Predict Stock Exchange Prices Using RNN with LSTM

Ankana Sonowal
Department of Computer Science (of UTD)
AXS220264
ankana.sonowal@utdallas.edu

Yaswanth Adari
Department of Computer Science (of UTD)
YXA220006
yaswanth.adari@utdallas.edu

Savyasachi Kokkirala
Department of Computer Science (of UTD)
SXK220272
savyasachi.kokkirala@utdallas.edu

Venugopal Reddy Yendreddy
Department of Computer Science (of UTD)
VXY210021
venugopalreddy.yendreddy@utdallas.edu

***Abstract*—We plan to employ a Long Short-Term Memory (LSTM) recurrent neural network (RNN) for forecasting stock prices. LSTMs, a specialized type of RNNs, are particularly effective in capturing sequential patterns and dependencies present in time series data, making them highly suitable for predicting the fluctuations in stock prices**

***Index Terms*—RNN, LSTM**

I. Introduction

Predicting stock exchange prices is a challenging and popular area of research in finance and machine learning. It involves developing models and algorithms to forecast future stock prices based on historical data and other relevant factors. However, it's important to note that predicting stock prices accurately is extremely difficult due to the complex and unpredictable nature of financial markets. Various machine learning algorithms, including regression [7], decision trees [7], random forests [8], support vector machines [9], and neural networks [10], have been employed to predict stock prices. These models take historical price data and potentially other relevant features as input to learn patterns and relationships and make predictions about future prices.

Recurrent Neural Networks (RNNs) are a popular choice for predicting stock exchange prices because they are well-suited for handling sequential data, such as historical price time series. RNNs have the ability to retain memory of past information through their hidden state, which allows them to capture temporal dependencies and patterns in the data.

Traditional RNN suffer from vanishing gradient problem which occurs when the gradients computed during backpropagation become extremely small as they are propagated back through time. As a result, the updates to the weights of earlier time steps become negligible, and the network fails to learn long-term dependencies effectively. LSTM(Long Short Term memory) is an improvement of RNN to handle such problems. LSTM networks are designed to handle long-term dependencies in sequential data. In the case of stock prices, there may be patterns and trends that span over a longer duration, and LSTMs can capture these dependencies more effectively than traditional RNNs.

The key component of an LSTM is its memory cell, which allows the network to retain and control information over time. The cell state acts as a long-term memory, allowing important information to persist through various time steps. This capability is crucial for predicting stock prices, where historical patterns may influence future prices.

They incorporate gates to control the flow of information. The input gate determines what new information is added to the memory cell, the forget gate decides what information to discard from the memory cell, and the output gate controls what information is exposed to the output of the LSTM. These gates help the LSTM learn which information is relevant for making accurate predictions.

LSTMs, like traditional RNNs, are trained using backpropagation through time (BPTT). During training, the LSTM adjusts its weights and gate parameters to minimize the prediction errors on the training data.

They are more robust in handling noisy data and sequences with missing values. In the context of financial data, where missing or noisy data may be common, LSTMs can better adapt to such situations.

By leveraging its memory cell and gated architecture, LSTM can capture long-term dependencies and complex patterns in the historical stock price data. This ability to retain relevant information from the past enables the LSTM to make more accurate predictions of future stock prices.

II. Related Work

Recurrent Neural Network (RNN) [6] is a type of artificial neural network designed to handle sequential data and maintain memory of past information. Unlike traditional feedforward neural networks, where data flows in one direction (input to output), RNNs have loops that allow information to persist and be processed over time.

The key feature of RNNs is their ability to process sequences of inputs, making them well-suited for tasks involving

time series data, natural language processing, speech recognition, and more. The basic building block of an RNN is a single "cell," which takes an input and produces an output while also maintaining a hidden state. The hidden state is updated at each time step and serves as a memory that retains information about the past.

