IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

UNDERSTANDING TRENDS AND PREDICTING THE STOCK MARKET

USING STACKED-LSTM MODEL

[1] Surya Ganesh, [2]Atharva Mhaske, [3] Rahul Shekade, [4] Aayush Rane, [5] Dr. Shudhodhan Bokefode

[1]Student, [2]Student, [3]Student, [4]Student, [5]Associate Professor

[1]Computer Engineering, [2]Computer Engineering, [3]Computer Engineering, [4]Computer Engineering, [5]Computer Engineering

Terna Engineering College, Nerul, India

Abstract: This research project explores the application of Long Short-Term Memory (LSTM) networks in predicting stock market movements and compares its performance against the Prophet model, a popular time series forecasting tool. The project delves into the theoretical foundations of LSTM networks, elucidating their architecture and working principles, with a focus on their ability to capture both short-term fluctuations and long-term dependencies within sequential data. Historical stock price data from Yahoo Finance are collected and pre-processed, ensuring data quality and compatibility with the LSTM and Prophet models. Through a comprehensive experimental evaluation, the LSTM model's predictive accuracy and performance metrics, such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE), are assessed and compared with those of the Prophet model. The results of the comparative analysis shed light on the strengths and limitations of each model, offering valuable insights into their efficacy and suitability for stock market prediction tasks. This project contributes to the ongoing discourse at the intersection of deep learning and finance, guiding practitioners and researchers seeking to leverage advanced machine learning techniques for forecasting financial markets.

Index Terms **– Stacked-LSTM model, forecasting, stock market.**

Introduction

In the dynamic landscape of artificial intelligence, neural networks have emerged as powerful computational models inspired by the interconnected structure of the human brain. Originating in the mid-20th century, neural networks underwent periods of both enthusiasm and dormancy due to limitations in computing power and theoretical understanding. However, with the advent of deep learning and advancements in hardware capabilities, neural networks experienced a resurgence, offering unparalleled potential in various domains, including pattern recognition, natural language processing, and financial analysis. Among the diverse architectures within the realm of neural networks, recurrent neural networks (RNNs) gained prominence for their ability to handle sequential data. RNNs introduced the concept of feedback loops, enabling them to retain the memory of past observations, making them particularly adept at processing time-series data. However, early RNNs faced challenges in retaining long-term dependencies, often suffering from the vanishing gradient problem, where gradients diminish exponentially over time, hindering learning. To address these limitations, Long Short-Term Memory (LSTM) networks were introduced by Hochreiter and Schmidhuber in 1997. Representing a significant advancement in the field of recurrent neural networks, LSTM networks incorporated specialized memory cells, gating mechanisms, and a unique architecture designed to mitigate the vanishing gradient problem. This innovative design allowed LSTMs to effectively capture both short-term and long-term dependencies within sequential data, making them exceptionally well-suited for tasks involving temporal dynamics. Since their inception, LSTM networks have revolutionized the landscape of deep learning, finding applications in diverse domains, including speech recognition, language translation, and notably, stock market

prediction. This paper delves into the evolutionary trajectory and theoretical foundations of LSTM networks, elucidates their methodology in stock market prediction, conducts a comparative analysis with traditional methods, and draws conclusions regarding their efficacy and implications for the future of financial markets. Through this exploration, we aim to contribute to the ongoing discourse at the intersection of deep learning and finance, offering insights and recommendations for further advancements in this dynamic field..

I. OVERVIEW OF DEEP LEARNING

RNNs:

Recurrent Neural Networks (RNNs) are a class of artificial neural networks specifically designed to handle sequential data, where the order of elements matters. Unlike traditional feedforward neural networks, RNNs possess connections that form directed cycles, allowing them to exhibit dynamic temporal behavior and capture dependencies across time steps.

The architecture of RNNs:

- RNNs consist of neurons organized into layers: an input layer, one or more hidden layers, and an output layer.
- Each neuron in an RNN has recurrent connections, enabling it to receive input not only from the current time step but also from previous time steps through the hidden state..

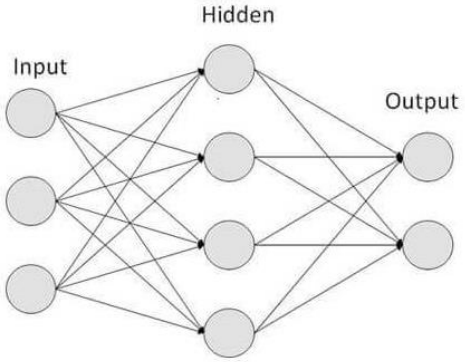


Figure 1: RNN Diagram

Hidden State:

- The hidden state of an RNN neuron serves as memory, retaining information from previous time steps.
- At each time step, the hidden state is updated based on the current input and the previous hidden state.

Recurrent Connections:

- Recurrent connections allow information to persist across time steps, enabling RNNs to model sequences and capture temporal dependencies.
- These connections create a feedback loop, where the output of a neuron at one time step becomes part of the input to the same neuron at the next time step.

Working of RNNs:

- During each time step, an RNN receives an input vector and the previous hidden state as input.
- The input vector is combined with the previous hidden state and multiplied by a weight matrix to produce the current hidden state.
- The current hidden state is then passed through an activation function to produce the output of the neuron, which can be used for prediction or passed to subsequent neurons in the network.
- This process is repeated for each time step in the sequence, allowing the network to capture dependencies and generate predictions based on sequential data.

Challenges and Limitations:

- RNNs are susceptible to the vanishing and exploding gradient problem, where gradients either diminish exponentially or grow uncontrollably during backpropagation.
- The vanishing gradient problem hinders the ability of RNNs to capture long-range dependencies effectively, limiting their performance on tasks involving long sequences.

In summary, Recurrent Neural Networks (RNNs) are powerful tools for processing sequential data, with applications in natural language processing, time series analysis, and more. By capturing dependencies between elements in a sequence, RNNs can model complex temporal dynamics and make predictions based on sequential data. However, they face challenges such as the vanishing gradient problem, which can impact their effectiveness on tasks requiring the modeling of long sequences.

LSTM:

Long Short-Term Memory (LSTM) networks represent a sophisticated architecture within the realm of recurrent neural networks (RNNs), specifically designed to overcome the challenges of capturing long-term dependencies in sequential data. The key components of an LSTM network include memory cells, input gates, forget gates, and output gates, each playing a crucial role in the network's ability to retain and utilize information over extended sequences.

