International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Hyperparameter Tuning of the LSTM model for Stock Price Prediction

Vikas Deswal [1], Dharminder Kumar [2]

Submitted:12/03/2024 **Revised**: 27/04/2024 **Accepted**: 04/05/2024

Abstract: The stock market serves as a mirror, revealing the actual state of the nation's economy. Experts can monitor the nation's economic status by following the stock market's fluctuations. Predicting the stock market is so essential in the cutthroat world of today. Because stock prices are chaotic, dynamic, and nonlinear, predicting them is challenging. Stock price forecasting is aided by deep learning methods such as LSTM. The algorithm's forecast is erroneous since its hyperparameter was not chosen correctly.

This work contributes to developing a hyper-tuned LSTM model for stock price prediction. Numerous hyperparameters are included, such as neurons, batch size, epoch, learning rate, and dropout rate. The primary goal is to identify the optimal set of parameters that will enable the LSTM forecasting algorithm to operate at a high level of performance. Three widely used error metrics are used to assess algorithm performance: R^2, which indicates how well our predictions match the actual data; MSE, which displays the discrepancy between the predicted and actual data; and MAE, which indicates the average deviation between our predictions and the actual data. For training, testing, and validating the data set, values of three error metrics for various parameter combinations are gathered. The best value of these error metrics helps in selecting the best possible combination of parameters.

A proven prediction method called Adaboost is compared with the output error metrics of the LSTM model to confirm our hyperparameter tunning efforts. The LSTM model's potential for precise stock price prediction is strongly confirmed if its error metrics value is comparable to or superior to Adaboost's.

Keywords: *LSTM, Adaboost, MSE, MAE, R^2, epoch, batch size, neurons, learning rate, and dropout rate1.*

Introduction

The buying and selling of stocks on the financial markets has attracted a large number of players who are looking to make money. Forecasting stock movements has grown in popularity, particularly within the last 20 years. Regrettably, most people find it difficult to generate a steady income from stock trading. Many people, especially those who are new to the game, compare it closely to gambling since they believe that future price swings are only educated guesses. Given the inherent volatility and unpredictability of the stock market, even seasoned traders recognize that profits cannot be assured. To reduce risk in the financial markets, particularly during trading activities, stock movement prediction is essential. To help people make wise financial decisions, this topic is the focus of extensive research efforts [1]. One common method is to analyze stock price data using time-series analysis. Such predictions can be made using methods such as neural networks, LSTM, SVM, Random Forest, and linear regression. Research has indicated that LSTM frequently performs better than alternative techniques like neural networks and linear regression [2]. Simple LSTM models produce less-than-ideal outcomes because they fail to minimize the loss function, which increases model error. This means that a hyper-tuned LSTM model must be used. What distinguishes a hyperparameter from a parameter? The suggested approach learns by estimating model parameters for the provided data set and then keeps updating these values. These parameters are incorporated into the model once learning is finished. On the other hand, hyperparameters are algorithm-specific. Therefore, the data cannot be used to determine their values. To calculate the model parameters, hyperparameters are needed [3].

Hyperparameter tuning includes searching a pair of the best possible combinations of parameters for the LSTM algorithm and applying this improved method to every set of data. That combination of hyperparameters maximizes the model's performance and minimizes a predetermined loss function, yielding better outcomes with fewer errors. Using the input data as a basis, the learning algorithm optimizes the loss and looks for the best answer within the parameters.

1.1 LSTM model hyperparameters:

Number of neurons per layer: Searching for the accurate quantity of neurons for every LSTM network layer is a difficult problem. too few neurons will cause the LSTM to become unable to recall every part of information required for generating complete forecasting, while too many neurons will cause the LSTM to overfit the training set [4].

[1] Research Scholar, Phd.(CSE),Department of Computer Science & Engineering, GJUS&T, Hisar, Haryana, India
ORCID ID: 0000-0002-0753-6807
[2] Professor(Retd.), Department of Computer Science & Engineering, GJUS&T, Hisar, Haryana, India
ORCID ID:0000-0002-4662-7584
** Corresponding Author Email: ervikasdeswal@gmail.com*

Number of hidden layers: This layer suggests that a deep learning neural network's model is much more productive as compared to a simple model. As a result, it is crucial to use multiple layers to test the model's performance [5].

Dropout rate: By boosting the model's generations, using a different dropout rate could enhance the LSTM model's performance. [6].