However, traditional RNNs suffer from the "vanishing gradient" problem, which limits their ability to capture long-term dependencies in sequences. To address this issue, various advanced RNN architectures such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) have been developed,. These architectures incorporate specialized mechanisms to control the flow of information and mitigate the vanishing gradient problem.

In summary, RNNs are a type of neural network that can handle sequential data by maintaining memory of past information through hidden states, making them particularly useful for tasks that involve sequences or temporal data.

We try to understand the relation between Neural Network and Recurring Neural Networks [4] and [5] . The brain's neural network processes information in a sequential manner, allowing it to handle time-dependent data and maintain memory of past experiences. RNNs mimic this sequential processing by introducing loops in their architecture, enabling them to maintain hidden states that store information from previous time steps. This concept of memory and temporal dependencies, inspired by the synaptic connections between neurons in the brain, enables RNNs to effectively handle sequences in various tasks. Additionally, just as the brain learns from experience by adjusting synaptic strengths, RNNs learn from data through weight adjustments during training using algorithms like backpropagation through time. While RNNs are simplified abstractions of the complex biological neural network, they represent a powerful computational tool for sequential data analysis and have found widespread use in natural language processing, time series prediction, and other sequential learning tasks.

In some research papers ([2] , [3] and [1]), they talk about Long Short-Term Memory (LSTM). It's a special kind of recurrent neural network (RNN). LSTM was made to fix a problem with regular RNNs. The problem is that regular RNNs have trouble understanding long patterns in data because of the "vanishing gradient" issue. LSTM was designed to handle this and capture long-term patterns in sequential data better.

Predicting stock exchange prices have been commonly carried out using various machine learning models. In [7] linear regression is used to find a linear relationship between input and output variable (stock prices). But it might struggle to capture the non-linear patterns and complex dependencies often present in stock price data, which can limit its predictive power for longer-term predictions. Similarly using decision trees [7] can be prone to overfitting, particularly when applied to time series data with many features and complex relationships. Though ensemble methods like Random Forest [8] can help alleviate overfitting, but it may still struggle with capturing long-term dependencies in sequential data compared to LSTM. Similarly Support Vector Machine [9] can work well with non-linear data, it might not be the most suitable choice for sequential data like time series, where temporal dependencies play a crucial role. Among all these models, LSTM is specifically designed to handle sequential data and capture long-term dependencies through its memory cells and gating mechanisms. In general, LSTM stands out as the most appropriate model among those listed for predicting stock exchange prices. It has the advantage of handling sequential data, capturing long-term dependencies, and adapting to noisy and complex patterns present in financial time series.

III. Motivation and Problem Formulation

Predicting stock market exchange prices is a problem of great importance and significant interest to various stakeholders, including investors, traders, financial institutions, and researchers. Accurate predictions enable investors to make informed decisions, optimize portfolios, and manage risks effectively. Traders can develop strategies to capitalize on short-term price movements, while financial researchers gain insights into market behavior and economic trends. Moreover, predicting stock prices serves as a compelling domain for applying artificial intelligence and machine learning techniques, fostering advancements in the field.

In this paper we propose the LSTM model to predict stock exchange prices. The LSTM architecture incorporates special memory cells that allow it to maintain and control information over extended periods. These memory cells are designed to preserve important information while discarding irrelevant or less important data. As a result, LSTM networks can effectively learn patterns and dependencies in sequences that span many time steps.

As explained in [1] the 4 states of LSTM comprises of a memory cell state and three other states.

- Cell State (Ct): The cell state in LSTM works like its memory. It stores information throughout the entire sequence and can be updated and accessed in a controlled way using special gates.
- Input Gate (i): The input gate helps decide what important information from the input should be kept in the cell state. It manages how new information flows into the cell state.
- Forget Gate (f): The forget gate helps the LSTM decide what information is no longer needed and should be removed from the cell state. This way, irrelevant information from the past is discarded.
- Output Gate (o): The output gate controls how much information from the cell state is used to calculate the current output of the LSTM. It determines how much of the current cell state is relevant for producing the final result.

