Ushus-Journal of Business Management
2023, Vol. 22, No. 4, *15-28*
ISSN 0975-3311 | https://doi.org/10.12725/ujbm.65.2

Maximizing Profit Prediction: Forecasting Future Trends with LSTM Algorithm and compared with Loss function and Mean error code using Python

S. Sindhu*, N. Vijayalakshmi*, S. Sanjay Kumar* and G. Deepan Kumar*

Abstract

Profit prediction is a pivotal task in financial markets, empowering investors and traders to make informed decisions. In recent years, the advent of deep learning techniques has revolutionized the field of financial forecasting, offering the potential to extract intricate patterns and relationships from vast and complex datasets. This paper presents an innovative approach to profit prediction using Long Short-Term Memory (LSTM) networks, a specialized type of Recurrent Neural Network {RNN). LSTMs excel at capturing long term dependencies in sequential data, making them well-suited for modeling the dynamics of the financial markets. The core of the paper lies in the practical application of LSTM model architecture specially tailored for profit prediction. This includes defining the input layer, LSTM layers, fully connected layers and the output layer. The training and validation process is elucidated, covering data splitting, model training, validation techniques and hyper parameter tuning to enter ensure the model performance. The paper also explores the practical application of the LSTM-based profit prediction algorithm through a case

* Department of Computer Science and Application, SRM Institute of Scie--nce and Technology, Bharathi Salai, Ramapuram, Chennai, Tamil Nadu; sindhus1@srmist.edu.in, ss5506@srmist.edu.in, dk2128@srmist.edu.in

study involving real-world financial data. Evaluation metrics such as Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are employed to assess the algorithm's predictive accuracy and effectiveness. Additionally, the paper addresses risk assessment, a critical aspect of profit prediction in financial markets. It sheds light on the promising potential of LSTM-based profit prediction algorithms as a powerful tool for financial forecasting. It summarizes key findings, acknowledges limitations and challenges, and outlines future directions for improving the algorithm, including incorporating additional data sources and fine-tuning hyper parameters. The presented approach offers a significant advancement in the realm of profit prediction, enabling investors and traders to make more informed and data-driven decisions in an ever-evolving financial landscape.

Keywords: LSTM, Profit prediction, Evaluation metrics, Financial-forecasting

1. Introduction

In the world of finance, the ability to forecast profits accurately is the cornerstone of success. It's a domain where decisions are made in the blink of an eye, and the stakes are high[1]. Historically, profit prediction in financial markets has relied heavily on traditional statistical methods and econometric models. While these techniques have served us well, they often struggle to capture the intricate and evolving dynamics of today's financial landscapes.[2-4]

This is where LSTM, a specialized type of recurrent neural network (RNN), enters the scene. LSTM networks have emerged as a powerful tool in the realm of deep learning, renowned for their ability to handle sequential data with remarkable precision. What sets LSTMs apart from other neural network architectures is their unique capacity to capture long-term dependencies in data, making them particularly well-suited for modeling complex financial time series.

In this paper, we delve into the world of LSTM-based profit prediction, and we do so with Python—a programming language renowned for its simplicity[5], versatility, and a vast ecosystem of libraries and tools. Python has become the lingua franca of data science and machine learning, making it an ideal choice for implementing and experimenting with LSTM networks in our profit prediction endeavour[6].

Our proposed will walk you through the key components of this exciting project:

a. **Data Preprocessing:** We will explore how we collected and prepared our financial data, cleaning and transforming it to make it suitable for LSTM modeling.

b. **LSTM Model Architecture in Python:** We will unveil the inner workings of our LSTM-based profit prediction model, showcasing how Python code brings this architecture to life.

c. **Training and Validation:** We'll discuss the critical steps of training and validating our LSTM model, illustrating how Python libraries like TensorFlow and Keras streamline this process.

d. **Profit Prediction and Risk Assessment:** We will demonstrate how our LSTM-based algorithm can make profit predictions and assess the associated risks using Python's data visualization and analytics capabilities.

e. **Case Study:** To provide a tangible understanding of our approach, we'll present a real-world case study that showcases the practical application of our LSTM model in financial markets.

2. Proposed Methodology

In this paper, Long Short-Term Memory (LSTM) Networks method is proposed for the forecasting of the profit with higher accuracy score. LSTM is a type of recurrent neural network (RNN) designed to handle sequences of data[7-9]. It's particularly well-suited for tasks involving time series data, such as predicting stock prices or financial market trends, due to its ability to capture long-term dependencies within sequences[10]. Unlike traditional RNNs, LSTMs have mechanisms that allow them to remember and forget information over extended time steps, making them highly effective for modeling complex and dynamic patterns.