Learning rate: The weighting and biases of the model are altered by this hyperparameter. Selecting the right learning rate is essential for achieving the best possible model performance because it affects the model's bias and weighting [4].

Optimizers: LSTM models use an optimizer whose job is to minimize an objective function. Combining optimizer with learning can boost model performance [4].

Epochs: The epochs must be supplied while training an LSTM model to prevent underfitting or overfitting. Underfitting will occur when the quantity of epochs is too less, while overfitting will occur when it is too large.

It is now necessary to understand how to select their ideal values. Both manual and automated techniques can be used to determine these ideal hyperparameter values. While automated hyperparameter tuning uses an algorithm to find the ideal settings, human hyperparameter tuning selects values by trial and error. Grid, random, and Bayesian methods are a few of the techniques [7]. Due to the challenge of precisely fine-tuning LSTM to yield an ideal and reliable outcome, multiple hyperparameter adjustments are required to enhance model performance in time series applications. As a result, it will be beneficial to train an algorithm on multiple similar datasets to build up a body of knowledge that might be used to enhance predictions for certain time series [8].

The grid search technique is used to adjust the LSTM algorithm's hyperparameters. It is similar to using a hyperparameter tuning brute force approach. The model was then fitted using a grid of potential discrete hyperparameter values in every feasible combination. LSTM has been optimized for stock market prediction using this approach, and the results have been compared with the benchmark model. The hyper-tuned LSTM model's result is almost identical to the benchmark model, demonstrating the hyper-tuned model's efficacy [9].

The target of this study is to determine how changing the LSTM model's hyperparameter values affects the precision of stock price predictions. These hyperparameters include the learning algorithm, optimizer, epoch, learning rate, number of layers, batch size, and dropout. Performance metrics like as MSE, MAE, and R2 values will be used to assess the impact of changing the combination of hyperparameter values.

The key aim of this research is

- To search the optimal pair of hyperparameters of the LSTM model.
- To find the outcome of the suggested model using various hyperparameter settings during the training, testing, and validation phases.

The data set is split into 80%,10%, and 10% for training, testing, and validation of the dataset. The performance of the LSTM model is then accessed using MSE, MAE, and coefficient of determination (R2) and compared with the Ada boost ensembled learning algorithm.

This is how the remainder of the paper is structured. A review of relevant literature on the hyperparameter tuning of the LSTM model is presented in Section 2. Section 3 will detail the setup and configuration of the experiment. In Section 4, research data and experimentation are presented, and the suggested approach is contrasted with a reference model. The thorough result analysis and discussion are described in Section 5. Section 6 shows a summary and suggestions for further research.

2. Literature Review

This paper helps in the development of the LSTM model for share price forecasting. Optuna framework is given as input to the LSTM model for finding the best combination of hyperparameters. The optimum pair of hyperparameters offers the lowest loss and RMSE score [10].

The suggested hybrid model is examined in this work for famous machine learning-based CART, SVM, and XGBoost models as well as conventional NN. The precision levels achieved in NN, CART, SVM, and XGBoost models are 72.69%, 84.21%, 73.51%, and 90.81%. The suggested hybrid model substantially outperforms conventional ML models [11].

This study explains how a new approach to hyperparameter optimization is developed using the BO framework and swarm particle optimization. The particles can naturally be optimized in parallel using the suggested technique, which does not need the computation of gradients. Therefore, it is possible to effectively lower the computational complexity, resulting in better hyperparameters being acquired with the same amount of processing. Tests can be conducted using actual power load data, demonstrating that the suggested approach works better than the most advanced algorithms currently in use. The prediction error findings across many models demonstrate the effectiveness of BO-PSO as a hyper-parameter optimization technique [12].

In this research, a simple genetic algorithm strategy for hyperparameter tuning of a common language model is explored. It is significantly more efficient than fine-tuning

(grid search and trial & error search) and attains fine-tuning productivity without requiring an exhaustive search [13].

This work involves the optimization of hyperparameters through the use of two new greedy sequential methods and random search, all based on the expected improvement criterion. This work provides new methods for creating response surface models where many hyperparameter assignment elements are known to be meaningless given the specific values of the other elements [14].

In this work, the investigator offers both theoretical and empirical support for the claim that trials chosen at random are a more efficient method for hyperparameter tuning than grid-based trials. This work shows that adaptive hyperparameter optimization algorithm development can be evaluated naturally against the baseline of random search. Experts predict that the increased need for large hierarchical models will put more strain on methods for hyperparameter tuning [15].