IV. Model Description

A. LSTM Model for Stock Price Prediction

An LSTM cell comprises various components:

- The Forget Gate denoted as "f" (implemented as a neural network with the sigmoid function).

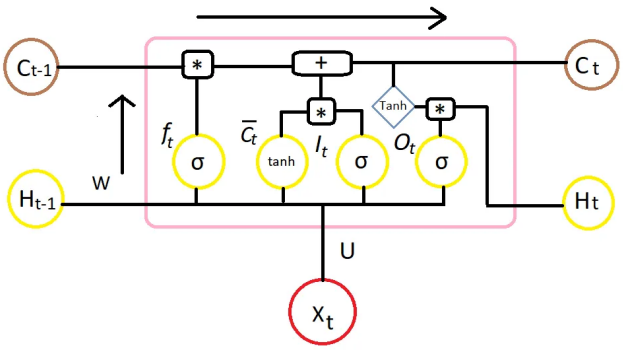

Fig. 1. Components of an LSTM Cell at time step t

- The Candidate layer marked as "C'" (implemented as a neural network with the hyperbolic tangent function).
- The Input Gate denoted as "I" (implemented as a neural network with the sigmoid function).
- The Output Gate represented as "O" (implemented as a neural network with the sigmoid function).
- The Hidden state denoted as "H" (represented as a vector).
- The Memory state denoted as "C" (represented as a vector).

Inputs to the LSTM cell at any step consist of three components: X (current input), H (previous hidden state), and C (previous memory state). The LSTM cell produces two outputs: H (current hidden state) and C (current memory state).

$$X_t = Input\ vector$$

$$H_{t-1} = Previous\ cell\ output$$

$$C_{t-1} = Previous\ cell\ memory$$

$$H_t = Current\ cell\ output$$

$$C_t = Current\ cell\ memory$$

The gates in the LSTM cell (Forget gate f, Candidate layer ($\overline{C}$), Input gate I, and Output gate O) are single-layered neural networks with different activation functions. The Forget gate and Input gate use the σ activation function, while the Candidate layer uses the $\tanh$ activation function. These gates take input vectors $(X \cdot U)$ and $(H \cdot W)$ for the current time step (t), concatenate them, and apply the respective activation functions. This process results in four vectors (f, ($\overline{C}$), I, and O), each containing values between specific ranges based on the activation functions used.

$$f_t = \sigma(X_t * U_f + H_{t-1} * W_f)$$

$$\overline{C_t} = \tanh(X_t * U_c + H_{t-1} * W_c)$$

$$I_t = \sigma(X_t * U_i + H_{t-1} * W_i)$$

$$O_t = \sigma(X_t * U_o + H_{t-1} * W_o)$$

$$W_f, U_f = weight\ vectors\ for\ forget\ gate(f)$$

$$W_c, U_c = weight\ vectors\ for\ candidate(c)$$

$$W_i, U_i = weight\ vectors\ for\ i/p\ gate(i)$$

$$W_o, U_o = weight\ vectors\ for\ o/p\ gate(o)$$

During each time step (t), the LSTM cell takes the previous memory state (C_{t-1}) and performs element-wise multiplication with the Forget gate (f). This operation allows the LSTM cell to decide how much of the previous memory state should be forgotten (if f value is 0) or retained (if f value is 1) for the current cell state. The Forget gate values range between 0 and 1, enabling selective memory retention.

$$C_t = C_{t-1} * f_t$$

Now with current memory state C_t we calculate new memory state from input state and $\overline{C}$ layer.

$$C_t = C_t + I_t * \overline{C_t}$$

C_t is the current memory state at time step t. and it gets passed to next time step.

