM for stock market price prediction

Input: NIFTY-50, NYSE, and S&P 500 stock data

Output: Predictions

Step 1 Import the necessary libraries.

Step 2 Prepare the input data:

X_train: Training input features (time series data)

y_train: Training target labels (stock prices)

Partition of the Data in training, and testing

Step 3 Define the model architecture:

Create a Sequential model.

Add a Conv1D layer with filters, kernel size, and activation function.

Add Bidirectional LSTM layers with units and return_sequences set to True.

Add any additional LSTM layers as needed.

Add a Dense output layer.

model.add(Dense(units=1))

Step 4 Compile the model:

Choose an optimiser (e.g., 'adam') and a suitable loss function (e.g., 'mse').

Step 5 Train the model:

Fit the model to the training data (X_{train}, y_{train}) with the desired number of epochs and batch size.

for epoch in epochs do

Model set to training mode, activating dropout for each data in training data do

Step 6 Make predictions:

Use the trained model to predict stock prices on the test data.

Predict using training data and LSTM

Step 7 Evaluate the model (optional): Calculate evaluation metrics such as mean squared error (MSE) using the predictions and actual values.

Step 8 Perform any further analysis or visualisation based on the predictions and evaluation metrics.

4 Results and discussion

In this section, we describe the hyper parameters that were used for the models, the specifications on the computer that the model was trained on, and how the data was collected. Further, the details of how the data was partitioned are explained and what measurements are applied for evaluation. At last, the experimental results of the models are shown. We tested and experimented with the model with different datasets from Kaggle consisting of sample data of mixed content from different stock markets (Kaggle,

2017), NIFTY-50 (Kaggle, 2020), NASDAQ, and NYSE (Kaggle, 2022) to find how the model copes with the different stock market. The proposed research work is implemented using the libraries Pytorch and Sklearn.

4.1 Experimental setup

The number of tuneable parameters, hyper parameters, of the model is 5 and is set to, batch size: 1, hidden size: 128, number of stacked LSTMs: 2, dropout: 0.3, and learning rate: 0.001. A changeable parameter that does not involve the model is the size of the window that is used for the rolling window method, which according to Nti et al. (2021) changes the performance. In this research work, the window size is set to 20. Table 3 shows the model parameters used in implementing this research work.

Table 3 Model parameter

Hyper parameters	*Value*
Batch size	40
Hidden size	128
LSTM modules (integrated into the stacked environment)	2
Dropout	0.5
Kernel size	3
Learning rate	0.001
Activation function	ReLu
Optimiser	Adam
Window size	20

The training and validation data is shuffled, meaning each time the data is propagating through the model it is reshuffled. If the validation error is lower than the epoch before, the model is then saved. To minimise the training for the model, the algorithm will monitor if the validation error has not decreased in 50 epochs the model will break and continue to the test data. The number of components was chosen based on the inbuilt function in sklearn for feature selection.

4.2 Results on NIFTY-50 (NSE), NASDAQ

MSE and MAE were determined during training to assess the CNN-Stacked LSTM model's performance on the NIFTY-50 (NSE) and NASDAQ datasets. When comparing projected and observed values, the MSE measures the average squared discrepancy, while the MAE shows the average absolute discrepancy. These metrics indicate the level of accuracy and precision achieved by the model during training on the NIFTY-50 (NSE) and NASDAQ datasets. By monitoring and analysing the MSE and MAE values obtained during training, it is possible to assess the model's convergence, identify areas for improvement, and gauge its suitability for predicting stock market trends and making informed investment decisions. Figures 6 and 7 represent MSE obtained during training and MAE obtained during training for the NIFTY-50 dataset.

Figure 6 MSE performance during training (see online version for colours)