In this study researcher mainly covered the application of several cutting-edge optimization approaches to machine learning algorithms. There are numerous frameworks and libraries available for solving hyperparameter optimization difficulties, and some unresolved issues in the field are also covered. Benchmark datasets are used in experiments to evaluate the efficiency of various optimization techniques and to offer real-world examples of hyperparameter optimization [16].

In this work researcher mainly addresses different tuning tactics and their effects on algorithm performance, as well as how parameters affect prediction performance. Additionally, tuneRanger's forecast accuracy and runtime are compared to those of other tuning implementations using the default hyperparameters [17].

In this study, the researchers aim to develop a dynamic online tuning strategy for the deep LSTM model's hyperparameters. The suggested method can be tailored to feed any time series-based application, especially ones that use data streams. The experiment's outcome demonstrates that dynamic deep LSTM hyperparameter adjustment outperforms the initial static tuning method [18].

This research paper focuses on hyperparameter optimization for LSTM to forecast SARS-CoV-2 infection cases in the Russian Federation, aiming to find the optimum set of parameters for a well-fitting model. The study evaluates nearly 10 unique forecasting models and conducts a comprehensive analysis and comparison of their results. The hyper-tuned LSTM model proves to be the best prediction model among all other models [19].

In this study, researchers use an LSTM model that has been fine-tuned to predict stock prices for the Indonesia composite index dataset. A metaheuristic approach was employed to optimize the tuning of hyperparameters. For this method, RMSE serves as both a fitness function and a key indicator. In this study, benchmarking algorithms GA and PSO were employed. With an RMSE of 78.79, the hybrid SOS-LSTM model fared better than both GA-LSTM and PSO-LSTM models, which had respective RMSEs of 142.663 and 529.170 [20].

In this research paper comparison is done among the ARIMA model, Se-optimized LSTM, and unoptimized LSTM. The main metric for optimization is RMSE. The best model found, SE-LSTM, yields an R2 score of approximately 961, an RMSE of approximately 538,914, an MAE of approximately 402.99, and a MAPE of approximately 1.437% [21].

3 Methodology

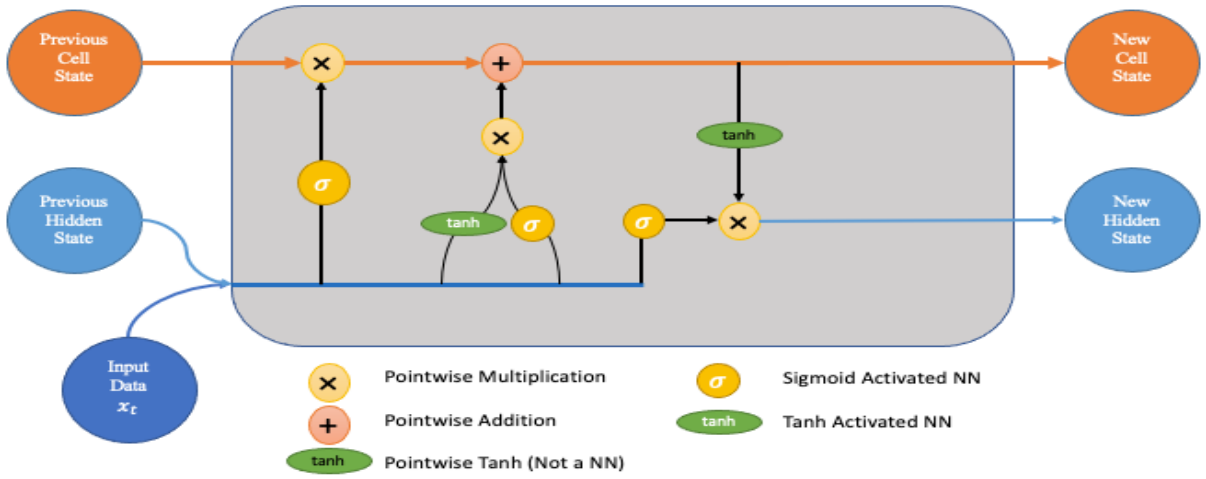
3.1 Long Short-Term Memory (LSTM) Network

This section provides an overview of the LSTM model, the experimental setup, and the hyperparameter settings. At the end, several performance metrics used to measure the performance of the LSTM will be discussed.