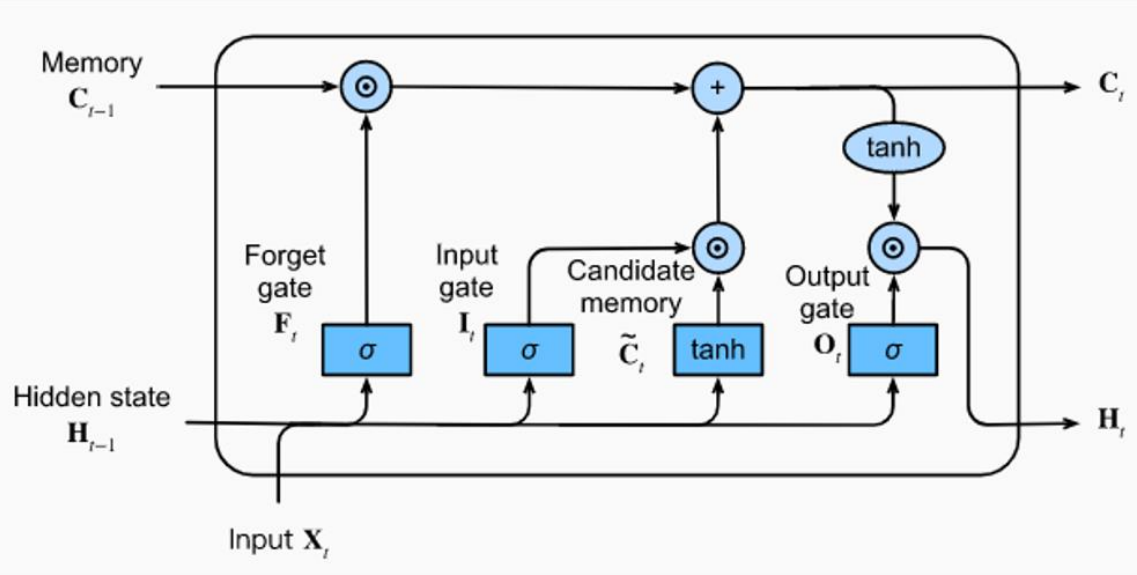


Figure 2: LSTM Diagram

1. Memory Cells:

- At the heart of LSTM networks are memory cells, which serve as containers for storing information over time. Each memory cell maintains an internal state, referred to as the cell state or long-term memory, which can be updated and modified based on incoming data and gate activations.

2. Gates:

- LSTM networks incorporate three types of gates to regulate the flow of information:
  - Input Gate: Responsible for determining which information from the current input should be stored in the cell state. It is controlled by a sigmoid activation function, allowing it to output values between 0 and 1, representing the importance of each input.

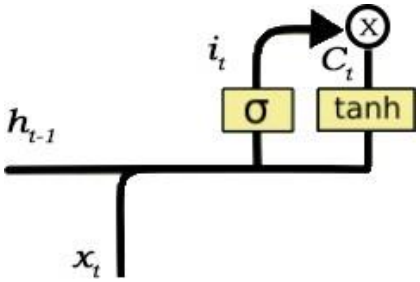


Figure 3: Input Gate

- Forget Gate: Determines which information from the previous cell state should be discarded or forgotten. It is also controlled by a sigmoid activation function, enabling the model to selectively retain or discard information from the long-term memory.

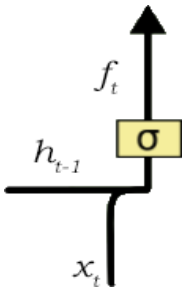


Figure 4: Forget Gate

- Output Gate: Dictates the information to be output from the current cell state. It regulates the flow of information from the cell state to the output and is controlled by a sigmoid activation function followed by a hyperbolic tangent function.

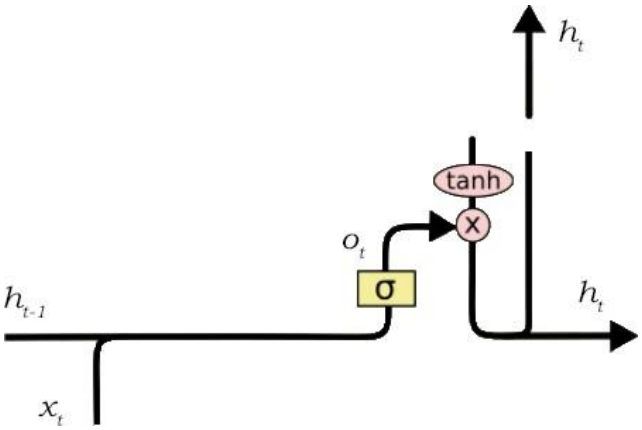


Figure 5: Output Gate

3. Cell State Update:

- The cell state is updated based on the inputs, previous cell state, and gate activations.
- The forget gate decides which information from the previous cell state to discard, while the input gate determines which new information to store in the cell state.
- The cell state is updated by adding the information from the input gate and forgetting the information determined by the forget gate.
- The output gate then regulates the flow of information from the updated cell state to the output of the LSTM unit.

4. Working of LSTM:

- During each time step, LSTM units receive input data and the previous cell state as inputs.
- The input gate, forget gate, and output gate activations are computed based on the input data and the previous cell state.
- The cell state is updated according to the gate activations, selectively storing or discarding information based on the relevance and importance of the input.
- The updated cell state is then passed through the output gate to produce the output of the LSTM unit, which can be used for prediction or passed to subsequent LSTM units in a multi-layer architecture.

Procedure:

1. Data Preprocessing:

- Before feeding data into the LSTM model, preprocessing steps are crucial to ensure data quality and compatibility.
- Historical stock price data, including features such as opening price, closing price, volume, and technical indicators, are collected from reliable sources.
- Data is normalized or scaled to improve convergence during training and prevent numerical instabilities.

2. Model Architecture:

- The LSTM model architecture typically consists of input, LSTM layers, and output layers.
- Input data is fed into the LSTM layers, where the network learns to capture temporal dependencies.
- Multiple LSTM layers may be stacked to capture more complex patterns in the data.

- Dropout layers may be incorporated to prevent overfitting by randomly dropping a fraction of connections during training.

3. Training Process:
- The LSTM model is trained using historical stock price data with a supervised learning approach.
- The model learns to predict future stock prices based on past price movements and other relevant features.
- Training involves minimizing a loss function, such as mean squared error (MSE), through backpropagation and gradient descent optimization.

4. Prediction and Evaluation:
- Once trained, the LSTM model can be used to make predictions on unseen data.
- Predictions are made for future stock prices, often over a specified time horizon.
- Evaluation metrics such as Root Mean Squared Error (RMSE) are used to assess the model's performance.
- The effectiveness of the LSTM model is compared against baseline models or other forecasting methods to determine its relative performance.

5. Factors Affecting Effectiveness:
- The effectiveness of LSTM in stock market prediction depends on various factors, including:
  - Quality and relevance of input features: Incorporating relevant features such as technical indicators, market sentiment, and macroeconomic factors can enhance prediction accuracy.
  - Training data quality and quantity: Larger and more diverse training datasets can improve the model's ability to learn complex patterns.
  - Model hyperparameters: Tuning hyperparameters such as learning rate, batch size, and network architecture can significantly impact model performance.
  - Market dynamics: Stock market behaviour is influenced by various factors, including geopolitical events, economic indicators, and investor sentiment, which can affect prediction accuracy.

6. Effectiveness and Challenges:
- LSTM models have demonstrated promising results in stock market prediction, outperforming traditional statistical models in many cases.
- However, predicting stock prices accurately remains a challenging task due to the inherent uncertainty and volatility of financial markets.
- LSTM models may struggle to generalize to unseen market conditions or abrupt changes in market dynamics.
- Continuous refinement and adaptation of LSTM models are necessary to improve prediction accuracy and adapt to evolving market conditions.