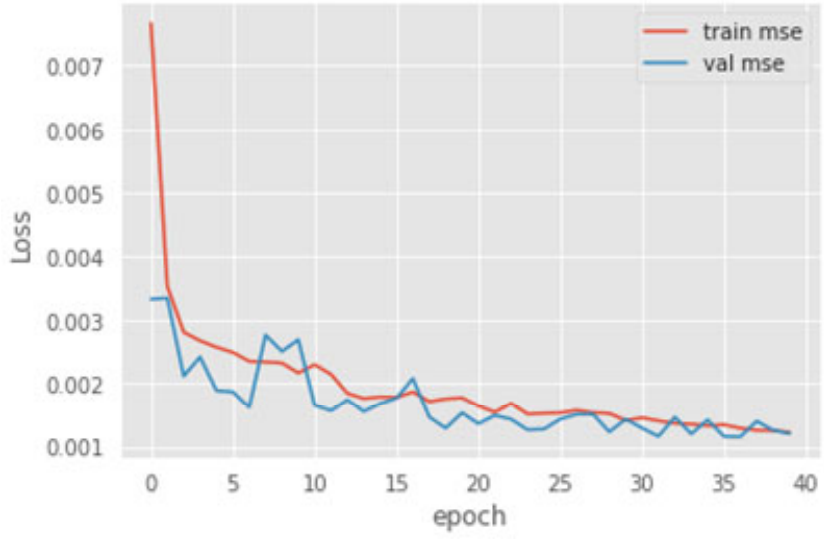
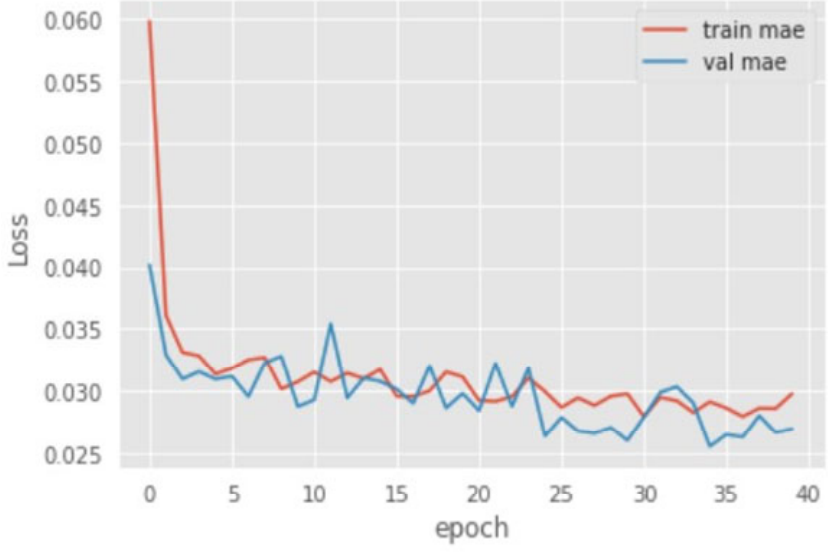


Figure 7 MAE performance during training (see online version for colours)



The CNN-stacked LSTM model has been applied to both the NIFTY-50 (NSE) dataset and the NASDAQ dataset. The prediction graphs were generated to visualise the model's performance. For the NIFTY-50 (NSE) dataset, the prediction graph was based on shuffled sample data, while for the real stock data of NIFTY-50, the prediction graph was generated without shuffling the data. These graphs provide a visual representation of how well the model predicts stock market trends and fluctuations in both the sample and real datasets. Analysing these prediction graphs can provide insights into the model's ability to capture and forecast the patterns and movements in the NIFTY-50 (NSE) and

NASDAQ datasets, assisting in financial forecasting and investment decision-making. Figure 8 the prediction graph for sample NSE data (shuffled) and the prediction graph for real stock data NIFTY-50 (un-shuffled) is shown in Figure 9.

Figure 8 The prediction graph for training data (shuffled) (see online version for colours)