Key Components of LSTM:

Cell State (Ct): The cell state is the central component of an LSTM. It acts as a conveyor belt that runs through the entire sequence, carrying information from one time step to another. It can selectively forget or store information using gates[11].

Hidden State (ht): The hidden state at each time step is computed based on the input data, the previous hidden state, and the cell state. It carries information that the LSTM has deemed relevant[12].

Gates: LSTMs have three types of gates:

i. **Forget Gate:** Determines what information from the cell state should be thrown away or kept.

ii. **Input Gate:** Updates the cell state with new information.

iii. **Output Gate:** Produces the output based on the cell state.

3. Proposed Architecture

LSTMs deal with both Long-Term Memory (LTM) and Short-Term Memory (STM) and for making the calculations simple and effective it uses the concept of gates.

- Forget Gate: LTM goes to forget gate and it forgets information that is not useful.
- Learn Gate: Event (current input) and STM are combined together so that necessary information that we have recently learned from STM can be applied to the current input[13].
- Remember Gate: LTM information that we haven't forget and STM and Event are combined together in Remember gate which works as updated LTM.

- Use Gate: This gate also uses LTM, STM, and Event to predict the output of the current event which works as an updated STM.

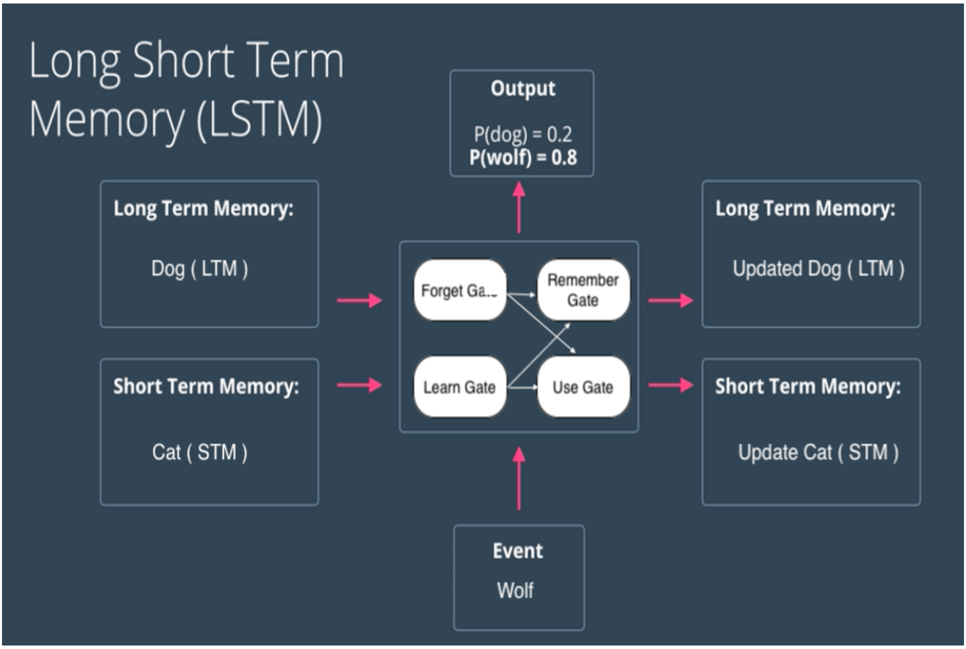

Fig:1 Common architecture for the proposed LSTM methodology

The above figure shows the simplified architecture of LSTMs. The actual mathematical architecture of LSTM is represented using the following figure

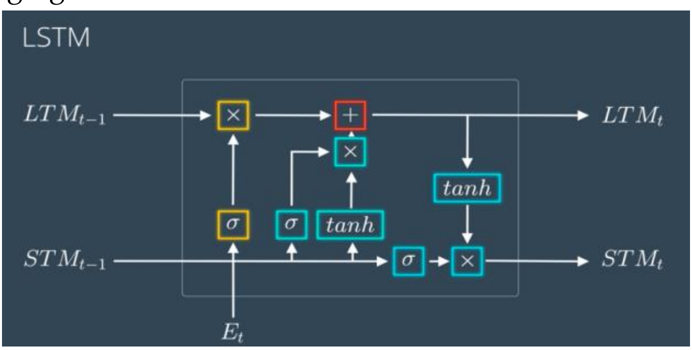

Fig 2: LSTM Architecture

don't go haywire with this architecture we will break it down into simpler steps which will make this a piece of cake to grab.