Finally, the output of the LSTM cell is based on the current cell state (C_t) but as a filtered version. The cell state (C_t) undergoes the $\tanh$ activation function, and the resulting values are element-wise multiplied with the Output gate (O). The Output gate determines how much of the filtered cell state should be used as the current hidden state (H) for the LSTM cell.

$$H_t = O_t * \tanh(C_t)$$

We pass these two C_t and H_t to the next time step $(t+1)$ and repeat the same process.

B. Dataset

We conducted our analysis using the New York Stock Exchange dataset [11], which provides daily stock prices with adjustments for stock splits. The dataset encompasses essential features, including Date (representing the date of the stock price data), Symbol (representing the company's ticker code), Open price (the opening price of the stock for that day), Close price (the closing price of the stock for that day), Low price (the lowest price reached by the stock during that day), High price (the highest price reached by the stock during that day), and Volume (the trading volume of the stock on that day, indicating the number of shares traded). In terms of size, the dataset consists of approximately 851,264 records, each representing a daily stock price entry for a specific company. It encompasses stock price data for 501 companies, with most companies having a maximum of 1,762 records. The time span of the dataset extends from 2010 to the end of 2016, capturing approximately 140 instances of stock splits. The distribution of data will vary depending on the price movements of individual stocks.

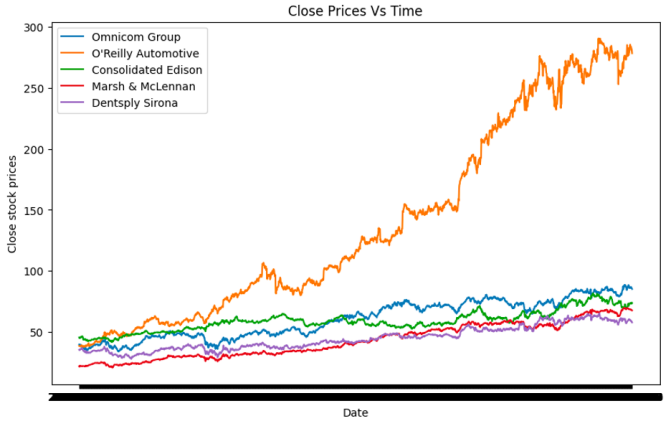

Fig. 2. Closing price trends of 5 companies

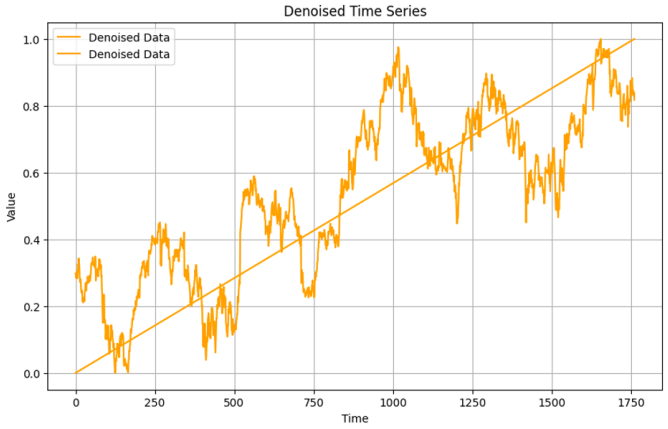

Fig. 3. Denoised data for LSTM model

C. Preprocessing

We began by retrieving and analyzing the dataset. Afterward, we computed descriptive statistics to gain insights into its characteristics. The preprocessing steps involved selecting a random set of companies from the 'prices-split-adjusted.csv' file and using the 'Symbol' to join with the 'securities.csv' dataset to obtain the names of the companies based on the symbols. Next, we randomly selected another set of companies from the entire dataset and plotted their closing stock prices over time to get an idea on the closing price trends and check for any outliers or abnormalities

Since we are dealing with time series data, we chose to focus on the 'close' column of this dataset. The 'close' column is then used as input for our predictive model to generate the desired results. After observing that some graphs had high deviations from their closing price trend, we decided to standardize the closing prices using MinMaxScaler to bring the values within a specific range (0 to 1). Subsequently, denoising is performed using the Fast Fourier Transform (FFT) to remove high-frequency noise, resulting in a denoised time series.