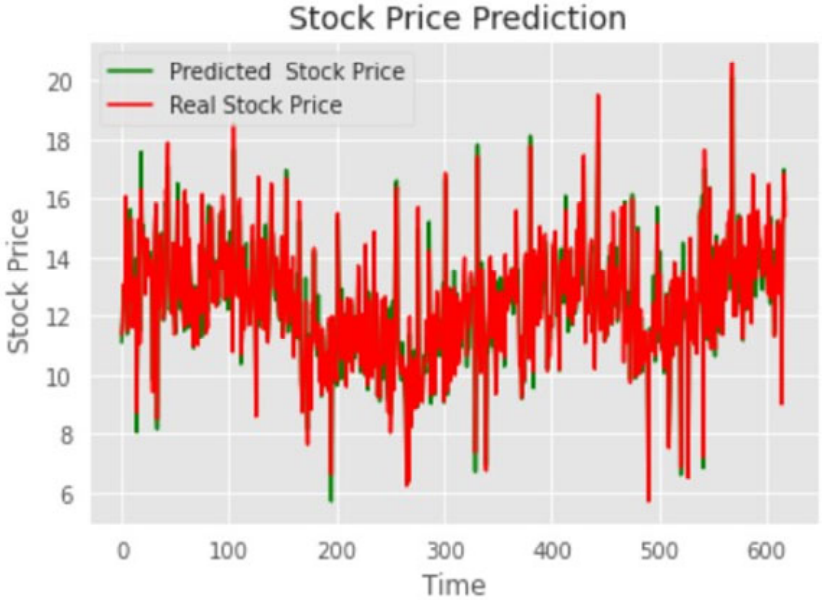


Figure 9 The prediction graph for NIFTY-50 (un-shuffled) the prediction graph for testing data (see online version for colours)

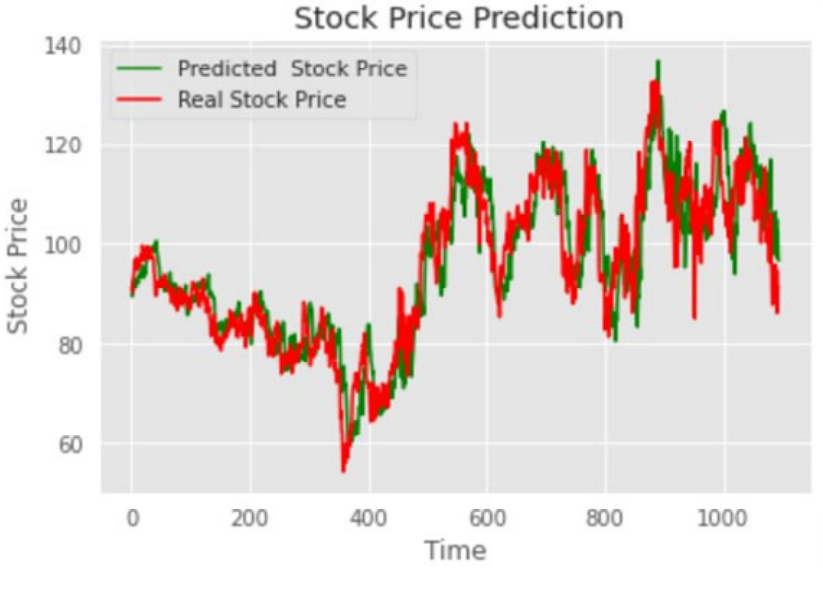


Table 4 presents accuracy scores of the CNN-Stacked LSTM model on the NIFTY-50 dataset. Metrics such as MSE, MAE, variance, and R^2 score assess the model's performance in predicting stock prices. The low MSE value of 0.030 indicates accurate predictions and successful capture of patterns and trends in the dataset. The model's ability to recommend stocks demonstrates its practical value for investment decisions. By utilising the model's accurate predictions, investors can make informed choices and potentially enhance their investment performance. Table 5 represents the model performance on different stock markets (Kaggle, 2017), NIFTY-50 (Kaggle, 2020), NASDAQ, and NYSE (Kaggle, 2022) to find how the model copes with the different stock markets. The model tested with stock data both shuffled and un-shuffled. It was able to predict most of the stocks as shown in Table 5.

Table 4 Various performance scores on NIFTY-50 dataset

Name	*Score*
Loss	0.0012
Train MSE	0.0012
Train MAE	0.0265
Test MSE	0.0148
Test MAE	0.0814
Variance	0.938731
R^2 score	0.938751
Max error	0.250160

Table 5 MSE score with datasets NIFTY-50 (Kaggle, 2020), NASDAQ, and NYSE (Kaggle, 2022)

Dataset and MSE scores	
Dataset	*MSE score*
NIFTY (SBIN – sample)	0.001
NASDAQ (ACTG – sample)	0.1565
NASDAQ (AAOI – sample)	0.0016
NYSE (IBM – real)	0.0027
BSE (RELIANCE – real)	0.0145

The MSE performance of the LSTM, CNN-LSTM, and CNN-stacked LSTM models was compared to evaluate their effectiveness in predicting stock prices or other time series data. The CNN-Stacked LSTM model achieved the lowest MSE of 0.030, followed by the CNN-LSTM model with an MSE of 0.035, and the LSTM model with an MSE of 0.045. These MSE values indicate the average squared difference between the predicted and actual values, serving as a measure of accuracy for the models' predictions. The lower MSE values for the CNN-Stacked LSTM and CNN-LSTM models suggest their superior performance in capturing patterns and trends in the data, leading to more accurate predictions compared to the LSTM model. Table 6 provides a visual representation of the MSE comparison among the models.