Stacked LSTM:
- Stacked LSTM networks consist of multiple LSTM layers stacked on top of each other, forming a deep architecture.
- Each LSTM layer processes the input sequence sequentially and passes its output to the subsequent layer.
- Stacked LSTM networks can capture hierarchical and abstract representations of sequential data by learning increasingly complex features across layers.
- By stacking multiple LSTM layers, the network can capture long-term dependencies and extract higher-level abstractions from the input sequence.

Advantages of Stacked LSTM over LSTM:
1. Hierarchical Representation Learning:
- Stacked LSTM networks can learn hierarchical representations of sequential data through multiple layers of abstraction.
- Each LSTM layer captures different levels of temporal dependencies and patterns, enabling the network to extract increasingly complex features.
2. Capturing Long-Term Dependencies:
- Stacked LSTM architectures are better equipped to capture long-term dependencies in sequential data.
- By stacking multiple LSTM layers, the network can maintain information over longer sequences, facilitating the modelling of extended temporal dependencies.

3. Improved Performance:
- Stacked LSTM networks often achieve higher predictive performance compared to single-layer LSTM models.
- The deeper architecture allows for more nuanced feature extraction and better representation learning, leading to improved accuracy and generalization on unseen data.

4. Flexibility and Adaptability:
- Stacked LSTM networks offer greater flexibility in modelling sequential data by allowing the incorporation of multiple LSTM layers with varying complexities.
- This flexibility enables adaptation to diverse application domains and datasets, providing a versatile framework for sequential data analysis.

5. Regularization and Overfitting Mitigation:
- The deep architecture of Stacked LSTM networks inherently provides a form of regularization, helping to mitigate overfitting.
- Techniques such as dropout regularization can be applied to individual LSTM layers within the stacked architecture, further enhancing generalization performance and preventing overfitting.

Advantages of Stacked LSTM Models:

While LSTM networks exhibit remarkable capabilities, the concept of "stacking" (Fig 1.3) these networks, known as Stacked LSTMs, enhances their performance even further. The primary advantages of employing Stacked LSTMs in stock price prediction are:

1. Improved Representation Learning: Stacking multiple LSTM layers allows the network to learn increasingly abstract and hierarchical representations of the input data. This enables the model to capture intricate patterns and dependencies within the stock price time series, leading to more accurate predictions.

2. Enhanced Model Capacity: Stacking LSTMs increases the model's capacity to handle complex data by leveraging multiple levels of abstraction. This additional capacity is instrumental in capturing subtle nuances in stock price behaviour.

3. Reduced Overfitting: The hierarchical structure of Stacked LSTMs provides a form of regularization, helping to prevent overfitting, a common issue in deep learning. This results in more robust and reliable predictions.

II. PROBLEM STATEMENT

The problem statement for this project is to develop and evaluate Long Short-Term Memory (LSTM) and Stacked LSTM models for predicting stock market movements. The objective is to leverage deep learning techniques to capture temporal dependencies and patterns in historical stock price data, thereby enabling accurate forecasting of future stock prices. The project aims to compare the effectiveness of single-layer LSTM and stacked LSTM architectures in modelling complex temporal dynamics and capturing long-term dependencies in sequential financial data. Additionally, the study seeks to assess the predictive performance of LSTM-based models against traditional forecasting methods, such as autoregressive models and technical indicators.

III. ALGORITHM

The Stacked Long Short-Term Memory (LSTM) algorithm is a powerful tool for capturing complex patterns and dependencies in sequential data, such as stock prices, time series, or text. In simplest terms, imagine you're trying to predict the weather tomorrow based on past weather data. You wouldn't just look at today's weather; you'd consider the weather over the past few days or weeks to understand patterns like temperature trends, seasonal changes, or sudden shifts. Stacked LSTM works similarly but with numbers and mathematical computations.

Understanding LSTMs:

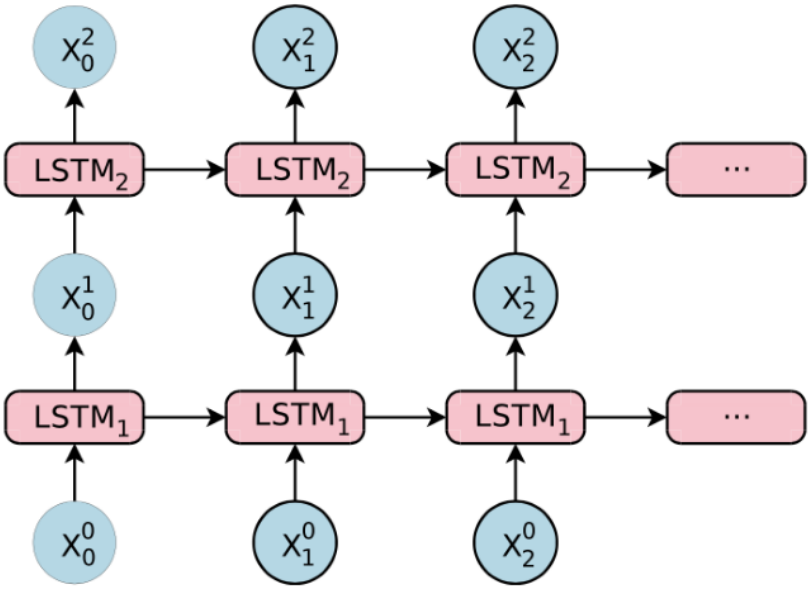
LSTMs are like smart cells in a brain that remember important information from the past and use it to make predictions. They're called "Long Short-Term Memory" because they can remember things from a long time ago (long-term memory) and forget things that are not important anymore (short-term memory).

Building Blocks:

Imagine stacking these smart cells on top of each other, forming layers like a tower. Each layer (or level) of cells can learn different things about the data. The bottom layer might focus on short-term patterns, like daily fluctuations, while the top layer looks at longer-term trends, like weekly or monthly cycles.

Passing Information:

When you feed data into the first layer of the Stacked LSTM tower, each smart cell processes the information it receives and passes it to the next cell in the same layer. This process continues within the layer, with each



cell contributing its understanding of the data.

Figure 6: Stacked LSTM Diagram

Learning Hierarchical Representations:

As the data moves up the tower through each layer, the cells in each layer learn more complex patterns and dependencies. It's like going from simple shapes to intricate designs as you climb higher in a building. Each layer builds upon the knowledge learned in the previous layer, creating a hierarchical representation of the data.

Making Predictions:

At the top of the tower, the last layer of cells combines all the learned information to make predictions about future data points. These predictions are based on the patterns and trends identified throughout the data hierarchy. Just like a weather forecaster who considers various factors to predict tomorrow's weather, the Stacked LSTM algorithm integrates multiple layers of information to forecast future outcomes.

Training and Learning:

To make accurate predictions, the Stacked LSTM algorithm goes through a training process where it learns from historical data. It adjusts the connections between cells (like strengthening or weakening connections in the brain) to improve its ability to capture patterns and make predictions.

Evaluating Performance:

After training, the algorithm is tested on new data to evaluate its performance. It compares its predictions with the actual outcomes to see how well it has learned to understand the data and make accurate forecasts.