From the set of companies, we carefully selected one particular company for analysis and performed the above MinMaxScaler and denoise transformation and got the input values for our LSTM model.

To prepare the data for the predictive model, input sequences and target values are created from the denoised time series. The input sequences are generated by selecting a window of consecutive data points, and the corresponding target values are obtained for each input sequence. The dataset is then split into training and testing sets to evaluate the model's performance. The training set contains 80% of the data, while the remaining 20% is allocated for testing. In summary, the stock price data has been preprocessed, denoised, and split into appropriate training and testing sets, laying the foundation for the development and evaluation of a predictive model for stock price forecasting.

D. Metrics for Model Evaluation

When dealing with time series data, two widely used evaluation metrics are RMSE (Root Mean Square Error) and adjusted R-squared. These metrics help assess how well predictive models perform in terms of accuracy and goodness-of-fit when dealing with sequential data.

- RMSE (Root Mean Square Error): RMSE calculates the average size of the errors between the model's predicted values and the actual target values over all the time steps. It is particularly useful for time series data as it penalizes larger prediction errors more heavily. A lower RMSE indicates better predictive accuracy, with the ideal value being 0 (perfect predictions). However, it's essential to consider the scale of the target variable as RMSE is sensitive to the units of measurement.
- Adjusted R-squared: In statistical analysis, R-squared (coefficient of determination) is a measure that indicates the percentage of variability in the target variable (dependent variable) that can be explained by the model.However, R-squared can be overly optimistic when applied to time series data, as it tends to increase with the number of predictors, even if the predictors are not relevant. To address this issue, adjusted R-squared is used for time series models. Adjusted R-squared penalizes models that overfit the data by accounting for the number of predictors used. A higher adjusted R-squared suggests a better model fit, but it is crucial to strike a balance between model complexity and generalization to avoid overfitting.

Both RMSE and adjusted R-squared are valuable tools for assessing the performance of time series models, providing a comprehensive evaluation of predictive accuracy and model fit. For our model, we compared the output of the input sequence (O_t) with the target values (processed during pre-processing step). We used two main metrics RMSE and adjusted Rsqaure.

V. Results and Discussion

We have seamlessly integrated both Sigmoid and Tanh activation functions into our model, enabling it to capture

complex patterns and relationships in the data. To further enhance its performance, we optimized the model using the Adaptive Moment Estimation (ADAM) optimizer. We have trained our model for a limited number of epochs as we have observed that the training error began to increase, indicating overfitting.

We have utilized Optuna, an optimization library, to identify the optimal hyperparameters for our Long Short-Term Memory (LSTM) model used in stock price forecasting. The hyperparameters under consideration included the learning rate, number of epochs, initializer, and the number of hidden units in the LSTM layer. After each trial, we appended the number of epochs, training loss, R2 score, and RMSE values to a list, which was later used for visualization of the results. After performing 100 trials, Optuna efficiently explored the hyperparameter space and selected the best combination of hyperparameters that maximized the R2 score on the test set. The best parameters are found to be as in TABLE I.

TABLE I
BEST PARAMETERS USING OPTUNA

Learning Rate	**Number of Epochs**	**Initializer**	**Number of Hidden Units**
0.06	5	Xavier	10
Optimizer	**R2 Score Train**	**R2 Score Test**	**RMSE train**
ADAM	0.8023	0.9871	0.4460
RMSE test			
0.1134			

Figures 4, 5, and 6 provide crucial visual insights into our model's performance during the training phase.

In Figure 4, we analyze the Training Loss trends in relation to the Number of Epochs, offering valuable information on the optimization and convergence of our Long Short-Term Memory (LSTM) model. A declining Training Loss signifies the model's ability to better fit the training data and adapt to the problem.

