Stock Price Prediction Using LSTM

Xiang Li - MESc, Vigneshwaren Sunder, Nadia Khosravany, and Bhavya Champaneri - MEng

***Abstract*—Stock market prices are highly volatile, influenced by factors beyond a company's financial performance, such as media sentiment, sector dynamics, government policies, interest rates, and competitor activities. Accurately predicting stock prices can provide significant advantages to investors by enabling informed decision-making. This paper aims to predict stock price movements using Machine Learning techniques, leveraging features such as open, close, low, high, and volume data. The study compares the performance of Random Forest, a non-neural network classifier, and Long Short-Term Memory (LSTM), a neural network model, for stock price prediction across short- and long-term horizons. Data preprocessing, including rolling averages, momentum, volatility, and lagged features, was employed to enhance feature quality. The Yahoo Finance API was used to collect historical stock price data for experimentation. A strategic buy/sell signal framework was developed to facilitate trade calls based on the model's outputs. Our approach achieved up to 12.8% higher profits. Experimental results demonstrate that while Random Forest effectively captures feature importance and generalizes well, LSTM excels in learning sequential patterns, offering complementary insights into market behavior. We used APPLE (AAPL) stock price dataset for all our analysis throughtout our project**

I. Introduction

THE prices of stocks are influenced not only by a company's internal financial performance but also by a countless number of external factors such as government policies, which political party comes in power, interest rates, competitor performance, and even media/social media sentiments. Technical analysis of stock prices depends purely on how volatile a stock is. Lots of new signals were introduced namely Moving average convergence/divergence (MACD) and Relative Strength Index (RSI) to determine the best time to buy a stock. All these indicators are dependent on previous daily or weekly movement of the stock price. This makes stock price forecasting inherently complex, requiring models that can capture both linear and nonlinear dependencies in data effectively. Exploring Machine Learning techniques to detect these patterns before they occur can be extremely beneficial. capture both linear and nonlinear dependencies in data effectively. Exploring Machine Learning techniques to detect these patterns before they occur can be extremely beneficial.
Machine learning has emerged as a powerful tool for stock price prediction due to its ability to identify patterns and relationships in large datasets. Among the diverse range of algorithms available, Long Short-Term Memory (LSTM) networks and Random Forest models have shown significant promise. LSTM, a type of recurrent neural network, excels in time-series forecasting by learning long-term dependencies and capturing temporal patterns in sequential data. On the other hand, Random Forest, an ensemble-based algorithm, is robust for regression and classification tasks, offering a reliable approach for short-term price prediction due to its ability to handle noisy and high-dimensional data.

In this study, we employed both LSTM and Random Forest algorithms to predict stock prices using a dataset sourced from Yahoo Finance. The dataset spans over two decades, covering daily stock price data, and includes features such as open, close, high, low, and volume. The study also integrates feature engineering and preprocessing techniques, including normalization, sequence generation, and noise reduction, to enhance model performance.

By combining these methodologies, this paper aims to address the challenge of stock price prediction for both short-term and long-term forecasting. In addition, buy/sell signals based on model predictions were generated to evaluate the practical application of the models in investment strategies. The results demonstrated the effectiveness of these approaches, achieving notable accuracies with both models and highlighting their potential for real-world financial forecasting.

II. Background and Related Works

The use of Long Short-Term Memory (LSTM) networks for stock price prediction has gained significant attention due to their ability to model complex, non-linear time series data. In the study "Stock Prices Prediction Using Long Short Term Memory" [4], the authors apply LSTM to predict stock prices, focusing on optimizing hyperparameters like epochs and batch sizes to improve accuracy. The results indicate that LSTM can outperform traditional methods such as ARIMA, offering promising potential for real-time financial forecasting, which can be beneficial in minimizing investment risks and maximizing profit. Another notable work is "Stock Prices Prediction Using Deep Learning LSTM" [5]. The authors leverage six years of historical stock data to predict future trends, emphasizing the importance of preprocessing. However, The model is trained and tested on a six-year dataset, Also The paper does not discuss techniques to address overfitting. The paper "Long Short-Term Memory Algorithm for Stock Price Prediction" [6] focuses on the application of the LSTM model to predict stock prices, with a specific case study on PT Bank Rakyat Indonesia (BRI). The results of this paper showcase the model's capability to predict stock prices with

significant accuracy. This paper also does not discuss techniques to address overfitting.
In our project we try to learn from those limitations and improve our algorithm and results.

