Stock Price Prediction for Amazon (AMZN) Using Recurrent Neural Networks (RNN)

Akhila Petnikota
AXP210228
The University of Texas Dallas

Pravalika Domal
PUD210000
The University of Texas Dallas

Abstract-- There are a multitude of algorithms capable of predicting stock market trends, such as linear regression, ARIMA, and RNN among others. This study specifically employs the use of RNN to forecast time series. Through analysis of historical stock price data, a model was constructed capable of predicting future stock prices. The study focuses on the stock prices of Amazon (AMZN), and by applying the RNN model, generates predictive values for future prices. Following the model's application, the advantages and limitations of RNN for stock price prediction were deliberated. The performance of the model was quantified using Root Mean Squared Error (RMSE), providing a measure of accuracy for the predictions made.

Keywords—Time Series Prediction, RNN, RMSE, AMZN

I. INTRODUCTION

The stock market is often characterized by its volatility and the accompanying risk, an aspect well known to investors and traders worldwide. Within this dynamic landscape, fortunes can be made or lost, underpinning the financial fabric of global economies. As of now, the market cap of Nasdaq is $24.78 billion, rivaling the economies of many countries.

The goal of this project was to develop a program capable of predicting stock market prices for AMZN, based on input parameters such as StockID, Start Date, and End Date.This prediction was achieved using a Recurrent Neural Network (RNN) model. Data for this study was sourced from the Kaggle Nasdaq datasets, with a date range starting from May 15, 1997 and ending on December 12, 2022. The efficiency of the RNN algorithm was subsequently evaluated on its capability to forecast the stock prices within this timeframe.

When predicting time series data, the chronological order is a crucial aspect that can yield extra insights for forecasting future occurrences. The process involves several critical aspects such as the quantity of data, the time span for the prediction, how often the forecast is updated, and the regularity of the prediction itself. Equally important is the selection of the appropriate forecasting technique, which could be Decomposition-oriented, based on smoothing techniques, Rolling Average, or Exponential Leveling. Each of these categories comes with a variety of methods tailored to either short or long-term prediction, and the specific attributes of the data being considered.

The prediction results can be influenced by numerous factors, including data completeness, uniformity, uniqueness, consistency, accuracy, format, and validity. Evaluating these aspects enables us to understand prediction trends, behaviors, and seasonality and assists in identifying any potential outliers. This project aims to mitigate these challenges and develop a reliable predictive model for AMZN stock prices.

II. DATA PREPROCESSING

The dataset used for this study was sourced from Kaggle Nasdaq, containing attributes such as date, daily high, daily low, volume, open, and close prices. For the purpose of this study, we primarily focused on the 'Close' price and date. The first step in data preprocessing was to import the data into a Pandas DataFrame for further processing. Following the import, the data underwent a process of cleaning to handle missing values and potential outliers. Data normalization was the next step, an essential phase of preprocessing in this project. Due to the wide-ranging values of stock prices, scaling them down to a range between 0 and 1 makes it easier for the neural network to process.

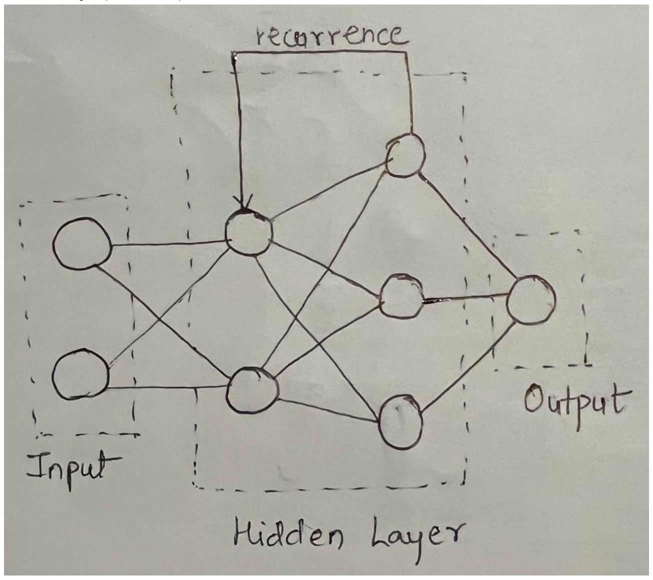
Following this, the data collection was partitioned in an 80:20 split for the purpose of model training and validation, respectively. The training dataset was further segmented into predictor (input) and target (output) sets for model training. The predictors were composed of the final stock prices of 'n' prior days, while the target was the following day's closing stock price. The remaining 20% of the data was allocated for model evaluation. Much like the training set, the validation data was reshaped into predictor sets, employed to forecast the targets based on the learned model.