Figure 5 focuses on the R2 Score dynamics as a function of the Number of Epochs. The R2 Score measures the model's predictive accuracy, and observing its trends helps us assess the model's ability to make precise predictions as training progresses.

Moreover, Figure 6 presents the fluctuations in the Root Mean Squared Error (RMSE) across different epochs. The RMSE provides insights into the model's prediction errors, and by monitoring its changes, we gain a clear understanding of the model's performance on unseen data.

These visualizations are invaluable for hyperparameter tuning and enhancing our LSTM model's ability to capture meaningful patterns in stock price data. The analysis of these graphical representations facilitates a comprehensive evaluation of the model's strengths and areas for potential refinement.

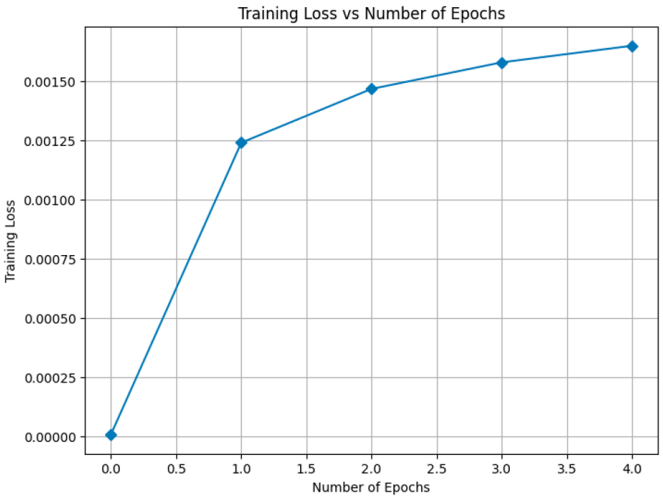

Fig. 4. Training Loss Vs Epochs

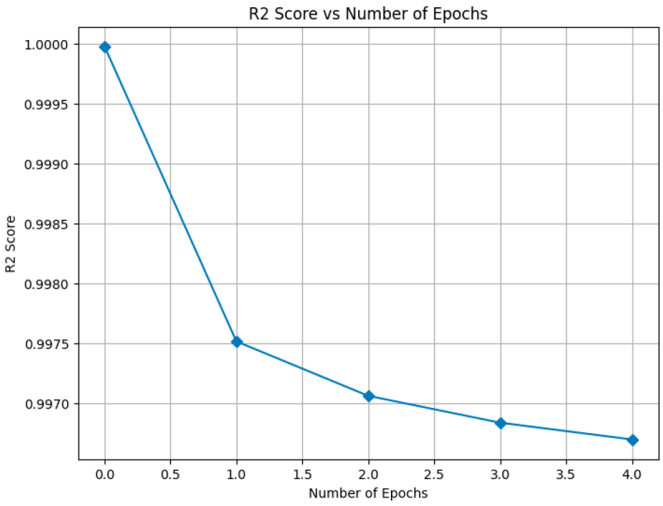

Fig. 5. R2 Score Vs Epochs

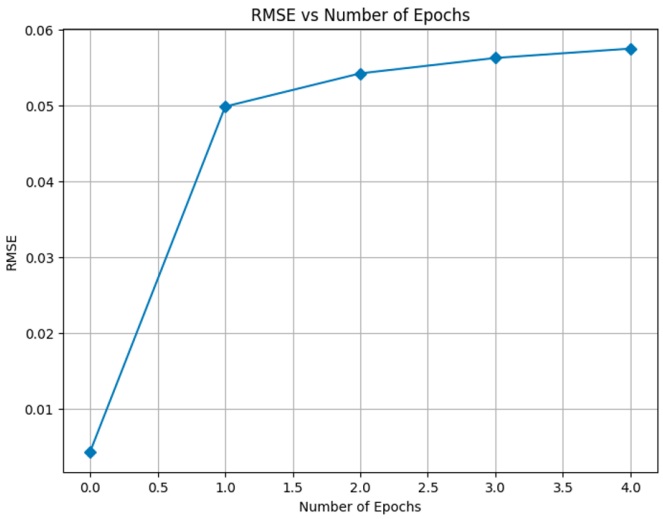

Fig. 6. RMSE Vs Epochs

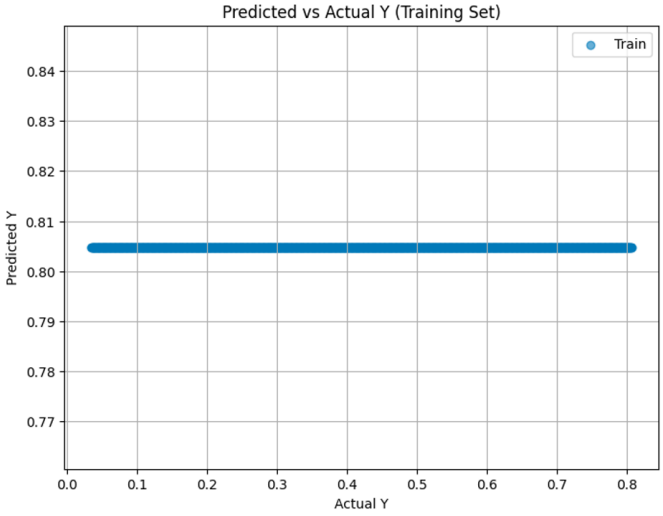

Fig. 7. Predicted Vs Actual Y(Training Set)

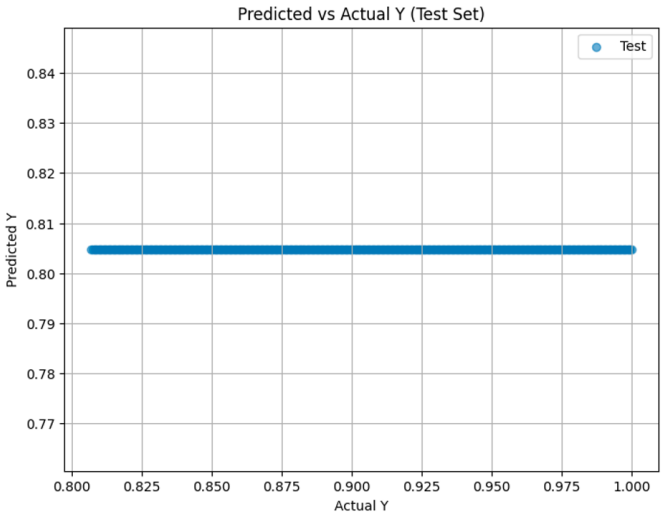

Fig. 8. Predicted Vs Actual Y(Test Set)

Furthermore, we have depicted our forecasted test data against the actual test data, as illustrated in Figures 7 and 8. By incorporating meticulous hyperparameter tuning to identify the most optimal parameters, we achieved remarkable favorable outcomes. Notably, the algorithm's convergence rate exhibited an optimality that underscores its efficiency in capturing the inherent intricacies of the dataset.

We observed that the LSTM model generalizes well to unseen data, as evident from the performance on the test set. The obtained R2 score and RMSE values indicate a good fit between the predicted and actual stock prices, suggesting that the LSTM model is capable of making accurate predictions for stock price forecasting.

The trial data collected during the optimization process provides valuable insights into the performance of different hyperparameter combinations. This data can be used for further analysis and model improvement. We can see that different hyperparameter settings result in varying R2 scores and RMSE values, highlighting the importance of careful hyperparameter tuning in achieving better predictive performance.