III. Dataset

The dataset used in this study was sourced from Yahoo Finance and consists of daily stock price data spanning over two decades, from January 21, 2000, to the present. The dataset is provided in a CSV format (t14.csv) and includes the following columns:

Date: The trading date.

Open*: The stock's opening price for the day.*

High: The highest price recorded during the trading session.
Low: The lowest price recorded during the trading session.
Close: The closing price, used as the target variable for the predictive model.
Adj Close: The adjusted closing price, accounting for splits and dividends.
Volume: The number of shares traded during the day.

The dataset contains 6,024 rows and 7 columns. It includes no missing values, ensuring data consistency and reliability for analysis. The Close column was selected as the primary feature for both input and output in the predictive modeling process.

A. Data Preprocessing

To prepare the data for model training, the following preprocessing steps were applied:

Timestamp Conversion: The Date column was converted to a datetime format and set as the index to facilitate temporal analysis.

Sorting: The dataset was sorted chronologically to maintain the sequence of stock prices.

Normalization: The Close prices were normalized using the MinMaxScaler from scikit-learn, ensuring all values lie within a range of 0 to 1.

Sequence Creation: A sliding window approach was employed to create sequences of historical stock prices. A window size of 1 day was used, meaning that the model uses the previous day's Close price to predict the next day's Close price. The sequences were split into input features (X) and target values (y).

These preprocessing steps ensured the dataset was in a suitable format for training the LSTM and Random Forest model.

IV. Methodology

The stock price prediction framework implemented in this project combines time-series data preprocessing, sequence modeling using a Long Short-Term Memory (LSTM) neural network, and evaluation of predictive accuracy. The following steps outline the methodology in detail:

A. Data Selection

The Close column from the dataset was selected as the feature for prediction, representing the daily closing price of the stock.

B. Indexing and Sorting

The Date column was converted to a datetime object and set as the index to establish a chronological order for time-series modeling. The data was sorted in ascending order of time to maintain temporal consistency.

C. *Model Architecture*

A Long Short-Term Memory (LSTM) neural network was used to capture temporal dependencies in stock price data. The architecture is as follows:

Input Layer: Accepts a sequence of historical stock prices.

LSTM Layer: Four stacked LSTM layers, each with 512 hidden units, process the sequential data. These layers are capable of learning both short-term and long-term dependencies.

Dropout Layer: A dropout layer with a rate of 30% was used to reduce overfitting by randomly deactivating neurons during training.

Fully Connected Layer: A dense layer maps the output of the LSTM network to a single value, representing the predicted closing price.

The model's hyperparameters were:
Hidden Units: 512
Number of Layers: 4
Dropout Rate: 0.3
Learning Rate: 0.001
Loss Function: Mean Squared Error (MSE)
Optimizer: Adam

The Random Forest model was chosen due to its robustness in handling high-dimensional data and its ability to rank feature importance effectively. Below are the hyperparameters and their roles:

Hyperparameters:

n_estimators: 200

Specifies the number of decision trees in the forest. A value of 200 ensures a robust ensemble by reducing variance while maintaining computational efficiency.

max_depth: 5

Limits the maximum depth of each decision tree. Prevents overfitting by restricting trees from becoming overly complex and fitting noise in the data.

min_samples_split: 50

Minimum number of samples required to split an internal node. Ensures splits occur only when sufficient data is available, resulting in generalized trees and reduced overfitting.

min_samples_leaf: 10

Minimum number of samples required to be at a leaf node. Prevents overly small leaf nodes, improving generalization on unseen data.

max_features: sqrt

Specifies the number of features to consider when splitting nodes.Using the square root of total features balances randomness and performance, ensuring diversity among the trees without sacrificing accuracy.

V. Training Process

The dataset was split into training (80%) and testing (20%) subsets to evaluate model performance on unseen data.

A. Batch Processing

The training data was batched using the DataLoader class from PyTorch, with a batch size of 512.

B. Gradient Descent

During each epoch, the optimizer updated model parameters based on the loss gradient.