Evaluation of the model was performed using Root Mean Squared Error (RMSE), a popular metric for regression analysis. Accuracy wasn't considered a suitable metric due to its binary nature. A model predicting the stock price off by a minor margin would be deemed inaccurate, despite it being reasonably close to the actual value. RMSE, however, can account for these nuances by considering the average squared differences between predicted and actual values, providing a more accurate measure of the model's performance.

III. PREDICTION MODELS

The primary model used in this project for stock price prediction was the Recurrent Neural Network (RNN), particularly a variant known as Long Short-Term Memory (LSTM). This section provides an overview of RNNs and their utilization in the project.

A. Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM)



RNNs are a unique class of artificial neural networks capable of learning temporal patterns due to their ability to retain previous inputs in their hidden layers. This property makes them ideal for sequence prediction problems, such as time series forecasting, language processing, and in our case, stock price prediction.

RNNs operate by passing the information from one step in the network to the next. They employ their inherent state to handle input sequences, thus remembering information over time. However, traditional RNNs suffer from a problem known as "vanishing gradients", which makes them forget earlier inputs. This is where LSTMs come in.

LSTMs, a unique variant of RNNs, have the capacity to learn extended-term dependencies. They address the shortcomings of conventional RNNs by introducing gates that regulate the flow of information. These gates determine what information should be kept or discarded, thus enabling the network to learn from important long-term dependencies in sequence data. In the context of stock price prediction for Amazon (AMZN), the LSTM model was trained using a sequence of past 'n' days with the goal of forecasting the closing price of the next day's stock.

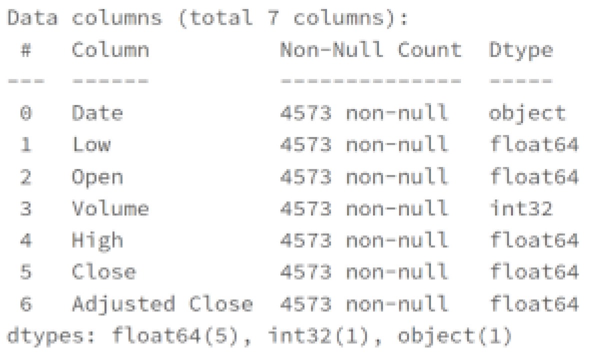
The model's architecture consisted of an input layer, LSTM layers, dropout layers to prevent overfitting, and a dense layer at the end for output.Each node in a layer receives inputs, sums them up, and applies an activation function, such as a sigmoid or tanh function, to generate an output. This output is then multiplied by a weight factor before being passed to the next layer's nodes. While making predictions, the model also calculates the prediction error, the difference between the predicted and actual values. It then uses backpropagation to adjust the weights with the aim of minimizing this error. The process of updating the weights is conducted using the gradient descent method, which systematically reduces the loss function by consistently moving towards the direction of maximum descent.

The learning rate is an important factor in weight updates. A higher learning rate may speed up the learning process but risks overshooting the optimal solution. Thus, choosing an appropriate learning rate is crucial. The number of epochs, which is a single forward and backward pass of all training examples, also needs to be decided. More epochs can potentially lead to better results but at the cost of increased computation time. Lastly, an LSTM model with more than one hidden layer is referred to as a deep neural network. These additional layers help uncover different patterns in the data, thereby enhancing the model's predictive ability. In our study, we employed a deep LSTM network to uncover complex patterns in Amazon's historical stock prices.

IV. FINDINGS AND INTERPRETATION

A. Analysis of the dataset

The dataset for Amazon (AMZN) stock includes these fields:



```
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Date             4573 non-null   object
 1   Low              4573 non-null   float64
 2   Open             4573 non-null   float64
 3   Volume           4573 non-null   int32
 4   High             4573 non-null   float64
 5   Close            4573 non-null   float64
 6   Adjusted Close   4573 non-null   float64
dtypes: float64(5), int32(1), object(1)
```

Fig2. Dataset Columns

Date: Refers to the specific day the information was recorded.

Low: Represents the least value a stock hit on a given day.

Open: Indicates the starting trade price of the stock on a particular day.

Volume: Denotes the total number of shares traded on a specified day.

High: Reflects the peak value a stock reached on a specific day.