Breaking Down the Architecture of LSTM:

a. **Learn Gate:** Takes Event (Et) and Previous Short-Term Memory (STMt-1) as input and keeps only relevant information for prediction.

- Previous Short Term Memory STMt-1 and Current Event vector Et are joined together [STMt-1, Et] and multiplied with the weight matrix W_n having some bias which is then passed to tanh (hyperbolic Tangent) function to introduce non-linearity to it, and finally creates a matrix N_t[14-18].
- For ignoring insignificant information, we calculate one Ignore Factor it, for which we join Short-Term Memory STMt-1 and Current Event vector Et and multiply with weight matrix W_i and passed through Sigmoid activation function with some bias.
- Learn Matrix N_t and Ignore Factor it is multiplied together to produce learn gate result.

b. **The Forget Gate**: Takes Previous Long Term Memory (LTMt-1) as input and decides on which information should be kept and which to forget[19-20].

- Previous Short-Term Memory STMt-1 and Current Event vector Et are joined together [STMt-1, Et] and multiplied with the weight matrix W_f and passed through the Sigmoid activation function with some bias to form Forget Factor ft.
- Forget Factor ft is then multiplied with the Previous Long-Term Memory (LTMt-1) to produce forget gate output.

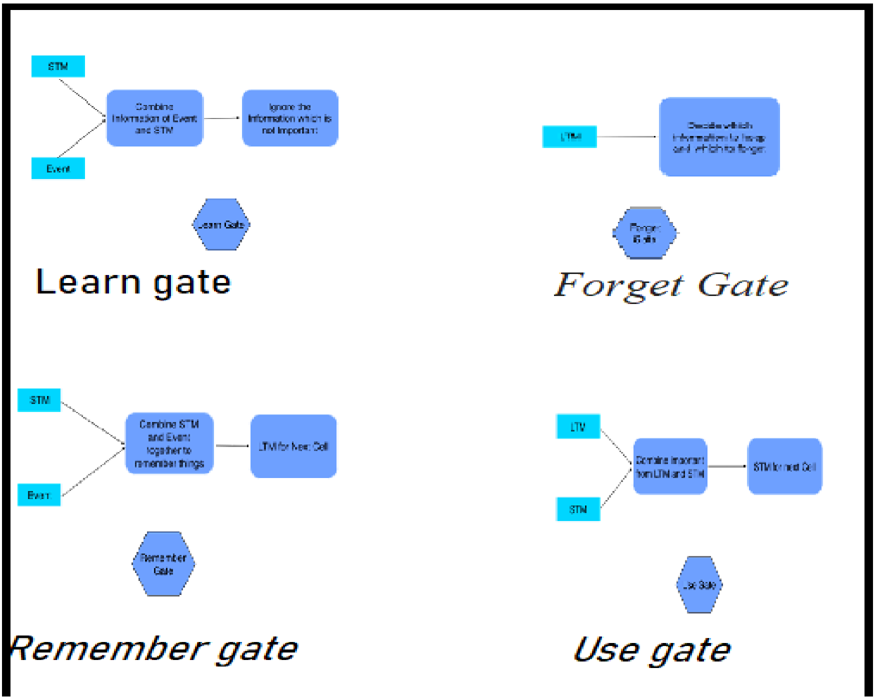

Fig 3: Various gate operations in LSTM (Forget, Use, Remember and Learn)

4. The Remember Gate: Combine Previous Short-Term Memory (STMt-1) and Current Event (Et) to produce output.

   - The output of Forget Gate and Learn Gate are added together to produce an output of Remember Gate which would be LTM for the next cell.

5. The Use Gate:

Combine important information from Previous Long-Term Memory and Previous Short-Term Memory to create STM for next and cell and produce output for the current event.

- Previous Long-Term Memory (LTM-1) is passed through Tangent activation function with some bias to produce Ut.
- Previous Short-Term Memory (STMt-1) and Current Event (Et)are joined together and passed through Sigmoid activation function with some bias to produce Vt.

- Output Ut and Vt are then multiplied together to produce the output of the use gate which also works as STM for the next cell [21-22].