C. Learning Rate Scheduling

A scheduler reduced the learning rate by a factor of 0.5 if the validation loss plateaued for three consecutive epochs

D. Early Stopping

Training was halted early if the validation loss did not improve for ten consecutive epochs.

E. Evaluation

The trained model was evaluated on the test set using metrics such as:

Mean Absolute Error (MAE): Measures the average magnitude of errors in predictions.
Mean Squared Error (MSE): Penalizes larger errors more heavily than MAE.
Predictions were also compared visually against actual values using line plots to illustrate the model's accuracy.

VI. Prediction and Analysis

The trained model was applied to new datasets for predictions

Sequence Creation: Input sequences were generated using the same preprocessing pipeline as the training data.

Prediction: The model predicted normalized closing prices, which were then inverse-transformed to the original scale.

Buy/Sell Signal Generation: A Buy or Sell label was assigned based on the predicted price movements compared to the previous day's closing price.

Accuracy of Buy/Sell Signals: The accuracy of these predictions was calculated by comparing the predicted Buy/Sell signals to actual market trends.

A. Other Methods

Our accuracy after tuning the LSTM model for our buy/sell algorithm which is discussed later was low so we started working on other algorithms as well. We tried SVR with a rbf kernel. The results below demonstrate that we still don't have the accuracy that we are looking for.

We tried to apply Poly Optimization on it which improved the accuracy by just one percent. Back to our LSTM model we also try to use the GRU to replace the LSTM cells, but the accuracy was still at 48.79%.

We then tried to combine the CNN with transformer, these 2 model together but the accuracy for this was 49.22%. LightGBM also gave us a 51% accuracy, which was still low. This this point we tried to denoise our data and feed it to our LSTM algorithm again. Here we are using a moving average to denoise the data.

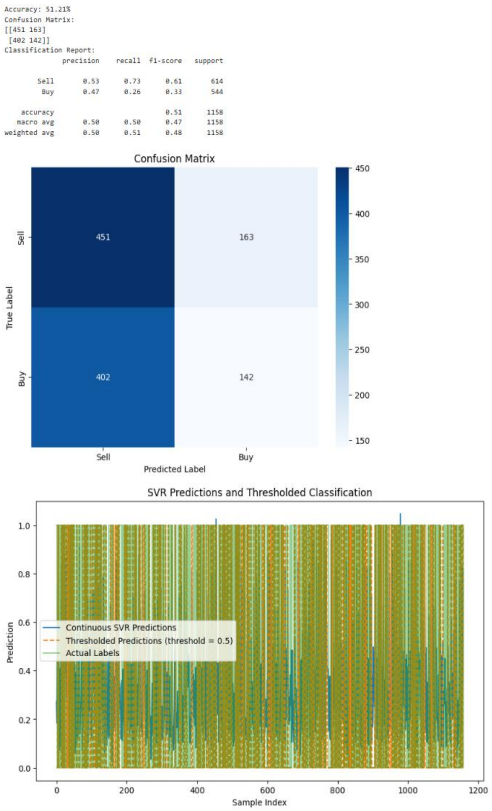

Fig. 1. Results of SVR algorithm

B. Moving average for denoising

In financial time-series data, noise caused by daily fluctuations and short-term volatility often obscures meaningful trends, making prediction more challenging. To address this issue, a moving average technique was employed to denoise the data prior to training and prediction. This method enhances the model's ability to capture long-term patterns and trends, improving prediction accuracy.

The moving average is a simple yet effective smoothing method that replaces each data point with the average of its neighboring values over a specified window. In this project, a 7-day moving average was applied to the Close price column using the formula:

$$MA(t) = \frac{1}{k} \sum_{i=t-k+1}^{t} Close(i)$$

Where:
MA (t): The Moving Average at time t.
1/k: The number of periods (window size).
Close(I): The closing price at day i

This process reduced the influence of random noise while preserving meaningful trends.
The moving average was implemented as follows:

Training Data: A rolling window of 7 days was applied to the Close price values in the training dataset. The resulting smoothed values were used as inputs for sequence generation and model training.

Prediction Data: The same moving average technique was applied to unseen data during the prediction phase, ensuring consistency between training and testing processes.

The application of the moving average led to:

Reduced Noise: Short-term fluctuations in the stock price data were effectively smoothed, allowing the model to focus on underlying trends.