A particular kind of RNN known as LSTM was developed to address the limitations of traditional RNNs in managing long-term dependencies in sequence data. The disappearing and ballooning gradient problem in the original RNN architecture makes it challenging for the network to grasp long-range dependencies in sequence data. By using memory cells and gating mechanisms, the LSTM model—which was first presented by Hochreiter and Schmidhuber in 1997—addresses these problems. This new method enables long-term storage and propagation of information in LSTMs, allowing the model to learn from longer sequences [22]. An LSTM network's input, output, forget gates and memory cells are its essential parts. Long-term information is stored in memory cells, and information entering and leaving these cells is managed by gating processes. These gates enable the LSTM to efficiently learn and retain pertinent features from the input data by using sigmoid activation functions to make judgments depending on the input [23].

The input may only be retained in the cell state if the input gate authorizes it. The forget gate controls the state unit's weight. The gate's output value is computed using the updated memory cell state. The output gate can block the cell's output, all gates employ sigmoidal nonlinearity, and the state unit can function as an additional input for the remaining gate. Long-term dependencies can be resolved using the LSTM architecture in this way, with minimal computational expense [24].

Although LSTM is a powerful machine learning model, it must be trained and tuned carefully to yield the best results.



Dia 1.1 The LSTM model

Hyperparameter tuning consists of finding the optimal pair of hyperparameters of the LSTM model for maximizing the model performance. The model calculates the performance improvement that can be attained for each of the hyperparameters under consideration by altering the initial values to the values indicated in the goal configuration.

3.2 Performance metrics

MSE, MAE, and R^2 are three common metrics used to estimate the outcome of regression models.

3.2.1 Mean Square Error (MSE): This error is the mean of the square value of the error. Here, error refers to the difference between the estimated and actual numbers. Model accuracy and MSE value have an inverse relationship. Lower values indicate better performance [25].

$$MSE = \sum \frac{(actual - predicted) \wedge 2}{n}$$

3.2.2 Mean Absolute Error (MAE): The average of the error difference is called MAE. Here, error refers to the discrepancy between the estimated and real quantities. It gives us a gauge for the difference between the expected and actual values. This gives us an understanding of how the real value differs from the projected value [26]. Formulae for given metrics are given below:

$$MAE = \sum \frac{|actual - predicted|}{n}$$

3.2.3 Coefficient of Determination (R^2): The degree of data fit by a regression model is indicated by this coefficient. The square of R between the predicted and actual values is used to calculate it. A model's fit to the data set is shown by an R2 value, which ranges from 0 to 1. A value of 1 shows that the model fully fits the data set. When attempting to explain the volatility in the dependent variable, this metric works well [26]. The model prediction is directly correlated with the R2 value.

4. Research Data and Experiment

4.1 Data Information

The 15-year TSLA stock was the source of research data for this study, which was collected from Yahoo Finance. The share price data in the dataset spans the years 2010 through 2024, excluding weekends and holidays. The observed variables in the data set are Date, Open price, High price, Low price, Close price, Adj Close price, and Volume. The total dataset values from June 2010 to January 2024 are shown in Table 1.

Date	Open	High	Low	Close	Adj Close	Volume
29-06-2010	1.266667	1.666667	1.169333	1.592667	1.592667	281494500
30-06-2010	1.719333	2.028	1.553333	1.588667	1.588667	257806500
01-07-2010	1.666667	1.728	1.351333	1.464	1.464	123282000
02-07-2010	1.533333	1.54	1.247333	1.28	1.28	77097000
06-07-2010	1.333333	1.333333	1.055333	1.074	1.074	103003500
...	...	...	...	...	...	...
23-01-2024	211.3	215.64999	207.75	209.14	209.14	106605900
24-01-2024	211.88001	212.73	206.77	207.83	207.83	123369900
25-01-2024	189.7	193	180.06	182.63001	182.63001	198076800
26-01-2024	185.5	186.78	182.10001	183.25	183.25	107063400
29-01-2024	185.63001	191.48	183.67	190.92999	190.92999	124503900

Table 1 TSLA Data Set

4.2 Data Scaling

Initially, make sure the data is successfully imported from Yahoo Finance. Afterward, scale it with a range of 0 to 1 using the MinMax Scaler function. To fit into the specified range between the lowest and highest observed values, the function rescaled the values in each column. The lowest and highest values were determined for every column to guarantee uniform data scaling. Based on this determined range, the remaining data points in each column were then proportionately adjusted.