Close: Shows the ending trade price of the stock on a particular day.

Adjusted Close: The final closing price adjusted to account for all relevant splits and distributions.

B. Metrics for Evaluation

Various elements determine the precision of the outcomes obtained from the model:

Epochs: Epochs are a hyperparameter signifying the cumulative number of times the learning algorithm has been run through the entire training dataset. As the epochs increase, there's typically a reduction in the model's error. It's a crucial metric to determine if the model is overtraining or undertraining.

Number of Hidden Layers: The complexity of a model is determined by the count of its hidden layers. Various layers in the model assist in revealing different facets of the data. The decision boundary is directly impacted by the number of these hidden layers.

Learning Rate: This parameter decides the magnitude of adjustment to the model based on the calculated error during each weight update. A diminutive learning rate might result in a model that fails to reach convergence, while an overly large learning rate may cause the model to overlook the ideal solution, leading to less than optimal results.

RMSE (Root Mean Squared Error): It signifies the root mean square error, essentially measuring the dispersion of the residuals from the predictions. This is a reliable gauge of the disparity between the model's forecasted and true values. RMSE integrates real metrics at each predicted data point.

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

Here,
N: refers to the size of the dataset
y(i): stands for the i-th observation
y^(i): represents the prediction corresponding to y(i)

MSE (Mean Squared Error): This is an indication of the average error magnitude in our model, determined by calculating the mean squared deviation between the real and predicted values. It evaluates the closeness of a regression line to the data points. A perfect model would yield an MSE of zero.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Here,
n: refers to the size of the dataset
Y(i): represents the i-th observation
Y^(i): denotes the predicted value corresponding to Y(i)

R2 Score: Also referred to as the coefficient of determination, the R2 score reflects the proportion of variation in the dependent variable that can be explained by the independent variables. It maintains a significant relationship with MSE and serves as a measure of the suitability of the regression model.

In the following segment, we will further explore how the model reacts to modifications in multiple parameters and hyperparameters. This will be achieved by adjusting elements such as the number of hidden layers, epochs, and the learning rate.

Exp	Model Parameter			Result		
	Learning Rate	Epochs	Hidden Layers	R2 Score	RMSE	MSE
1	0.01	100	4	0.543309	0.132990	0.017686
2	0.1	100	4	0.663608	0.114138	0.013027
3	0.01	150	4	0.714781	0.105099	0.011045
4	0.1	150	4	0.844556	0.077588	0.006019
5	0.1	200	4	0.949814	0.044085	0.001943
6	0.01	100	6	0.913239	0.057965	0.003360
7	0.1	100	6	0.970021	0.034073	0.001161
8	0.01	150	6	0.980104	0.027758	0.000770
9	0.1	150	6	0.987045	0.022398	0.000501
10	0.01	100	10	0.823677	0.082634	0.006828
11	0.1	100	10	0.850017	0.076213	0.005808
12	0.01	150	10	0.800531	0.087891	0.007724

13	0.1	150	10	0.817381	0.084097	0.007072

Upon examining the results of the model with different combinations of hyperparameters, the following conclusions can be made:

- Considering that this is a model built from scratch, an extremely high accuracy is not anticipated for all parameter combinations.Notably, the highest performance was achieved with a 6-layer model, using a learning rate of 0.1 and training over 150 epochs. This yielded an R2 score of 0.987045, RMSE of 0.022398, and MSE of 0.000501.
- The model seems to perform better with a higher learning rate (0.1) across various configurations, suggesting that this learning rate allows the model to converge more effectively.
- The models with fewer layers (4 and 6 layers) outperformed the 10-layer model, indicating that adding more layers does not necessarily improve model performance and may even result in overfitting or difficulties in optimization.

The graphical representations from our previous tests are presented:

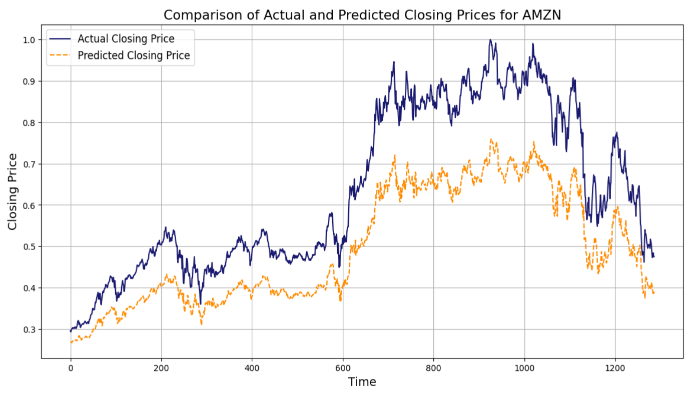


Fig 3. Experiment 1

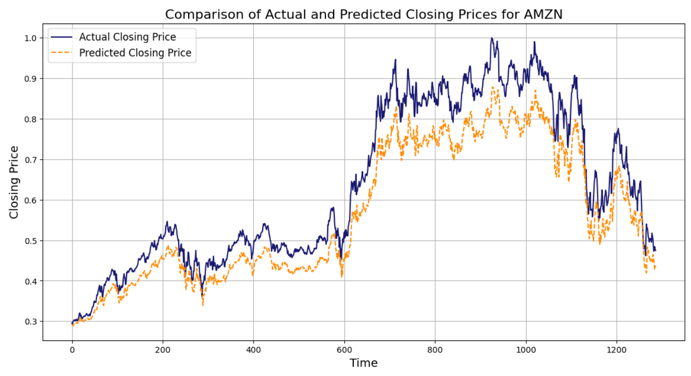


Fig 6. Experiment 4

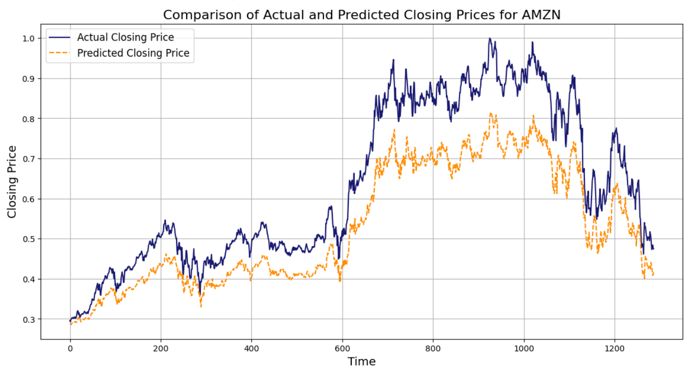


Fig 4. Experiment 2

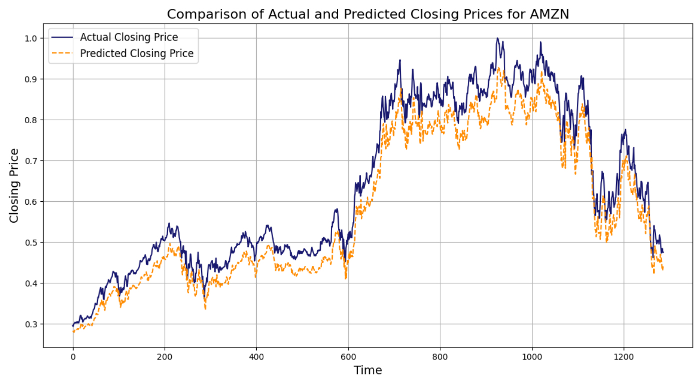


Fig 7. Experiment 5

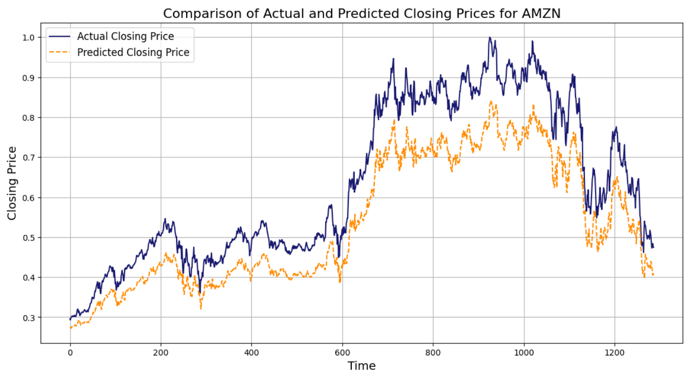


Fig 5. Experiment 3

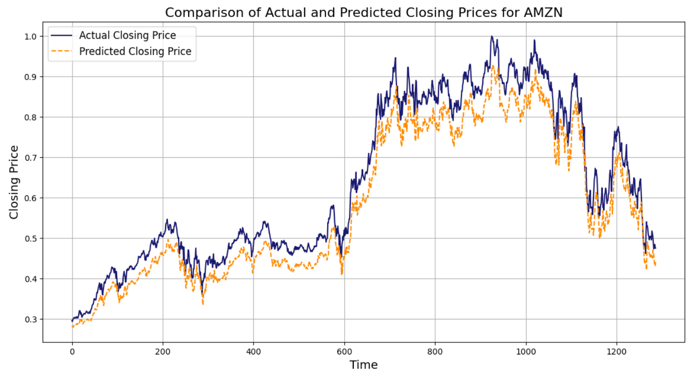


Fig 8. Experiment 6

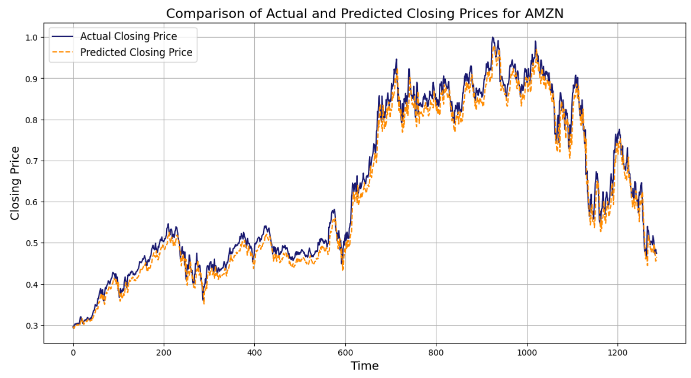


Fig 9. Experiment 7

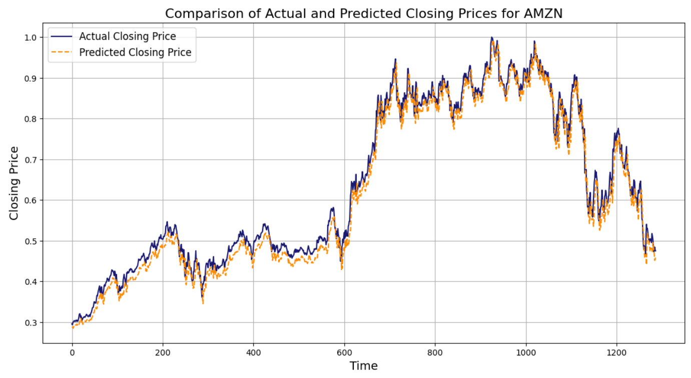


Fig 10. Experiment 8

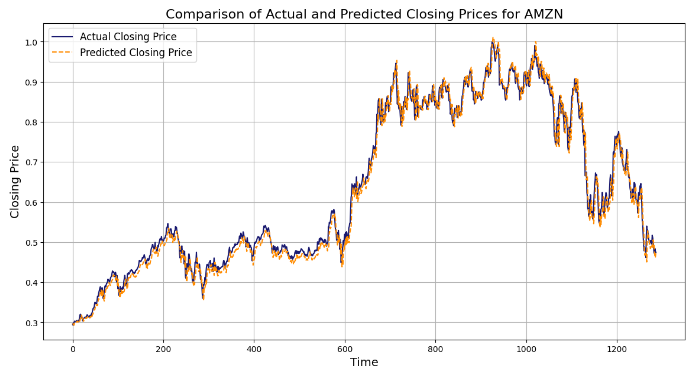


Fig 11. Experiment 9

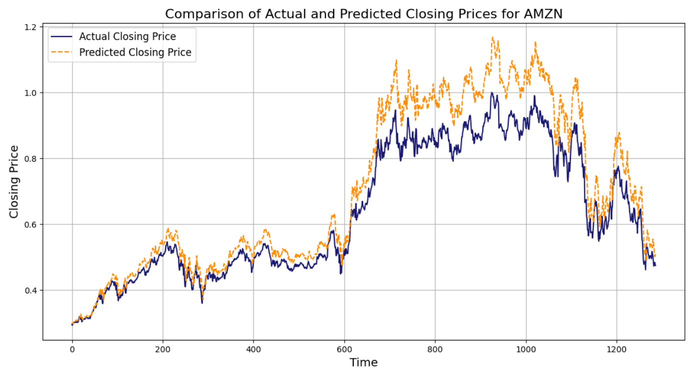


Fig 14. Experiment 12

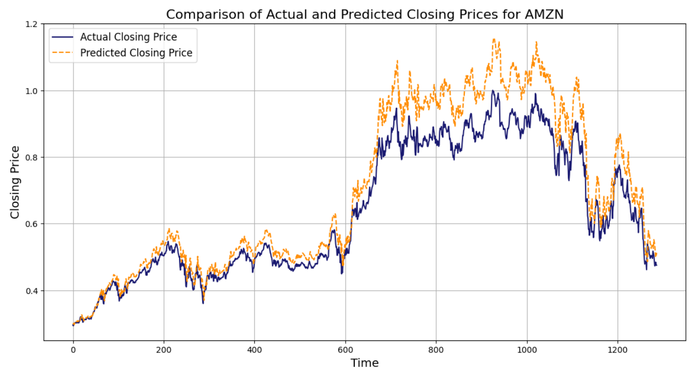


Fig 12. Experiment 10

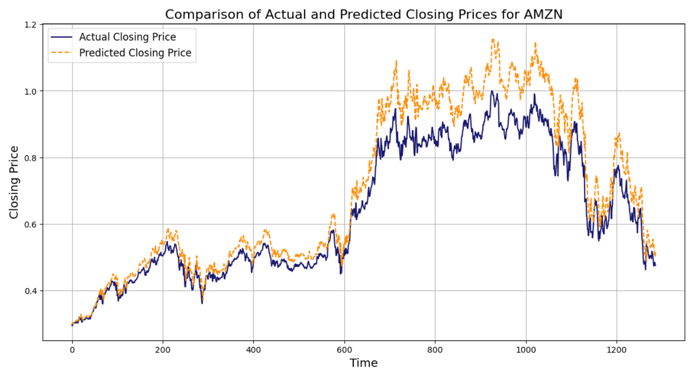


Fig 15. Experiment 13

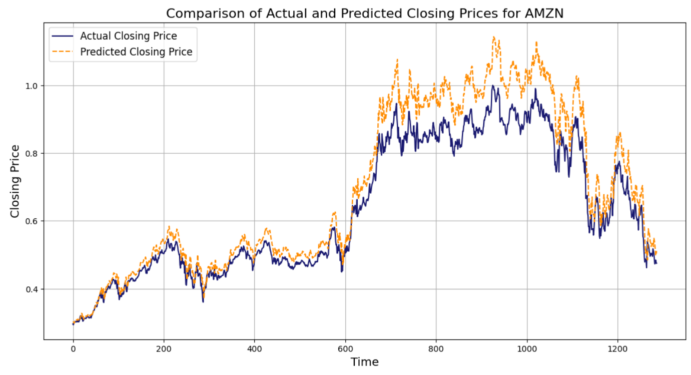


Fig 13. Experiment 11