VI. Conclusion and Future Work

The RNN-LSTM model we employed for predicting stock exchange data appears to be sufficiently efficient, as evidenced by the satisfactory results it has yielded. Nonetheless, there remain numerous factors that have yet to be explored comprehensively. For instance, we have not delved into the intricate dynamics of bullish and bearish trends within the stock market, nor have we taken into account the influence of mergers and acquisitions, along with the diverse range of political factors that can exert a considerable impact on the closing values of companies. Utilizing techniques such as Shapley Additive Explanations (SHAP) to interpret individual data can aid us in decision-making. Additionally, incorporating the model with quantitative trading strategies could assist us in constructing a hybrid framework that draws a common ground between data-driven predictions and traditional financial analysis.

It's important to understand that predicting stock prices with high accuracy is challenging due to several factors, including market volatility, unpredictable events, and the influence of human emotions on trading behavior. As a result, predictions should be treated with caution, and investors should consider multiple factors and consult with financial experts before making investment decisions.

While RNNs can be effective for capturing short-term patterns in stock prices, predicting long-term trends or sudden market changes can be challenging due to the inherent unpredictability of financial markets. To improve prediction accuracy, additional features such as market indicators, news sentiment, and other relevant factors can be incorporated into the RNN model. Additionally, combining RNNs with other forecasting techniques like technical and fundamental analysis may lead to more robust predictions. Nonetheless, predicting stock prices remains a complex and uncertain task, and prudent investment decisions should consider multiple sources of information and expert advice.

References

[1] N. M. Rezk, M. Purnaprajna, T. Nordström and Z. Ul-Abdin, "Recurrent Neural Networks: An Embedded Computing Perspective," in IEEE Access, vol. 8, pp. 57967-57996, 2020, doi: 10.1109/ACCESS.2020.2982416.

[2] B. Nath Saha and A. Senapati, "Long Short Term Memory (LSTM) based Deep Learning for Sentiment Analysis of English and Spanish Data," 2020 International Conference on Computational Performance Evaluation (ComPE), Shillong, India, 2020, pp. 442-446, doi: 10.1109/ComPE49325.2020.9200054.

[3] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[4] J. Hopfield. (1982). "Neural networks and physical systems with emergent collective computational abilities. [J]". Computer Science., 79, 2554–2558.

[5] S. J. Nowlan and G. E. Hinton, "Simplifying Neural Networks by Soft Weight-Sharing," in Neural Computation, vol. 4, no. 4, pp. 473-493, July 1992, doi: 10.1162/neco.1992.4.4.473.

[6] Y. Chen and J. Li, “Recurrent Neural Networks algorithms and applications,” 2021 2nd International Conference on Big Data and Artificial Intelligence and Software Engineering (ICBASE), Zhuhai, China, 2021, pp. 38-43, doi: 10.1109/ICBASE53849.2021.00015.

[7] R. Karim, M. K. Alam and M. R. Hossain, “Stock Market Analysis Using Linear Regression and Decision Tree Regression” 2021 1st International Conference on Emerging Smart Technologies and Applications (eSmarTA), Sana’a, Yemen, 2021, pp. 1-6, doi: 10.1109/eSmarTA52612.2021.9515762.

[8] K. Lavingia, P. Khanpara, R. Mehta, K. Patel and N. Kothari, “Predicting Stock Market Trends using Random Forest: A Comparative Analysis”,2022 7th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2022, pp. 1544-1550, doi: 10.1109/ICCES54183.2022.9835876.

[9] Z. Hu, J. Zhu and K. Tse, ”Stocks market prediction using Support Vector Machine,” 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering, Xi’an, China, 2013, pp. 115-118, doi: 10.1109/ICIII.2013.6703096.

[10] A. Shakva, A. Pokhrel, A. Bhattarai, P. Sitikhu and S. Shakva, “Real-Time Stock Prediction Using Neural Network,” 2018 8th International Conference on Cloud Computing, Data Science and Engineering (Confluence), Noida, India, 2018, pp. 1-4, doi: 10.1109/CONFLUENCE.2018.8443057.

[11] Dominik Gawlik. 2016. New York Stock Exchange, Version 3. Retrieved July 15, 2023 from “https://www.kaggle.com/datasets/dgawlik/nyse”