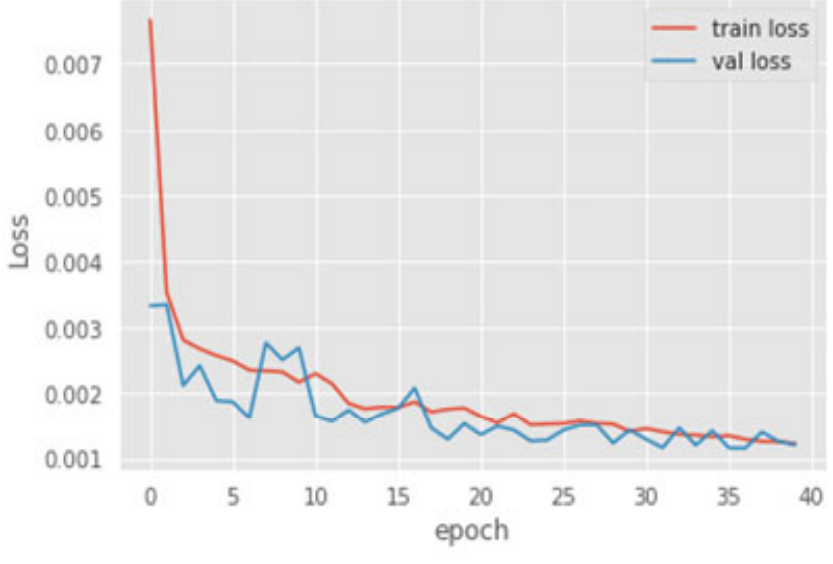
Table 6 MSE Scores for LSTM, CNN-LSTM, and CNN-stacked LSTM models

Model	*MSE score (Avg)*
LSTM	0.045
CNN-LSTM	0.035
CNN-Stacked LSTM	0.0014

4.3 CNN-stacked LSTM model performance loss on NIFTY-50 dataset

The performance loss function of the CNN-Stacked LSTM model on the Nifty-50 dataset plays a crucial role in evaluating the model's predictive accuracy. The choice of an appropriate loss function is essential for guiding the training process and optimising the model's parameters. Commonly used loss functions for regression tasks on financial datasets, such as the Nifty-50 dataset, include MSE and MAE. On the other hand, MAE computes the average absolute difference, which is useful for assessing the model's ability to capture the magnitude of the predicted values accurately. By minimising the loss function during training, the CNN-Stacked LSTM model aims to improve its predictive capabilities and achieve higher accuracy in forecasting the Nifty-50 dataset. Figure 10 represents the performance loss of the CNN-stacked LSTM model on the NIFTY-50 dataset.

Figure 10 The performance of the CNN-stacked LSTM model on the NIFTY-50 dataset (see online version for colours)



4.4 Performance of CNN-Stacked LSTM model on NIFTY-50

The performance of the CNN-Stacked LSTM model on the Nifty-50 dataset was assessed using metrics such as MSE, MAE, variance, and R^2 score. These metrics provide a comprehensive evaluation of the model's predictive accuracy, precision, and overall

performance on the Nifty-50 dataset. Figure 11 visually represents the model's performance, showing that it predicts the stock prices closely to the actual prices. By analysing the MSE, MAE, variance, and R^2 score, it is possible to determine the model's effectiveness in capturing and forecasting patterns and trends in the NIFTY-50 dataset.