IV. RESEARCH METHODOLOGY

5.1 Population and Sample

This project utilizes listed stocks from various global markets as its dataset. The developed model is adept at predicting stock prices for these listed stocks, providing valuable insights into future market trends and behaviors. By leveraging advanced techniques such as stacked Long Short-Term Memory (LSTM) networks, the model demonstrates the capability to analyze historical stock market data and make accurate predictions. This enables investors and analysts to make informed decisions, optimize investment strategies, and navigate the dynamic landscape of the global stock market with confidence.

5.2 Data and Sources of Data

For this study, secondary data has been collected. The data for this project is sourced by using APIs, specifically utilizing Yahoo Finance as the primary data source. Through Yahoo Finance's API, we obtain comprehensive and up-to-date information on listed stocks from various global markets. Leveraging this data, our model is equipped to analyze historical stock market trends and make accurate predictions, empowering investors and analysts with valuable insights for informed decision-making in the dynamic world of finance.

5.3 Theoretical framework

The theoretical framework of this project revolves around two main components: stacked Long Short-Term Memory (LSTM) networks and the Prophet model.

1. Stacked LSTM Networks:
   - Stacked LSTM networks are a type of recurrent neural network (RNN) architecture that excel at capturing temporal dependencies in sequential data, making them well-suited for time series forecasting tasks like stock price prediction.
   - The architecture consists of multiple LSTM layers stacked on top of each other, allowing the model to learn hierarchical representations of the data and capture complex patterns across different levels of abstraction.
   - Stacked LSTMs address the limitations of traditional single-layer LSTMs by providing increased model capacity and learning capabilities, leading to improved predictive performance.

2. Prophet Model:
   - The Prophet model, developed by Meta (formerly Facebook), is a time series forecasting tool designed for simplicity, flexibility, and high accuracy.
   - It decomposes time series data into trend, seasonality, and holiday components, enabling the model to capture various patterns and fluctuations inherent in the data.
   - Prophet incorporates sophisticated Bayesian modeling techniques to generate probabilistic forecasts, allowing for uncertainty estimation and robust predictions.
   - The model's intuitive interface and automatic handling of outliers and missing data make it an attractive choice for time series forecasting tasks, including stock market prediction.

By leveraging these theoretical frameworks, our project aims to develop and compare predictive models capable of accurately forecasting stock prices. Through the integration of stacked LSTM networks and the Prophet model, we seek to exploit the strengths of each approach and gain deeper insights into their respective performance characteristics in the context of stock market analysis.

V. Comparison and Result

6.1 Comparison

Comparing stacked LSTM with the Prophet algorithm made by meta on MAE,

MSE, RMSE and MAPE parameters.

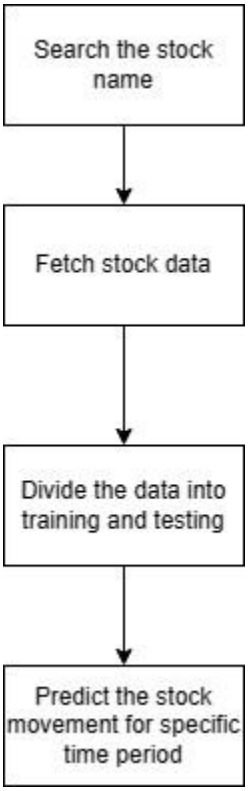


Figure 7: Flowchart

1. Root Mean Squared Error (RMSE):
   - RMSE is a measure of the average deviation of predicted values from the actual values, expressed in the same units as the data.
   - It calculates the square root of the average of squared differences between predicted and actual values.
   - RMSE penalizes larger errors more heavily than smaller errors, making it sensitive to outliers.
   - Lower RMSE values indicate better model performance, with 0 being a perfect fit.

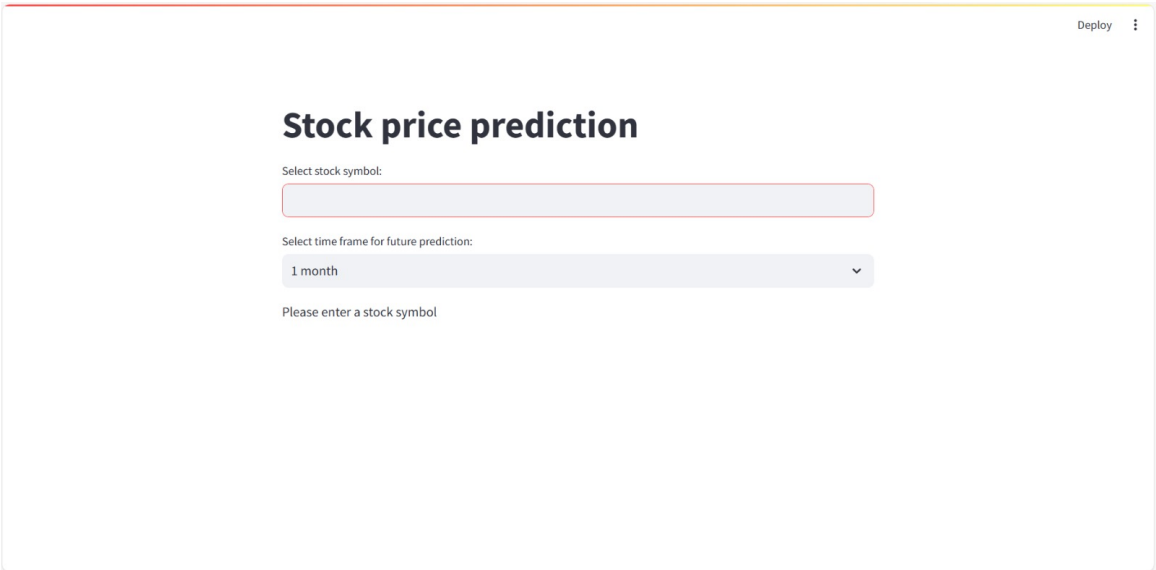
2. Mean Absolute Percentage Error (MAPE):
   - MAPE measures the average percentage difference between predicted and actual values.
   - It calculates the absolute percentage difference for each data point and averages them.
   - MAPE is expressed as a percentage, making it easy to interpret.
   - MAPE is sensitive to the scale of the data and can be misleading when actual values are close to zero.
   - Lower MAPE values indicate better accuracy, with 0 indicating a perfect fit.

3. Mean Squared Error (MSE):
   - MSE calculates the average of squared differences between predicted and actual values.
   - It is similar to RMSE but does not take the square root, so the values are in the original units squared.
   - MSE penalizes larger errors more heavily than smaller errors, similar to RMSE.
   - Lower MSE values indicate better model performance, with 0 being a perfect fit.

4. Mean Absolute Error (MAE):
   - MAE measures the average absolute difference between predicted and actual values.
   - It calculates the absolute difference for each data point and averages them.
   - MAE is less sensitive to outliers compared to RMSE and MSE.
   - MAE is expressed in the same units as the data, making it easy to interpret.

- Lower MAE values indicate better accuracy, with 0 indicating a perfect fit.

We have taken the example of GOOG or Alphabet (The parent company of Google) with a time frame of 1



month, 3 months, 6 months and 1 year predictions.

Figure 8: UI Screenshot

The UI consists of a text box to enter the name of the stock and the time period for which the prediction is expected.

The following are the results for the Prophet and LSTM models for 1-month prediction. We first get the stock data since the day of the listing which is then used for training and testing.