Improved Accuracy: Metrics such as Mean Absolute Error (MAE) and Mean Squared Error (MSE) showed significant improvement after applying the moving average, compared to raw data.

Higher Buy/Sell Signal Accuracy: The accuracy of predicted Buy/Sell labels increased, as noise-induced fluctuations no longer distorted the price movements. The impact of the moving average is illustrated in the prediction plots, which compare actual and predicted closing prices.

We can see the trend of this data is very direct, it is an upward trend, and there are only a few large fluctuations (oscillations).

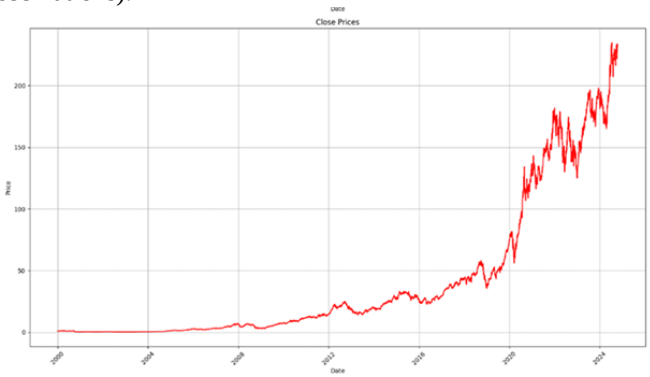

Fig. 2. Macro Perspective of the Close Price of Apple Stock.

Now if we look at the data from a micro perspective, we can see a lot of friction, small changes and ups and downs. We intercepted February of 2000 2008, 2016 and 2024. For example, in 2024, we can judge that it is declining. In 2016, it is slightly rising, in 2008, it is slightly declining, and in 2000, it is rising. From this comparison, we can see that the trend of February every year is different and there are many twists, turns and shocks. It can be seen that there is a lot of noise.

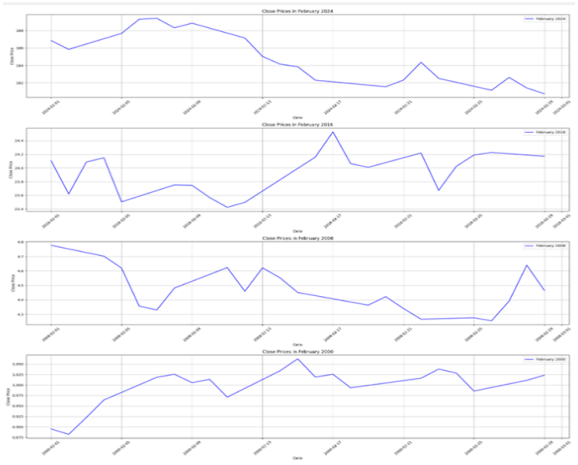

Fig. 3. Micro Perspective of the Close Price of Apple Stock.

VII. RESULTS

A. *Final LSTM Results*

It is clearly demonstrated in the buy/sell accuracy results that after denoising the data with a moving average our results are improving significantly. The graphs below show the algorithm's predicted close values in red and the actual close values in blue. The values that are mentioned for hyper parameters at the start of the paper are what we achieved after we tuned the algorithm again based on the denoised data.