Figure 11 Generalised performance of CNN-stacked LSTM model on NIFTY-50 dataset (see online version for colours)

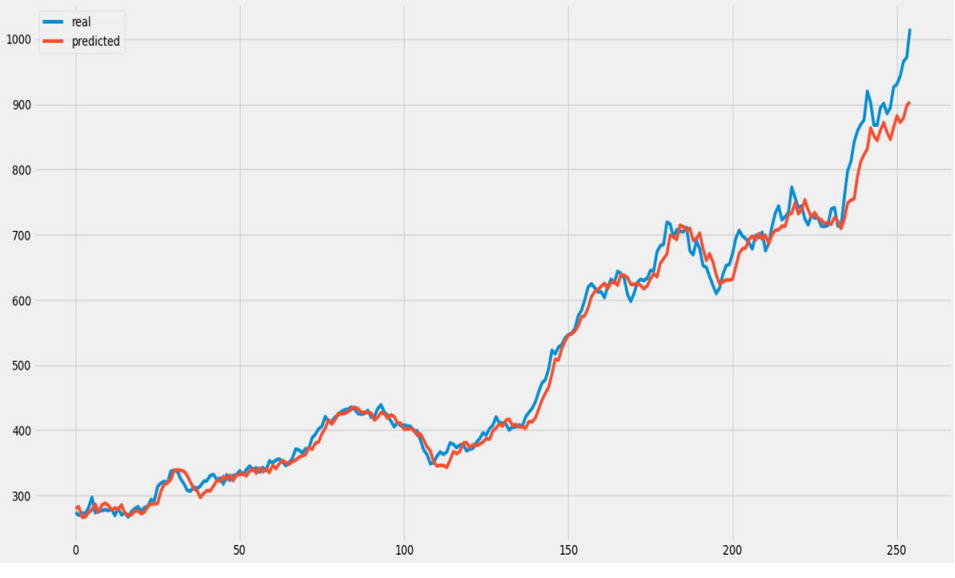
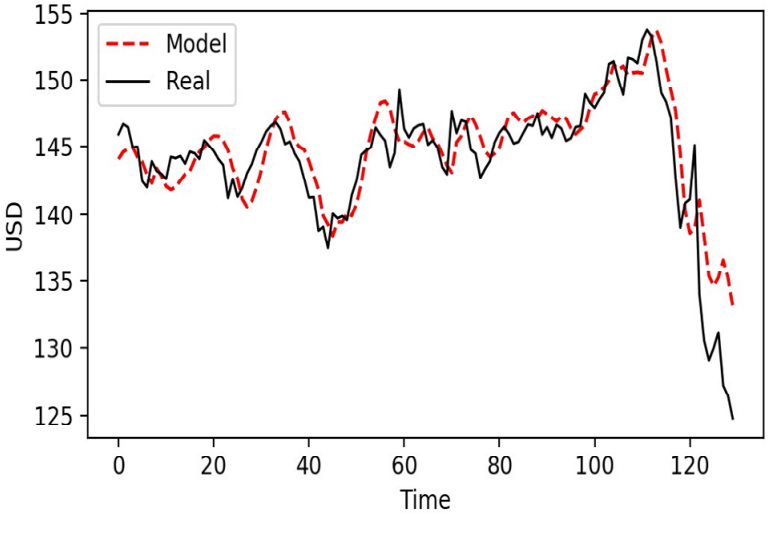


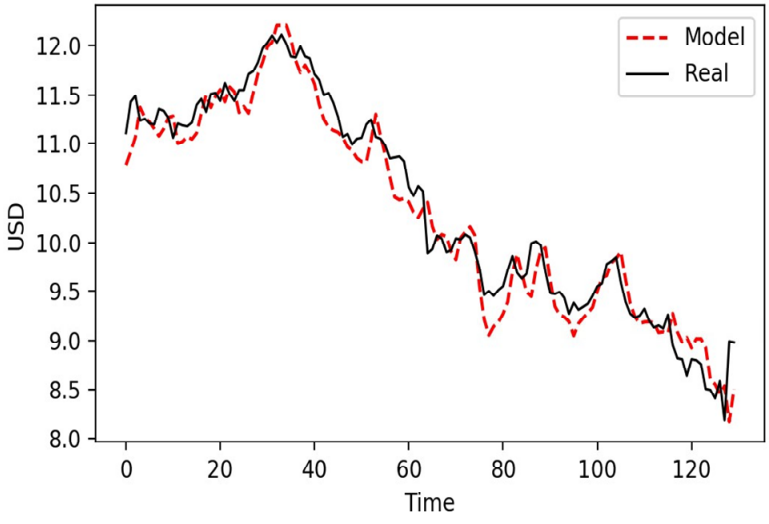
Figure 12 Models prediction NYSE stock (see online version for colours)



4.5 Performance of CNN-stacked LSTM model on the NYSE dataset

The CNN-stacked LSTM model was trained on the NYSE dataset, and the MSE and MAE were used to evaluate its performance. The MSE measures the average squared difference between predicted and actual values, while the MAE represents the average absolute difference. These metrics assess the model's accuracy and precision during training on the NYSE dataset. Monitoring the MSE and MAE values helps assess convergence, identify areas for improvement, and evaluate the model's suitability for predicting stock market trends and making informed investment decisions. Figures 12 and 13 display the stock prediction rate of the NYSE dataset.