6. Dataset

The Kaggle repository, which has 639 records and seven different fields, is where the dataset was gathered. Date, High, Low, Open, Close, Volume, and Adj Close are all included in the dataset, which is a TESLA'22 report. It includes the 2019–2022 report.

2022-01-03	1201.069946	1136.040039	1147.75	1199.780029	34643800	1199.780029
2022-01-04	1208	1123.050049	1189.550049	1149.589966	33416100	1149.589966
2022-01-05	1170.339966	1081.01001	1146.650024	1088.119995	26706600	1088.119995
2022-01-06	1088	1020.5	1077	1064.699951	30112200	1064.699951
2022-01-07	1080.930054	1010	1080.369995	1026.959961	28054900	1026.959961
2022-01-10	1059.099976	980	1000	1058.119995	30605000	1058.119995
2022-01-11	1075.849976	1038.819946	1053.670044	1064.40024	22021100	1064.400024
2022-01-12	1114.839966	1072.589966	1078.849976	1106.219971	27913000	1106.219971
2022-01-13	1115.599976	1026.540039	1109.069946	1031.560059	32403300	1031.560006
2022-01-14	1052	1013.380005	1019.880005	1049.609985	24308100	1049.609985
2022-01-18	1070.790039	1016.059998	1026.609985	1030.51001	22247800	1030.51001
2022-01-19	1054.670044	995	1041.709961	995.6500244	25147500	995.6500244
2022-01-20	1041.660034	994	1009.92998	996.2700195	23496200	996.2700195
2022-01-21	1004.549988	940.5	996.3400269	943.9000244	34472000	943.9000244
2022-01-24	933.510098	903.210022	914.2000122	918.4000244	288865300	918.4000244
2022-01-25	951.2600098	903.210022	914.2000122	918.4000244	28865300	918.4000244
2022-01-26	987.6900024	906	952.4299927	937.4099731	34955800	937.4099731
2022-01-27	935.3900146	829	933.3599854	829.0999756	49036500	829.0999756

Fig 4: TESLA'2022 price prediction report in the year of 2019 to 2022

This dataset was sufficient to completely train the LSTM model, giving it the ability to forecast closing and opening prices[24].

7. Implementation Proposed work

Python offers several libraries and frameworks for implementing LSTMs, with TensorFlow and Keras being among the most popular choices. Here's a simplified overview of how you can implement an LSTM model for profit prediction in Python:

Step 1: Data Preprocessing:

Data to be collected from the different resources. The dataset which is used for the proposed work contains around 1200 share profit report for the five different countries with the 14 atrributes. The collected dataset was pre-processed and removed the unwanted data from the dataset to avoid the overfitting to the model.

```
regressor.compile(optimizer = 'adam',loss = 'mean_squared_error')

regressor.fit(x_train,y_train,epochs =300, batch_size = 32)
```

```
Epoch 18/300
15/15 [==============================] - 3s 196ms/step - loss: 0.0028
Epoch 19/300
15/15 [==============================] - 4s 267ms/step - loss: 0.0021
Epoch 20/300
15/15 [==============================] - 2s 151ms/step - loss: 0.0019
Epoch 21/300
15/15 [==============================] - 2s 117ms/step - loss: 0.0019
Epoch 22/300
15/15 [==============================] - 2s 115ms/step - loss: 0.0026
Epoch 23/300
15/15 [==============================] - 3s 193ms/step - loss: 0.0019
Epoch 24/300
```

Fig 5: Model trained and find the error of loss function for monitoring the higher accuracy

Steps 2: To train the LSTM Model:

In Python code, you'll define your LSTM model. This typically involves creating a Sequential model and adding layers:

Input Layer-> Specify the input shape.

LSTM Layers-> Configure the number of units and other hyperparameters.

Fully Connected Layers-> Add one or more dense layers for prediction.

Output Layer-> Define the output layer, usually with a single neuron for profit prediction.

Step 3: Checks the accuracy level

With Error code metrics: After execution of the model, the model has already trained by the preferred dataset, to propose the check the current model test score with loss function and mean absolute error techniques. It helps to produce the better accuracy of the model. **O**ptimizer, and evaluation metrics for profit prediction, Mean Absolute Error (MAE) or Mean Squared Error (MSE) can be suitable loss functions.

Step 4: Train the Model

Fit your model to the training data. This step involves feeding your training data into the model and adjusting the model's weights through backpropagation.