4.3 Data Splitting and reshaping

After scaling, the data was split into three sets: test (10%), validation (10%), and training (80%). The test set was not seen for the final evaluation; instead, the LSTM model was trained using the training data and assessed using the validation set. Data is then later reshaped.

To optimize the data processing for LSTM models, it must be restructured into a format that is compatible with their design. This conversion makes the task of time series data analysis easy, extracts key characteristics that point to potential patterns, and ultimately enables the model to produce precise forecasts about the performance of Tesla's stock.

4.4 Hyperparameters

Epoch, Batch size, No of neurons, Learning rate, and Dropout rate are the hyperparameters used by the LSTM model which are optimized in the later stage. Table 2 contains a set of values for all hyperparameters optimized in this experiment.

Hyperparameter	Range Values			
Epoch	100	125	150	200
Batch size	32	64	128	
No of neurons	128	256	512	
Learning rate	0.001	0.002	0.01	
Drop out rate	0.01	0.02	0.5	

Table 2 List of optimized hyperparameters

4.5 Evaluation

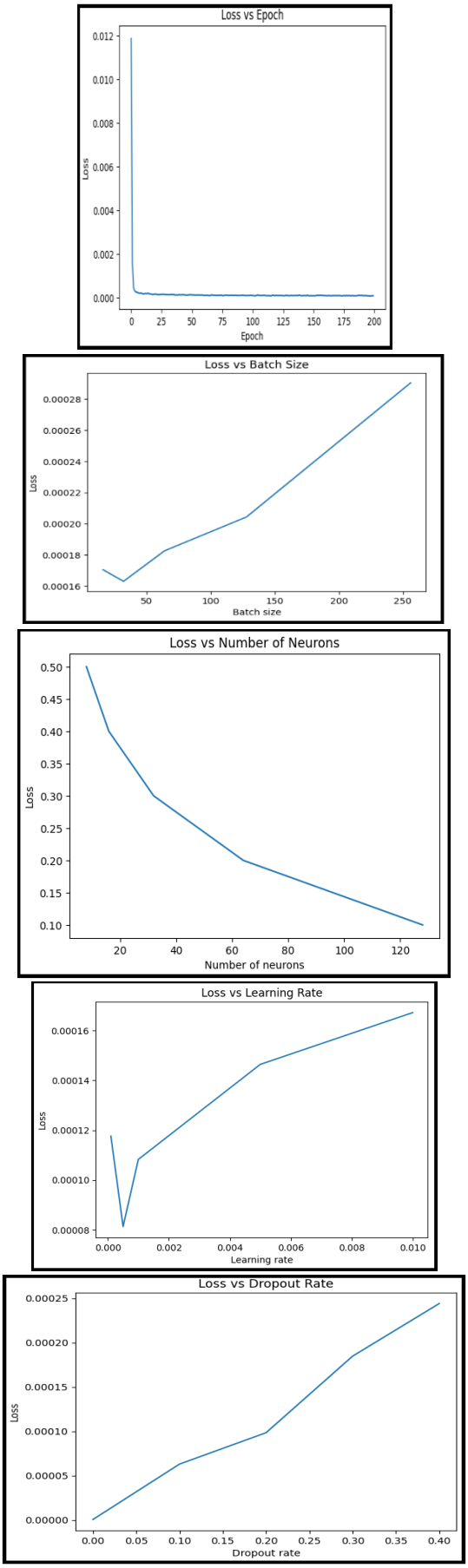
Error metrics typically range from 0 to positive infinity, with lower values representing better performance. However, R^2 scores range from negative infinity to 1, where closer to 1 signifies a better fit. An R^2 of 1 signifies perfect forecast, 0 implies no better than a constant model, and negative values indicate worse than constant.

4.6 Output comparison of hyper-tuned LSTM model with Adaboost (Stacked ensemble model)

First, a hyper-tuned LSTM model is created by choosing ideal hyper-parameters. Values of the evaluation metrics of the Adaboost model and the hyper-tuned LSTM model are compared. If the hyper-tuned LSTM model's performance metrics are almost identical to those of the Adaboost model, then our model's hyperparameter combination is appropriate. This comparison ensures that the hyper-tuned LSTM model will predict with precision.

5. Result Analysis and discussion

Plotting the relationship between loss v/s epoch, loss v/s batch size, loss v/s number of neurons, loss v/s learning rate, and loss v/s dropout rate yields the hyper-parameter range values. Loss is lowest when batch size is less than 50, loss is lowering as the number of neurons increases, the loss is lowest when the learning rate is less than or equal to 001, and loss is lowest when the dropout value is 01. These plots show that loss is decreasing as the epoch increases.