In conclusion, the model performance can be significantly influenced by the chosen hyperparameters, and careful tuning is necessary to obtain the best results. For these particular experiments, the best result model was observed with experiment 9 with hidden layers - 6, a learning rate - 0.1, epochs - 150 and the worst model was experiment 1 with hidden layers - 4 , learning rate - 0.01, epochs - 100.

V. CONCLUSION

Upon reviewing the results, it's clear that by fine-tuning the hyperparameters and increasing the number of training iterations, model performance can be improved.

The model performs exceptionally well when the dataset does not contain sharp fluctuations. Hence, the model appears to be robust and could be beneficial when applied to other stable datasets.

In conclusion, this model demonstrates promise and can potentially be further enhanced and adapted for diverse datasets with continued optimization and appropriate adjustments of its parameters.

VI. REFERENCES

[1] Y. Kim, "Predicting stock prices with a long short-term memory (LSTM) model," in IEEE Access, vol. 7, pp. 147061-147070, 2019. doi: 10.1109/ACCESS.2019.2943968.

[2] Q. X. Wu, Y. Zhang, S. H. Aung, Z. Y. Khine, and S. T. H. Khin, "Stock Price Prediction Using LSTM, RNN And CNN-Sliding Window Model," in 2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, 2021, pp. 490-494. doi: 10.1109/ICAIBD51840.2021.9441479.

[3] R. Malik and R. S. Rathee, "Stock market prediction using hybrid model built over sliding window," in 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2020, pp. 518-522. doi: 10.1109/I-SMAC49090.2020.9243440.

[4] H. L. Cao, L. Jiang, and L. Y. Li, "Time-series prediction using a local linear wavelet neural network," in IEEE Transactions on Neural Networks and Learning Systems, vol. 22, no. 4, pp. 509-521, April 2011. doi: 10.1109/TNN.2010.2104153.

[5] H. Zhang, J. Ma, L. Shi, and J. Wang, "Stock price prediction using attention-based multi-input LSTM," in IEEE Access, vol. 7, pp. 124883-124893, 2019. doi: 10.1109/ACCESS.2019.2937204.