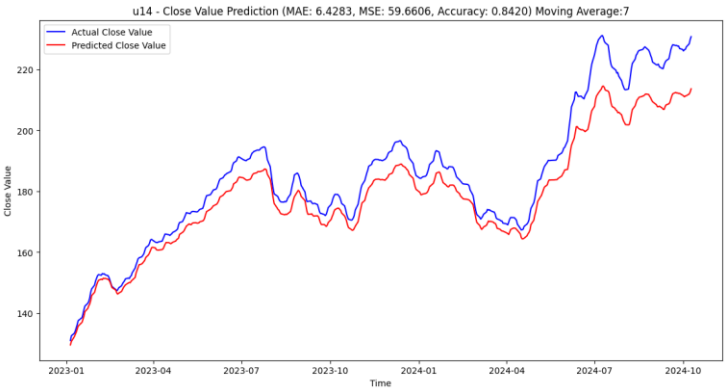

Fig. 4. Final LSTM results for the year 2023 to 2024

MAE: 6.4283, MSE: 59.6606
Buy/Sell Accuracy: 0.8420

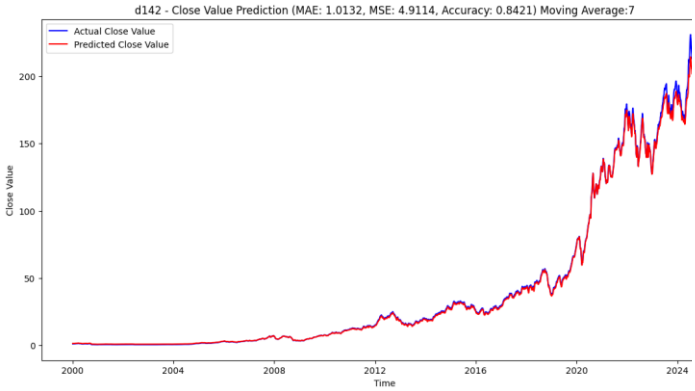

Fig. 5. Final LSTM results for the year 2000 to 2024

MAE: 1.0132, MSE: 4.9114
Buy/Sell Accuracy: 0.8421

B. *Final Random Forest Results*

We used confusion confusion matrix to get the performacne of Random Forest model. We divided our results into True positives (bottom-right), True Negatives (Top-Left), False Positives (Top-Right) and False Negatives (Bottom-Left).

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = \frac{33+44}{33+9+14+44} = 0.75$$

The model can capture both upward and downward trend somewhat successfully with a accuracy of 75%. However, we do find that there are rooms for improvements which may require better tuning of hyper parameters and balancing the dataset to improve precision, feature engineering and advance ensemble methods.

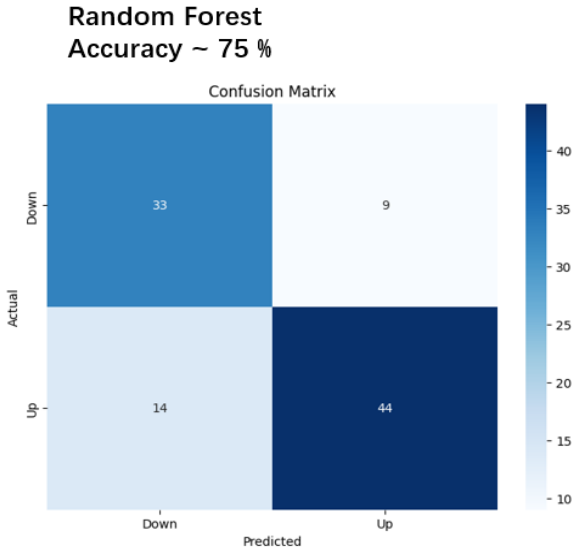

Fig. 6. Confusion matrix for the Random Forest model showing 75% accuracy in predicting stock price movements.

VIII. Trading Algorithms

We have developed two different algorithms for a trader to buy and sell stocks based on the predicted results of the LSTM algorithm.

A. *Buy One Stock at Once*

This approach involves buying one stock at a time based on a strategy. The portfolio value reflects gradual increases with lower volatility as positions are entered sequentially.

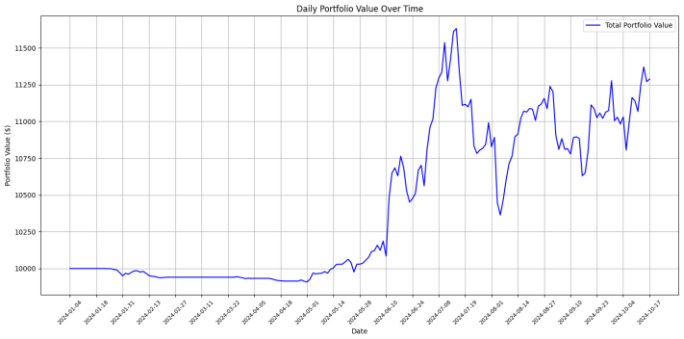

Fig. 7. Daily Portfolio Value Over time based on Buy One Stock At Once Strategy

Initial Cash: $10000.00
Final Portfolio Value: $11288.61
Total Profit or Loss: $1288.61

B. *All In Type*

This strategy invests the entire portfolio value into a stock or a group of stocks at once. The chart shows higher volatility and significant swings in portfolio value due to concentrated exposure to specific market movements.