Dia 5.1 Plot between Loss v/s Epoch, Batch Size, Number of Neurons, Learning Rate, Dropout Rate

The LSTM algorithm is run for a range of hyper-parameter

combinations. Performance metrics such as MSE, MAE, and R2 are generated for every iteration of the LSTM algorithm for different combinations of hyperparameters.

The range values of every conceivable combination of hyper-parameters are generated when the LSTM model is executed, yielding the best, average, and worst-case tables shown below. Each table contains five entries for the best, average, and worst cases.

Sno	epoch	batch size	neurons	learning rate	drop out rate	Validation r2	Validation MSE	Validation MAE	R2 Adaboost	MSE Adaboost	MAE Adaboost
1	100	64	512	0.002	0.01	0.999126032	3.71E-07	0.000496503	0.9975541	1.04E-06	0.000703145
2	200	32	256	0.01	0.5	0.988017246	3.91E-07	0.000425609	0.9687767	1.40E-05	0.003247811
3	200	32	512	0.01	0.02	0.998936304	4.57E-07	0.000523688	0.9980227	8.50E-07	0.000652646
4	200	64	256	0.001	0.02	0.998374513	7.27E-07	0.000658372	0.994085	2.65E-06	0.001291842
5	200	32	512	0.002	0.01	0.999465314	2.393E-07	0.00040469	0.9974794	1.128E-06	0.000774066

Table 5.1 Best case

Sno	epoch	batch size	neurons	learning rate	drop out rate	Validation r2	Validation MSE	Validation MAE	R2 Adaboost	MSE Adaboost	MAE Adaboost
1	100	32	128	0.001	0.02	0.958651382	1.89E-05	0.00410701	0.9973858	1.19E-06	0.000818212
2	100	64	128	0.01	0.02	0.946527999	1.09E-05	0.003500087	0.9981592	7.18E-07	0.000659639
3	100	32	128	0.01	0.5	0.946900484	2.03E-05	0.004131385	0.9978557	8.19E-07	0.000634448
4	100	32	256	0.002	0.01	0.954101723	1.62E-05	0.003730961	0.9973581	9.33E-07	0.000720849
5	200	32	128	0.01	0.02	0.956200243	4.10E-05	0.006200417	0.9980367	1.84E-06	0.001054375

Table 5.2 Average case

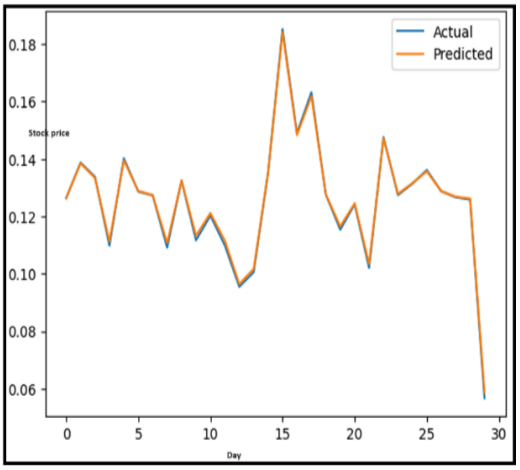
Sno	epoch	batch size	neurons	learning rate	drop out rate	Validation r2	Validation MSE	Validation MAE	R2 Adaboost	MSE Adaboost	MAE Adaboost
1	100	64	128	0.001	0.5	0.606368472	1.39E-04	0.011425277	0.9933893	2.34E-06	0.001155825
2	200	64	128	0.01	0.01	0.702764247	1.41E-04	0.01154768	0.997639	1.12E-06	0.00080448
3	200	64	128	0.01	0.5	0.594360498	1.86E-04	0.012125694	0.9967916	1.47E-06	0.000919985
4	100	64	128	0.01	0.5	0.439886751	2.51E-04	0.014640863	0.968263	1.42E-05	0.003011613
5	100	32	128	0.001	0.5	0.856145646	6.38E-05	0.007595311	0.9970705	1.30E-06	0.000892613