Date	Open	High	Low	Close	Adj Close	Volume
2004-08-19 00:00:00	2.4907	2.5918	2.39	2.4991	2.4991	897,427,216
2004-08-20 00:00:00	2.5158	2.7168	2.5031	2.6976	2.6976	458,857,488
2004-08-23 00:00:00	2.7584	2.8264	2.7161	2.7248	2.7248	366,857,939
2004-08-24 00:00:00	2.7706	2.7796	2.5796	2.612	2.612	306,396,159
2004-08-25 00:00:00	2.6142	2.6899	2.5873	2.6401	2.6401	184,645,512
2004-08-26 00:00:00	2.614	2.6887	2.6067	2.6877	2.6877	142,572,401
2004-08-27 00:00:00	2.6924	2.7054	2.6324	2.6438	2.6438	124,826,132
2004-08-30 00:00:00	2.6222	2.6274	2.5407	2.5407	2.5407	104,429,967
2004-08-31 00:00:00	2.548	2.5831	2.5445	2.5497	2.5497	98,825,037
2004-09-01 00:00:00	2.5579	2.5646	2.4824	2.4969	2.4969	183,633,734
2004-09-02 00:00:00	2.4705	2.5497	2.4643	2.5283	2.5283	303,810,504
2004-09-03 00:00:00	2.5143	2.534	2.4737	2.4909	2.4909	103,538,639
2004-09-07 00:00:00	2.5158	2.5405	2.481	2.53	2.53	117,506,800
2004-09-08 00:00:00	2.5091	2.5661	2.5031	2.548	2.548	100,186,120
2004-09-09 00:00:00	2.5537	2.5582	2.5156	2.5482	2.5482	81,620,792
2004-09-10 00:00:00	2.5305	2.6541	2.523	2.6234	2.6234	174,804,764

Figure 9: Raw Extracted Data

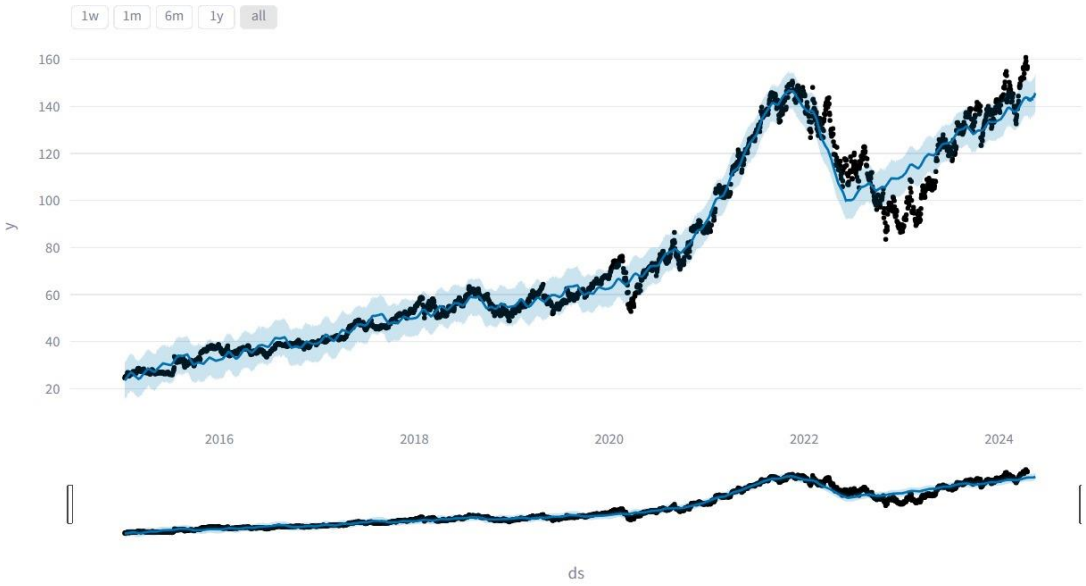


Figure 10: Prophet model 1 month prediction

Above is the prophet model's prediction. The Light Blue line represents the prediction for one month from the last day of available data.

The testing the model on the parameters we are testing on, we get the following results:

Model Evaluation for Prophet Model for 1 month:

Root Mean Squared Error (RMSE): 8.23

Mean Absolute Error (MAE): 5.86

Mean Squared Error (MSE): 67.76

Mean Absolute Percentage Error (MAPE): 4.10%

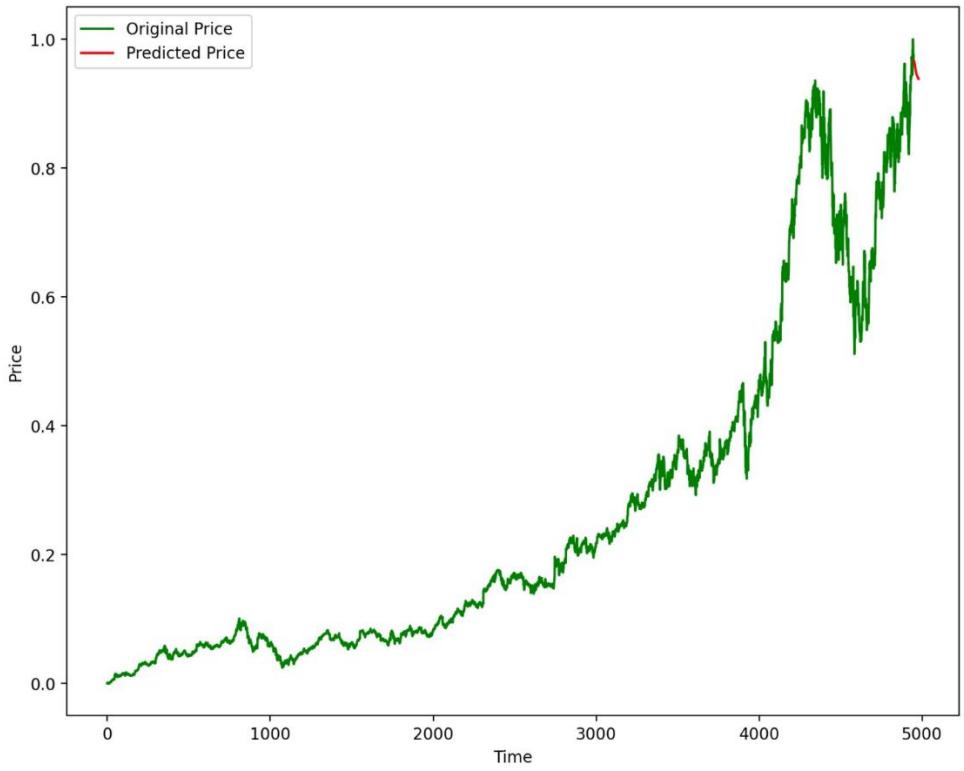


Figure 11: Stacked-LSTM model 1 month prediction

Above is the stacked-LSTM model's prediction. The red line represents the prediction for one month from the last day of available data.

The testing the model on the parameters we are testing on, we get the following results:

Model Evaluation for 1 month

Root Mean Squared Error (RMSE): 0.06

Mean Absolute Error (MAE): 0.05

Mean Squared Error (MSE): 0.00

Mean Absolute Percentage Error (MAPE): 4.45%

The following are the results for the Prophet and LSTM models for 3-month prediction.