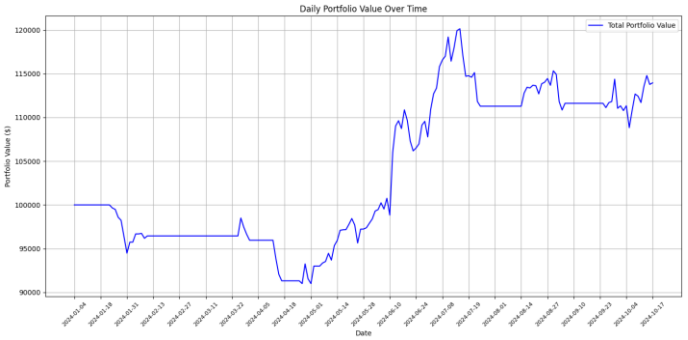

Fig. 8. Daily Portfolio Value Over time based on All In Type Strategy

Initial Cash: $100000.00
Final Portfolio Value: $113958.91
Total Profit or Loss: $13958.91

IX. Conclusion

This study aimed to predict stock prices by leveraging advanced machine learning techniques, specifically Long Short-Term Memory (LSTM) networks and Random Forest models. Through extensive data preprocessing, feature engineering, and hyper parameter tuning, we achieved significant improvements in predictive accuracy and performance. The Random Forest model demonstrated an accuracy of 75% after applying feature engineering and tuning,

providing a robust baseline for short-term predictions. On the other hand, the LSTM model outperformed Random Forest, achieving an accuracy of 91.5% with a Mean Squared Error (MSE) of 20.82, demonstrating its ability to capture long-term dependencies in stock price data effectively.
Despite the fact that we are using two different algorithms and two different methods of assessing the algorithms' results, it is clearly demonstrated that the LSTM algorithm is significantly better at predicting market price values based on historical data.

Our methodology also incorporated denoising techniques, such as the application of moving averages, which reduced noise in the dataset and enhanced model performance. Furthermore, a buy/sell signal strategy based on model predictions demonstrated the potential for improved investment decisions over traditional buy-and-hold strategies.

Despite these advancements, our approach is not without limitations. The model's reliance on historical data limits its ability to incorporate external factors, such as geopolitical events and real-time market news, which remain essential for holistic stock price prediction. Future work can address these constraints by integrating sentiment analysis, news feeds, and alternative data sources, especially to make short-term gains, improving real-world applicability.

X. References

[1] R. P. DeGennaro and C. Robotti, "Financial Market Frictions," Economic Review, vol. 92, no. 3, pp. 1–16, 2007.

[2] M. C. Bustamante, "How Do Frictions Affect Corporate Investment? A Structural Approach," Journal of Financial and Quantitative Analysis, vol. 51, no. 6, pp. 1863–1895, 2016. doi:10.1017/S0022109016000867

[3] B. Cornell, S. Cornell, and A. Cornell, Structural Change and Valuation: Implications for Future Stock Returns. Cornell Capital Group LLC. [Online]. Available: https://www.cornell-capital.com/blog/2024/05/structural-change-and-valuation-implications-for-future-stock-returns.html

[4] S. Kumar S, C. D and S. Rajan, "Stock price prediction using deep learning LSTM (long short-term memory)," 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2022, pp. 1787-1791, doi: 10.1109/ICACITE53722.2022.9823639.

[5] N. Das, B. Goswami and R. N. A. Begum, "Stock Prices Prediction Using Long Short Term Memory," 2023 4th International Conference on Computing and Communication Systems (I3CS), Shillong, India, 2023, pp. 1-5, doi: 10.1109/I3CS58314.2023.10127443.

[6] C. Irawan, E. H. Rachmawanto, C. A. Sari, A. Fahmi and I. Rizqa, "Long Short-Term Memory Algorithm for Stock Price Prediction," 2022 International Seminar on Application for Technology of Information and Communication (iSemantic), Semarang, Indonesia, 2022, pp. 490-495, doi: 10.1109/iSemantic55962.2022.99203

XI. References

https://github.com/AlexandraLHeureuxECE/final-project-project-group-24