Step 5: Validation and Testing:

Assess your model's performance using validation data and calculate evaluation metrics like RMSE or MAE.

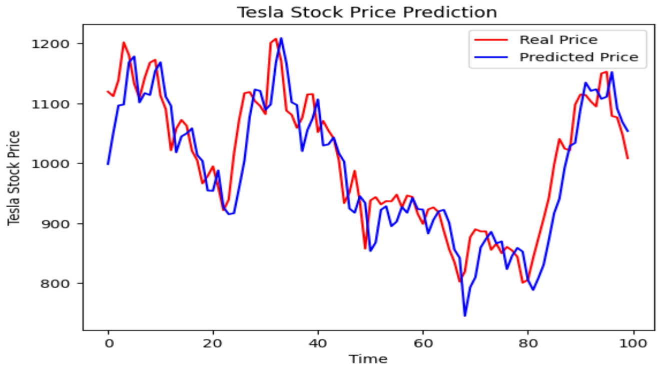

Fig 6: Tesla report price prediction time elapment

Step 6: Test the Model, Profit Prediction:

Once trained, your LSTM model can be used to predict future profits based on input data.

Step 7: Visualization:

Use Python's data visualization libraries to plot predictions, actual profits, and evaluate the model's performance visually.

Step 8: Fine-Tuning:

Iterate on your model by fine-tuning hyper parameters and exploring advanced LSTM variations to optimize performance.

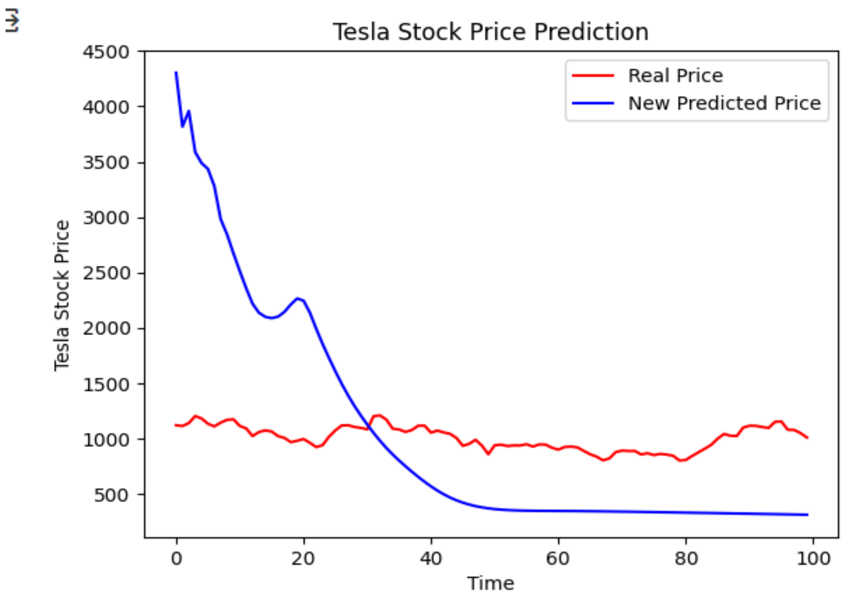

Fig 7: Price prediction time and new price predicted report with time taken

This high-level overview gives you a sense of how LSTM-based profit prediction can be implemented using Python. The specific code details would depend on your dataset and project requirements. Python’s rich ecosystem of libraries and resources makes it an ideal choice for implementing sophisticated deep learning models like LSTM for financial forecasting.

Conclusion

Our research offers a bridge between cutting-edge deep learning techniques and the world of financial analysis, underpinned by the elegance and efficiency of the Python programming language. We aim to highlight not only the potential but also the accessibility of

LSTM-based profit prediction in today's data-driven financial landscape. By the end of this presentation, we hope you will appreciate the synergy between LSTM and Python, which holds the promise of enhancing decision-making processes in the financial world. Thank you for joining us on this journey into the exciting realm of LSTM-based profit prediction with Python.