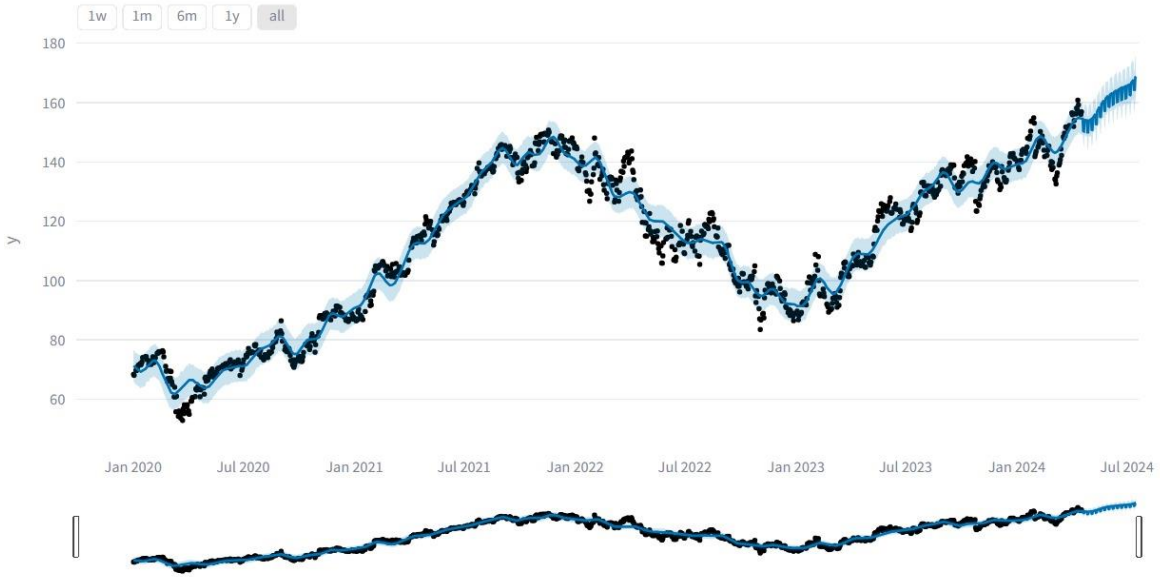


Figure 12: Prophet model 3 month prediction

Above is the prophet model's prediction. The Light Blue line represents the prediction for one month from the last day of available data.

The testing the model on the parameters we are testing on, we get the following results:

Model Evaluation for Prophet Model for 3 months:

Root Mean Squared Error (RMSE): 16.14

Mean Absolute Error (MAE): 14.94

Mean Squared Error (MSE): 260.52

Mean Absolute Percentage Error (MAPE): 10.48%

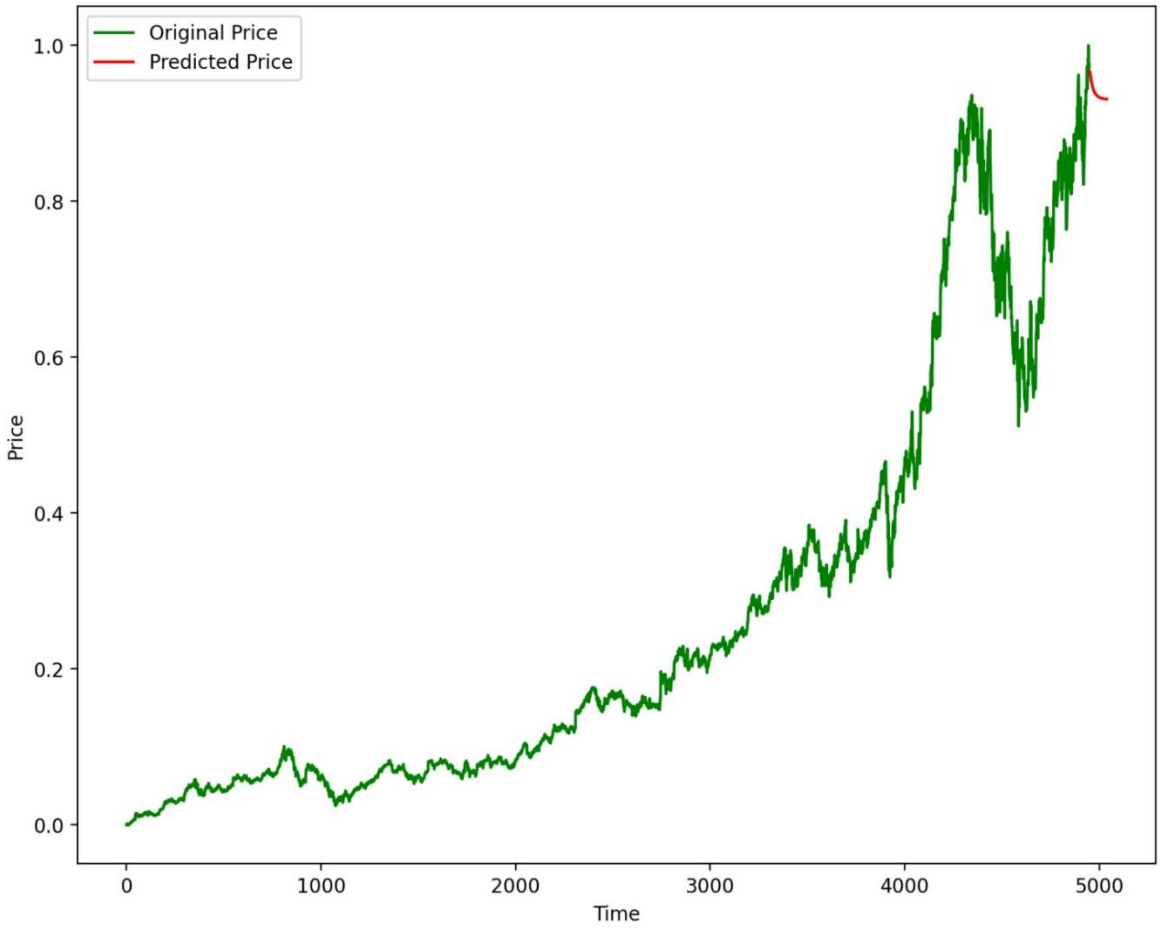


Figure 13: Stacked-LSTM prediction 3 month prediction

Above is the stacked-LSTM model's prediction. The red line represents the prediction for one month from the last day of available data.

The testing the model on the parameters we are testing on, we get the following results:

Model Evaluation for 3 months

Root Mean Squared Error (RMSE): 0.06

Mean Absolute Error (MAE): 0.05

Mean Squared Error (MSE): 0.00

Mean Absolute Percentage Error (MAPE): 5.84%

The following are the results for the Prophet and LSTM models for 6-month prediction.