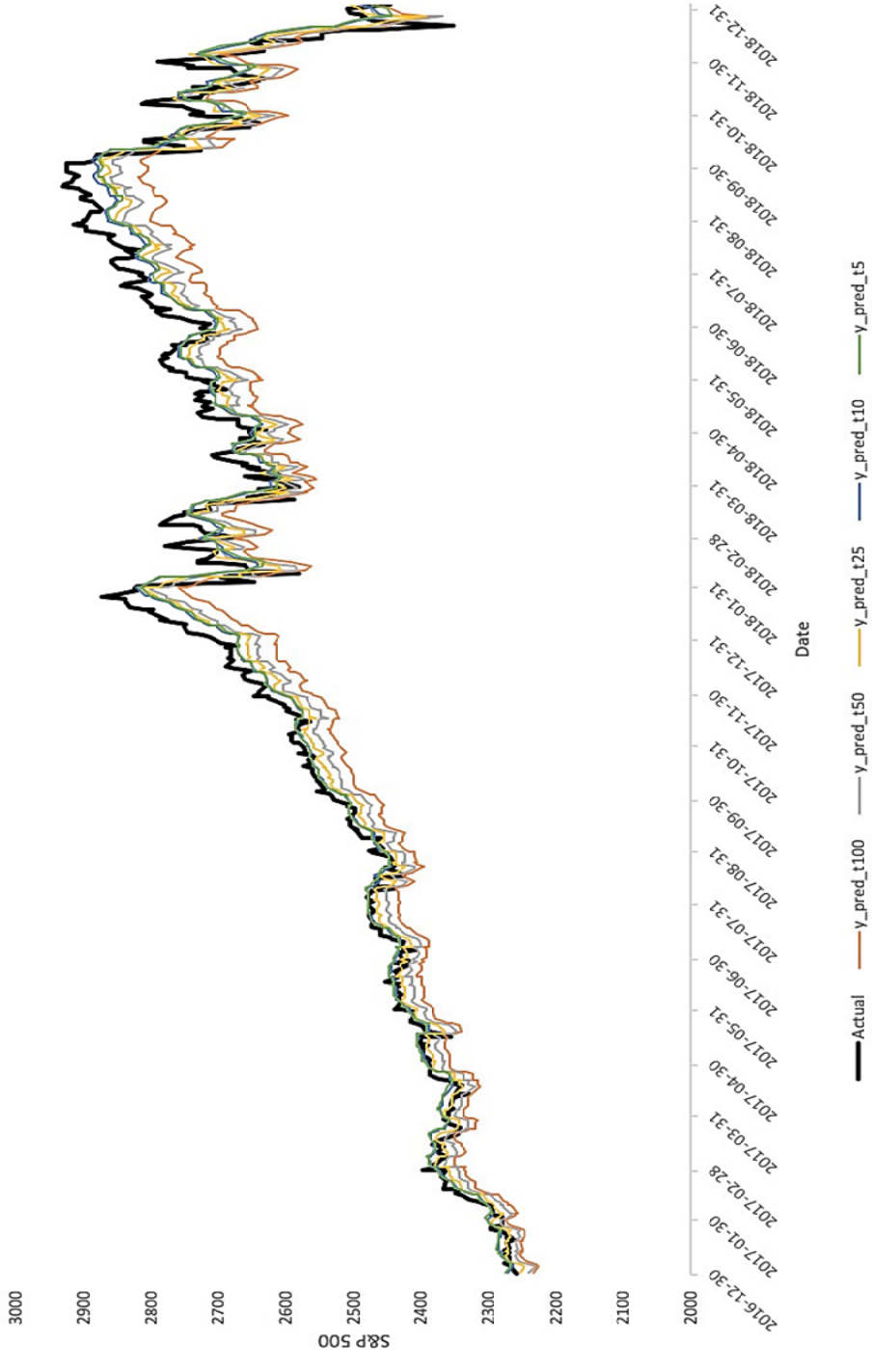
Figure 13 Models prediction NYSE stock (see online version for colours)