References:

1. S. Saydam, J. Coulton and C. Sammut, "Alternative techniques for forecasting mineral commodity prices", International Journal of Mining Science and Technology, vol. 28, no. 2, pp. 309-322, March 2018, [online] Available: https://doi.org/10.1016/j.ijmst.2017.09.001.
2. G. Cortazar, I. Kovacevic and E. Schwartz, "Commodity and Asset Pricing Models: An Integration", National Bureau of Economic Research, June 2013, [online] Available: https://doi.org/10.3386/w19167.
3. A. Sharma, D. Bhuriya and U. Singh, "Survey of stock market prediction using machine learning approach", 2017 International Conference of Electronics Communication and Aerospace Technology (ICECA), 2017, April, [online] Available: https://doi.org/10.1109/iceca.2017.8212715.
4. M. A. Istiake Sunny, M. M. S. Maswood and A.G. Alharbi, "Deep Learning-Based Stock Price Prediction Using LSTM and Bi-Directional LSTM Model", 2020 2nd Novel Intelligen t and Leading Emerging Sciences Conference (NILES), 2020, October 24, [online] Available: https://doi.or g/10.1109/niles-50944.2020.9257950.
5. Eugene F. Fama, "The Behavior of Stock Market Prices", the Journal of Business, vol. 2, no. 2, pp. 7-26, January 1965.
6. Stock market challenges, [online] Available: http://www.g oogle.com.

7. A Sheta, "Software Effort Estimation and Stock Market Prediction Using Takagi-Sugeno Fuzzy Models", InProceedings of The IEEE International Conference on Fuzzy Systems, pp. 171-178, 2006.
8. S. Selvin, R. Vinayakumar, E. Gopalakrishnan, V. K. Menon and K. Soman, "Stock price prediction using lstm rnn and cnn-sliding window model", 2017 international conference on advances in computing communications and informatics (icacci), pp. 1643-1647, 2017.
9. K. Chen, Y. Zhou and F. Dai, "A lstm-based method for stock returns prediction: A case study of china stock market", 2015 IEEE international conference on big data (big data), pp. 2823-2824, 2015.
10. M. Roondiwala, H. Patel and S. Varma, "Predicting stock prices using lstm", International Journal of Science and Research (IJSR), vol. 6, no. 4, pp. 1754-1756, 2017.
11. "Meaning Factors & Advantages", Groww, September 2022, [online] Available: https://groww.in/p/whatis-mcx-trading.
12. moneycontrol.com, [online] Available: https://www.moneycontrol.com/commodity/.
13. Commodity Forecast 2022/2023. (n.d.), September 2022, [online] Available: https://tradingeconomics.com/forecast/commodity.
14. International Monetary Fund, [online] Available: https://www.imf.org/external/pubs/ft/wp/2004/wp0441.pdf.
15. Understanding LSTM Networks -- colah’s blog. (n.d.-b), September 2022, [online] Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.
16. G. V. Jose, "Predicting Sequential Data using LSTM: An Introduction", Medium, December 2021, [online] Available: https://towardsdatascience.com/time-series-forecasting-withrecurrent-neural-networks-74674e289816.

17. S. M. L. Club, "Using LSTMs to Predict Future Stock Prices - Visionary Hub", Medium, January 2022, [online] Available https://medium.com/visionary-hub/using-lstms-to-predict-futurestock-prices-61f4458fc860.
18. TensorFlow, September 2022, [online] Available: https://www.tensorflow.org/tutorials/structured_data/time_series.
19. K. Kuguoglu, "Predicting future values with RNN LSTM and GRU using PyTorch", Medium, January 2022, [online] Available: https://towardsdatascience.com/predicting-future-valueswith-rnn-lstm-and-gru-using-pytorch-d9ef50991ec d9ef50991ec7.
20. Investopedia, August 2022, [online] Available: https://www.investopedia.com/terms/c/commodity.asp.
21. K. Shenmare, "Commodity Market", WallStreetMojo, September 2022, [online] Available: https://www.wallstreetmojo.com/commodity-market/.
22. 5paise.Com, September 2022, [online] Available: https://www.5paisa.com/stockmarket-guide/commodity-trading/how-commodity-market-works-inindia.
23. G. V. Jose, "Predicting Sequential Data using LSTM: An Introduction", Medium, December 2021, [online] Available: https://towardsdatascience.com/time-series-forecasting-withrecurrent-neural-networks-74674e289816.
24. Z. He, J. Zhou, H. N. Dai and H. Wang, "Gold Price Forecast Based on LSTM-CNN Model", 2019 IEEE Intl Conf on Dependable Autonomic and Secure Computing Intl Conf on Pervasive Intelligence and Computing Intl Conf on Cloud and Big Data Computing Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), 2019, August, [online] Available: https://doi.org/10.1109/dasc/picom/cbdcom/cyberscitech.2019.00188.