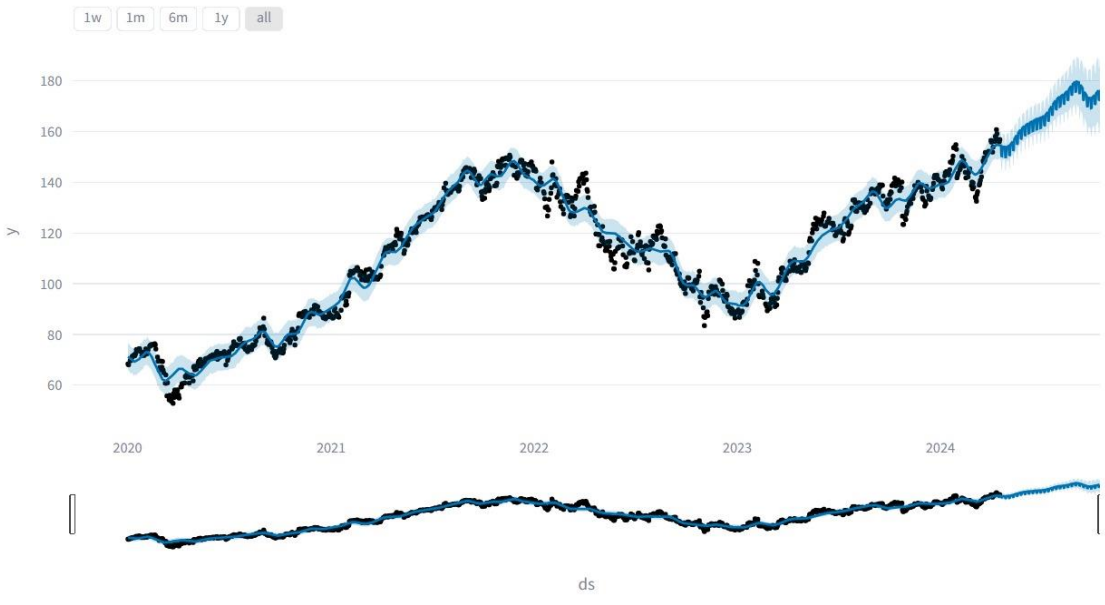


Figure 14: Prophet model 6 month prediction

Model Evaluation for 6 months prediction
Root Mean Squared Error (RMSE): 28.38
Mean Absolute Error (MAE): 27.57
Mean Squared Error (MSE): 805.61
Mean Absolute Percentage Error (MAPE): 19.93%

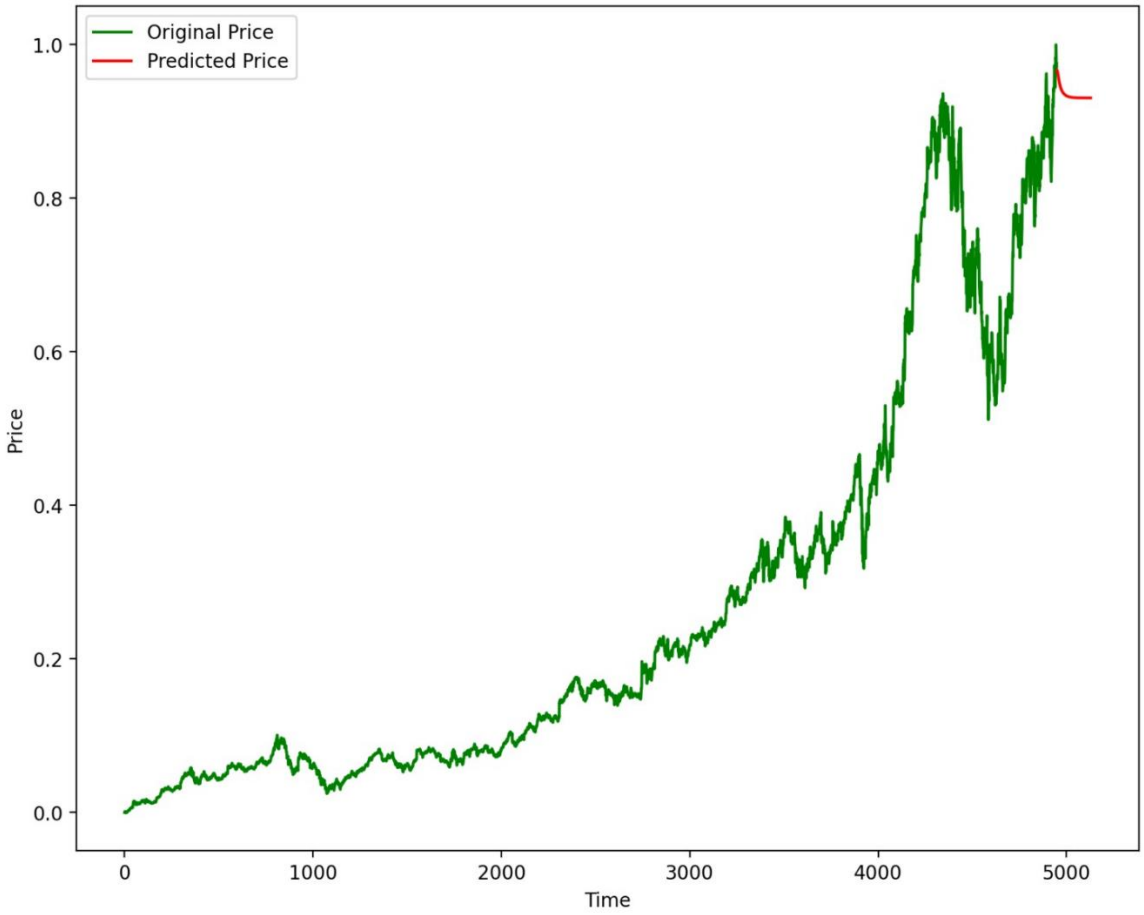


Figure 15: Stacked-LSTM 6 month prediction

Above is the stacked-LSTM model's prediction. The red line represents the prediction for one month from the last day of available data.

The testing the model on the parameters we are testing on, we get the following results:
Model Evaluation for 6 months
Root Mean Squared Error (RMSE): 0.09
Mean Absolute Error (MAE): 0.08
Mean Squared Error (MSE): 0.01
Mean Absolute Percentage Error (MAPE): 9.13%

The following are the results for the Prophet and LSTM models for 1 year prediction.