4.6 Performance of CNN-Stacked LSTM model on S&P 500 dataset

The performance of a CNN-Stacked LSTM model on the S&P 500 dataset was assessed through graphical analysis. The model's predictions for the next 5, 10, 25, 50, and 100 days were plotted and examined. Figure 14 provides visual representations of the model's forecasting capabilities and can be used to evaluate its accuracy and reliability in predicting future trends in the S&P 500 dataset. This hybrid architecture allows the model to effectively capture both spatial and temporal patterns in the data. The results showed that the CNN-Stacked LSTM model achieved promising performance in predicting the S&P 500 dataset. Further analysis of the graphs can offer insights into the model's performance and potential applications in financial forecasting and investment decision-making.

Figure 14 Performance of CNN-stacked LSTM model on S&P 500 dataset (see online version for colours)



5 Conclusions

Designing a customised CNN-LSTM model can initially be challenging due to the existence of numerous algorithms for stock data prediction that is not fully optimised. To address this, we conducted training and testing using various datasets such as NYSE, NASDAQ, and NIFTY and observed variations in the model's accuracy based on the dataset. Notably, for the NIFTY dataset, the model exhibited excellent performance, accurately predicting up to 99% of the stocks even during the testing phase. However, when dealing with datasets like NYSE, the accuracy varied during testing, possibly due to noise present in the dataset during that phase. We measured the MSE during training and testing, obtaining values ranging from 0.001 to 0.05 (approx) and 0.002 to 0.1 (approx) respectively, depending on the dataset. The paper focuses on demonstrating how the combination of CNN and LSTM can extract features from processed dataset tensors and detect patterns. It presents an approach for predicting stock market movements with a high level of accuracy.

As a future work, several potential areas for future exploration in the project include investigating the use of gated recurrent units (GRUs) as they have shown promising performance compared to LSTM models. Optimisation of hyper parameters, loss functions, and optimisers could enhance results, as the current parameters are arbitrary. Transfer learning could be expanded to train models on multiple stocks, potentially revealing hidden market structures. Ensembling techniques combining technical and fundamental indicators may improve model performance. Lastly, studying the impact of data variations, such as different technical indicators and timeframes, could yield better insights.

References

Adebiyi, A.A., Adewumi, A.O. and Ayo, C.K. (2014) 'Comparison of ARIMA and artificial neural networks models for stock price prediction', *Journal of Applied Mathematics*, Vol. 2014, pp.181–201.

Bansal, G., Hasija, V., Chamola, V., Kumar, N. and Guizani, M. (2019) 'Smart stock exchange market: a secure predictive decentralized model', in *2019 IEEE Global Communications Conference (GLOBECOM)*, IEEE, December, pp.1–6.

Gers, F.A., Schmidhuber, J. and Cummins, F. (1999) 'Learning to forget: continual prediction with LSTM', *1999 Ninth International Conference on Artificial Neural Networks ICANN 99*, Conf. Publ. No. 470, Edinburgh, UK, Vol. 2, pp.850–855, DOI: 10.1049/cp:19991218.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*, The MIT Press, Washington.

Gurav, U. and Kotrappa, D.S. (2020) 'Impact of COVID-19 on stock market performance using efficient and predictive LBL-LSTM based mathematical model', *International Journal on Emerging Technologies*, Vol. 11, No. 4, pp.108–115.

Guresen, E., Kayakutlu, G. and Daim, T.U. (2011) 'Using artificial neural network models in stock market index prediction', *Expert Systems with Applications*, Vol. 38, No. 8, pp.10389–10397.

Hiransha, M., Gopalakrishnan, E.A., Menon, V.K. and Soman, K.P. (2018) 'NSE stock market prediction using deep-learning models', *Procedia Computer Science*, Vol. 132, pp.1351–1362.

Kaggle (2017) *Huge Stock Market Dataset* [online] https://www.kaggle.com/borismarjanovic/price-volume-data-for-allus-stocks-etfs.