Table 5.3 Worst-case

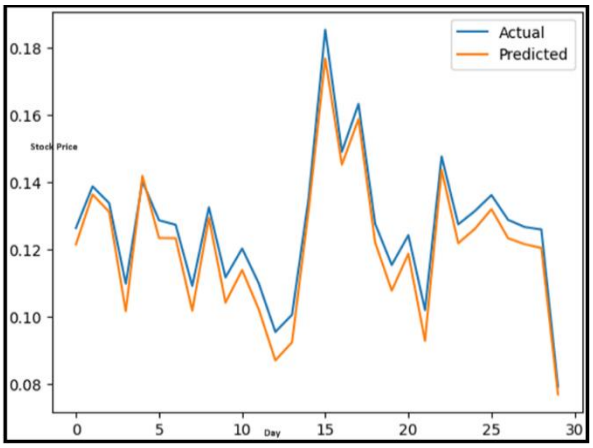
The validation R2 value determines the best, average, and worst case. Values closer to 1 fall into the best case, values greater than .5 and less than 1 fall into the average case, and values lower than .5 fall into the worst case. Lower values of validation MSE and MAE indicate a lesser loss function.

Finally, validation R2, MSE, and MAE values are compared with Adaboost (benchmark model). Adaboost model has already verified measurement metrics values. Suppose the value of measurement metrics in both models is nearly equal. In that case, our selected combination of hyperparameters in the hyper-tuned LSTM model is correct and up to the mark. This is a verification of our refined model.

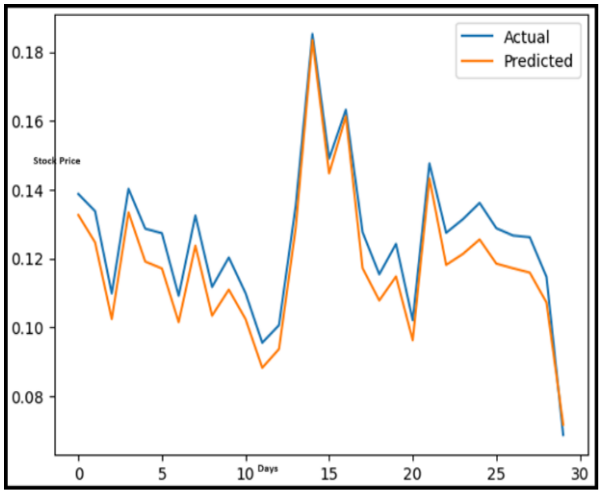
The plot between actual v/s predicted stock values for the best, average, and worst-case combination of hyperparameters of the hyper-tuned LSTM model also proves the accuracy and effectiveness of our refined model.



Dia 5.2 Best Case



Dia 5.3 Average Case



Dia 5.4 Worst case

6. Future Scope

The main contribution of this paper is to find an optimal pair of hyper-parameters of the LSTM algorithm that can give optimal output. Till now we have tuned the LSTM algorithm. We can do further refinement of our hyper-tuned LSTM algorithm by adding the impact of fundamental and technical indicators. Later we can also show sentiment analysis's impact on the hyper-tuned LSTM algorithm.

7. Conflicts of interest

The authors declare no conflicts of interest.

References

[1] D. S. N. Ulum and A. S. Girsang, “Hyperparameter Optimization of Long-Short Term Memory using Symbiotic Organism Search for Stock Prediction,” International Journal of Innovative Research and Scientific Studies, vol. 5, no. 2, pp. 121–133, 2022, doi: 10.53894/ijirss.v5i2.415.

[2] V. Deswal and D. Kumar, “Stock price prediction of AAPL stock by using machine learning techniques: A comparative study”, 2023 12th International Conference on System Modeling & Advancement in Research Trends (SMART),Moradabad, India, pp.189-197.

[3] K. E. Hoque and H. Aljamaan, “Impact of hyperparameter tuning on machine learning models in stock price forecasting,” IEEE Access, vol. 9, pp. 163815–163830, 2021, doi: 10.1109/ACCESS.2021.3134138.

[4] N. Reimers and I. Gurevych, "Optimal hyperparameters for deep lstm-networks for sequence labeling tasks," arXiv preprint arXiv:1707.06799, vol. 1, pp. 1 - 34, 2017.Available at: https://doi.org/10.48550/arXiv.1707.06799.

[5] M. Hermans and B. Schrauwen, "Training and analyzing deep recurrent neural networks," in Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013, pp. 190-198.

[6] Yadav, C. K. Jha, and A. Sharan, “Optimizing LSTM for time series prediction in Indian stock market,” in Procedia Computer Science, Elsevier B.V., 2020, pp. 2091–2100. doi: 10.1016/j.procs.2020.03.257.