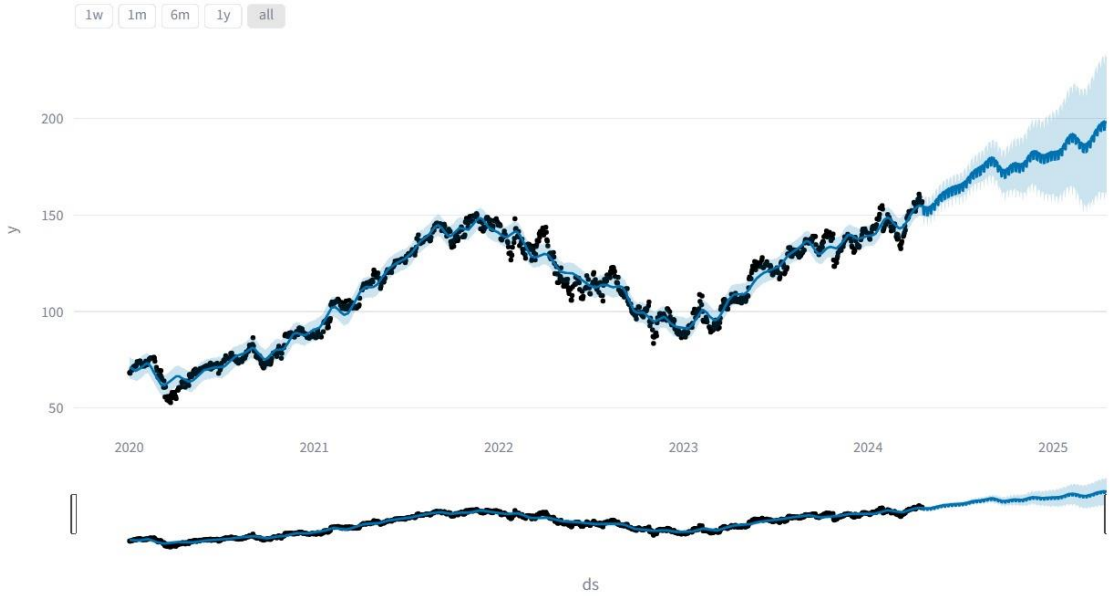


Figure 16: Prophet model 1 year prediction

Model Evaluation for 1 year prediction
Root Mean Squared Error (RMSE): 54.91
Mean Absolute Error (MAE): 53.81
Mean Squared Error (MSE): 3015.17
Mean Absolute Percentage Error (MAPE): 46.86%

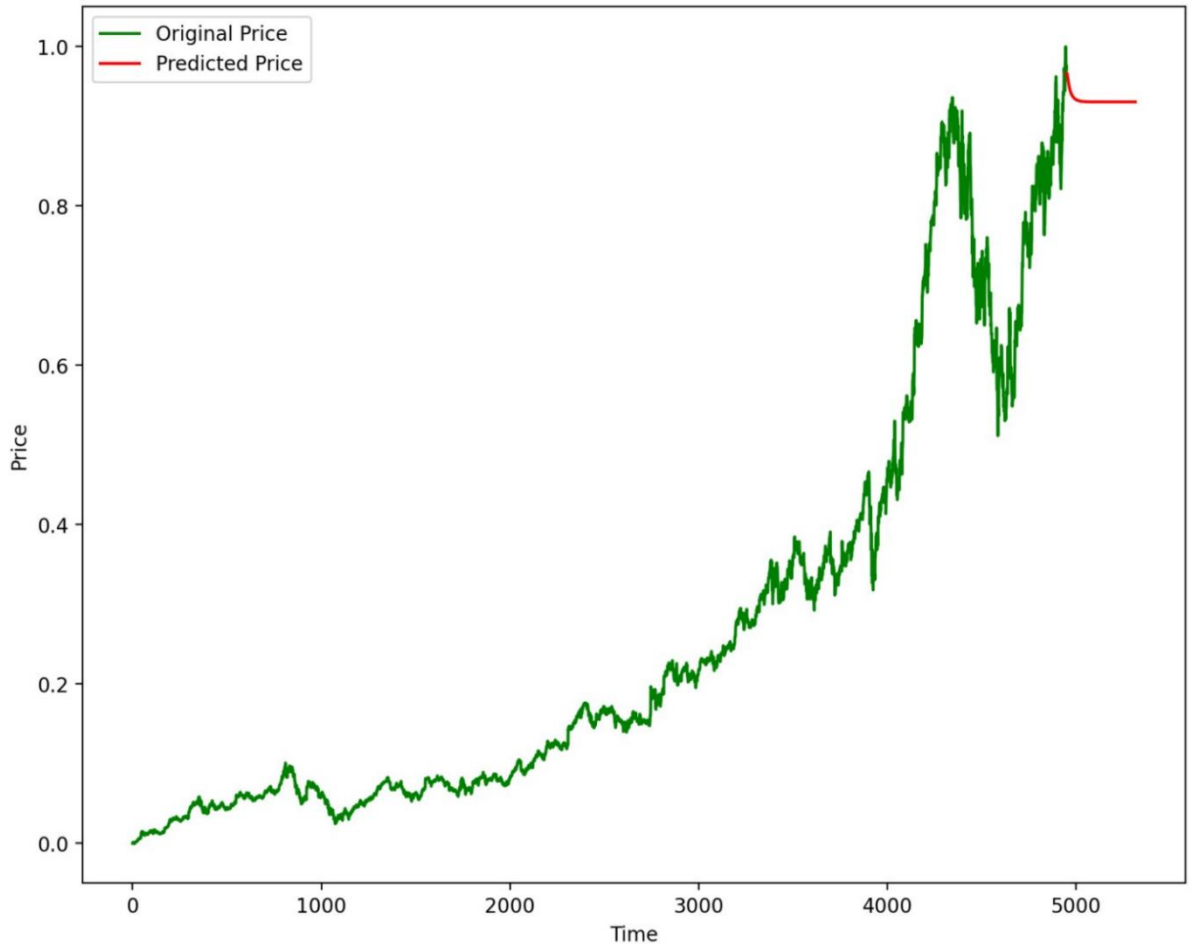


Figure 17: Stacked-LSTM model 1 year prediction

Above is the stacked-LSTM model's prediction. The red line represents the prediction for one month from the last day of available data.

The testing the model on the parameters we are testing on, we get the following results:

Model Evaluation LSTM FOR 1 YEAR
Root Mean Squared Error (RMSE): 0.22
Mean Absolute Error (MAE): 0.18
Mean Squared Error (MSE): 0.05
Mean Absolute Percentage Error (MAPE): 27.71%

6.2 Conclusion

We analyzed and compared the Prophet model to the stacked LSTM model. The parameters showcase that the stacked LSTM model performs much better than the Prophet model.

Stacked LSTM were well for the timeframes which we have analyzed them for. Their performance starts deteriorating as we increase the prediction timeframe expectations.

Using the simple UI/UX, we ensure that the user can use the model efficiently to analyze the trend and predict the stock market. By using high-quality data from Yahoo Finance, we ensure the results are not tainted.

A lot of research and innovation has been made in this field of study and more is yet to be done to make better models to predict and find the trends of the stock market.

VI. Acknowledgment

References

[1] A Survey on Stock Price Trend Prediction using Multi-Step Stacked LSTM by Yunish Kumar Kushwaha, Santosh Nagar, & Anurag Shrivastava in ISSN: 2582-5488, Volume-4 Issue-6, December 2022

[2] Stock Market Prediction via Deep Learning Techniques: A Survey by Jinan Zou, Qingying Zhao, Yang Jiao, Haiyao Cao, Yanxi Liu, Qingsen Yan, Ehsan Abbasnejad, Lingqiao Liu, Javen Qinfeng Shi. Citied as arXiv:2212.12717 [q-fin.GN] Feb 2023

[3] Stock Price Prediction Using Deep Learning Techniques by Abhilasha Raghate, Nachiket Tare, Sakshi Magare, Sharvari Holkar, and Varadh Sidgiddi in Volume:05/Issue:04/April-2023 of IRJMETS

[4] Stock Market Prediction Using LSTM by Shaikh Shoieb Abubaker and Syed Rouf Farid in Volume 10, Issue IV, April 2022 of IJRASET

[5] Short-term stock market price trend prediction using a comprehensive deep learning system by Jingyi Shen & M. Omair Shafiq in Journal of Big Data volume 7, Article number: 66 (2020)