Kaggle (2020) *NIFTY-50 Stock Market Data (2000–2021)* [online] https://www.kaggle.com/rohanrao/nifty50-stock-market-data.

Kaggle (2022) *Stock Market Data (NASDAQ, NYSE, S&P500)* [online] https://www.kaggle.com/paultimothymooney/stock-market-data.

Kimoto, T., Asakawa, K., Yoda, M. and Takeoka, M. (1990) 'Stock market prediction system with modular neural networks', in *1990 IJCNN International Joint Conference on Neural Networks*, IEEE, June, pp.1–6.

Kingma, D.P. and Ba, J. (2014) 'Adam: a method for stochastic optimization', *International Conference on Learning Representations*.

Kompella, S. and Chakravarthy Chilukuri, K.C.C. (2020) 'Stock market prediction using machine learning methods', *International Journal of Computer Engineering and Technology*, Vol. 10, No. 3, p.2019.

Kraus, M. and Feuerriegel, S. (2017) 'Decision support from financial disclosures with deep neural networks and transfer learning', *Decision Support Systems*, December, Vol. 104, No. C, pp.38–48.

Liu, J., Chao, F., Lin, Y-C. and Lin, C-M. (2019) *Stock Prices Prediction using Deep Learning Models*, arXiv preprint arXiv: 1909.12227.

Nelson, D.M., Pereira, A.C. and de Oliveira, R.A. (2017) 'Stock market's price movement prediction with LSTM neural networks', in *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, May, pp.1419–1426.

Nti, I.K., Adekoya, A.F. and Weyori, B.A. (2021) 'A novel multisource information-fusion predictive framework based on deep neural networks for accuracy enhancement in stock market prediction', *Journal of Big Data*, Vol. 8, No. 1, pp.1–28.

Olah, C. (2015) *Understanding LSTM Networks* [online] http://colah.github.io/posts/2015-08-Understanding-LSTMs.

Pang, X., Zhou, Y., Wang, P., Lin, W. and Chang, V. (2020) 'An innovative neural network approach for stock market prediction', *The Journal of Supercomputing*, Vol. 76, No. 3, pp.2098–2118.

Pascanu, R., Mikolov, T. and Bengio, Y. (2013) 'On the difficulty of training recurrent neural networks', in *International Conference on Machine Learning*, pp.1310–1318.

Powers, D. and Ailab (2011) 'Evaluation: from precision, recall and F-measure to ROC, informedness, markedness & correlation', *J. Mach. Learn. Technol*., Vol. 2. pp.2229–3981, DOI: 10.9735/2229-3981.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E.A., Menon, V.K. and Soman, K.P. (2017) 'Stock price prediction using LSTM, RNN and CNN-sliding window model', in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, September, pp.1643–1647.

Shah, D., Campbell, W. and Zulkernine, F.H. (2018) 'A comparative study of LSTM and DNN for stock market forecasting', in *IEEE International Conference on Big Data (Big Data)*, pp.4148–4155.

Song, D., Qin, Y., Chen, H., Cheng, W. and Jiang, G. (2022) *Deep Reinforcement Learning for Asset Allocation and Portfolio Management: A Survey*, arXiv preprint arXiv: 2201.02663.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014) 'Dropout: a simple way to prevent neural networks from overfitting', *Journal of Machine Learning Research*, Vol. 15, No. 56, pp.1929–1958.

Tibshirani, R. (1996) 'L1 regularization: regression shrinkage and selection via the Lasso', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 58, No. 1, pp.267–288.

Tikhonov, A.N. (1943) 'L2 regularization: on the stability of inverse problems', *Doklady Akademii Nauk SSSR*, Vol. 39, Nos. 5–6, pp.195–198.

Vasista, K. (2022) 'Role of a stock exchange in buying and selling shares', *Int. J. Curr. Sci*., Vol. 12, No. 1, pp.1770–2250.

Wold, S., Esbensen, K. and Geladi, P. (1987) 'Principal component analysis', *Chemometrics and Intelligent Laboratory Systems*, Vol. 2, Nos. 1–3, pp.37–52.

Yoo, J., Soun, Y., Park, Y. and Kang, U. (2022) 'Accurate stock movement prediction with self-supervised learning from sparse noisy tweets', in *2022 IEEE International Conference on Big Data (Big Data)*, IEEE, pp.1–11.