[7] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and F. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," Journal of Machine Learning Research, vol. 18, no. 185, pp. 1-52, 2017.

[8] Hewamalage, H., Bergmeir, C., Bandara, K.: Recurrent neural networks for time series forecasting: Current status and future directions. International Journal of Forecasting 37(1), 388–427 (2021). doi:10.1016/j.ijforecast.2020.06.008.

[9] S. Girsang, F. Lioexander, and D. Tanjung, "Stock price prediction using LSTM and search economics optimization," IAENG International Journal of Computer Science, vol. 47, pp. 758-764, 2020.

[10] E. Ismanto, “LSTM Network Hyperparameter Optimization for Stock Price Prediction Using the Optuna Framework,” Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI), vol. 9, no. 1, pp. 22–35, 2023, doi: 10.26555/jiteki.v9i1.24944.

[11] U. K. Lilhore et al., “Hybrid CNN-LSTM model with efficient hyperparameter tuning for prediction of Parkinson’s disease,” Sci Rep, vol. 13, no. 1, Dec. 2023, doi: 10.1038/s41598-023-41314-y.

[12] Y. Li, Y. Zhang, and Y. Cai, “A new hyper-parameter optimization method for power load forecast based on recurrent neural networks,” Algorithms, vol. 14, no. 6, 2021, doi: 10.3390/a14060163.

[13] N. Gorgolis, I. Hatzilygeroudis, Z. Istenes, and L. N. G. Gyenne, “Hyperparameter Optimization of LSTM Network Models through Genetic Algorithm,” in 10th

International Conference on Information, Intelligence, Systems and Applications, IISA 2019, Institute of Electrical and Electronics Engineers Inc., Jul. 2019. doi: 10.1109/IISA.2019.8900675.

[14] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization."

[15] J. Bergstra, J. B. Ca, and Y. B. Ca, "Random Search for Hyper-Parameter Optimization Yoshua Bengio," 2012. [Online]. Available: http://scikit-learn.sourceforge.net.

[16] L. Yang and A. Shami, "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice," Jul. 2020, doi: 10.1016/j.neucom.2020.07.061.

[17] P. Probst, M. Wright, and A.-L. Boulesteix, "Hyperparameters and Tuning Strategies for Random Forest," Apr. 2018, doi: 10.1002/widm.1301.

[18] N. Bakhashwain and A. Sagheer, "Online Tuning of Hyperparameters in Deep LSTM for Time Series Applications," International Journal of Intelligent Engineering and Systems, vol. 14, no. 1, pp. 212–220, 2020, doi: 10.22266/IJIES2021.0228.21.

[19] T. Makarovskikh, M. Abotaleb, Z. Albadran, and A. J. Ramadhan, "Hyper-parameter tuning for the long short-term memory algorithm," in 4TH INTERNATIONAL SCIENTIFIC CONFERENCE OF ALKAFEEL UNIVERSITY (ISCKU 2022), AIP Publishing, Dec. 2023, p. 020097. doi: 10.1063/5.0181833.

[20] D. S. N. Ulum and A. S. Girsang, "Hyperparameter Optimization of Long-Short Term Memory using Symbiotic Organism Search for Stock Prediction," International Journal of Innovative Research and Scientific Studies, vol. 5, no. 2, pp. 121–133, 2022, doi: 10.53894/ijirss.v5i2.415.

[21] S. Girsang, F. Lioexander, and D. Tanjung, "Stock Price Prediction Using LSTM and Search Economics Optimization."

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[23] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1724-1734.

[24] Kim, Y.; Roh, J.H.; Kim, H. Early Forecasting of Rice Blast Disease Using Long Short-Term Memory Recurrent Neural Networks. Sustainability 2017, 10, 34.

[25] Y. Liu, Y. Zhou, S. Wen, and C. Tang, "A Strategy on Selecting Performance Metrics for Classifier Evaluation," Int. J. Mob. Comput. Multimed. Commun., vol. 6, no. 4, pp. 20–35, 2014, doi: 10.4018/IJMCMC.2014100102.

[26] A. Behera and A. Chinmay, "Stock Price Prediction using Machine Learning," Proc. - 2022 Int. Conf. Mach. Learn. Comput. Syst. Secur. MLCSS 2022, no. January, pp. 3–5, 2022, doi: 10.1109/MLCSS57186.2022.00