**Implementation of Stock Market Trends Prediction via Recurrent Neural Network (RNN) Approaches.**

Sree Harsha Koyi, Vamshi Thadishetty, Srikar Kotra, Manikanta Bhardwaj

*The University of Texas at Dallas*

*{sxk230025, sxt220097, sxs230164, mxk230029}@utdallas.edu*

*Abstract*—This report presents a machine learning model using Recurrent Neural Networks (RNN) to predict stock market trends, specifically focusing on the 'Close/Last' price. The model utilizes a dataset of 1,258 daily instances of US stock market data from 2018 to 2023. Developed in Python with libraries such as NumPy, Pandas, and Matplotlib, the project details the process of data preprocessing, RNN model development, and training. The findings demonstrate the model's capability to effectively predict stock market trends, underscoring the potential of RNNs in financial time series analysis. This work contributes to both practical financial forecasting and academic research in machine learning applications in finance.

Keywords—Stock Market Prediction, Recurrent Neural Networks (RNN), Time Series Analysis, Financial Forecasting

# Introduction

The prediction of stock market trends has long been a subject of both financial importance and academic interest. The ability to accurately forecast market movements can significantly benefit investors, traders, and analysts. In recent years, the advent of advanced machine learning techniques has opened new avenues for predictive analytics in finance. This report focuses on the implementation of Recurrent Neural Networks (RNNs) to predict stock market trends, with a particular emphasis on analyzing the 'Close/Last' price of stocks. RNNs are especially suited for this task due to their ability to process sequences of data and capture temporal dependencies, a characteristic inherent in stock market time series.

# Background Work

The concept of using computational methods for stock market prediction is not new; however, the introduction of machine learning, particularly neural networks, has revolutionized this field. Traditional statistical methods often fall short in capturing the complex and dynamic nature of financial markets. RNNs, a class of neural networks, are uniquely capable of handling sequential data, making them ideal for time series analysis like stock prices.[2],[3].

Previous research has shown different degrees of effectiveness in using RNNs for financial forecasting, with approaches ranging from simple linear models to more complicated neural networks. RNNs in the forms of Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) networks, have demonstrated enhanced performance in capturing long-term dependencies in data.

The uniqueness of this project lies in its singular focus on the 'Close/Last' price as the primary feature for analysis, supplemented by additional stock market indicators for a comprehensive approach. By utilizing a dataset that spans five years of daily US stock market data, The purpose of this work is to add to the body of knowledge by providing fresh perspectives on the use of RNNs in stock market trend prediction. Theoretical And Conceptual Study

### **Time Series**

A time series is a collection or recording of data points at regular time intervals. This type of information is critical in a variety of sectors, including banking, economics, weather forecasting, and others. Time series data has the following characteristics:

#### **Temporal Dependence:** The value at a current time point is often dependent on previous time points.

#### **Seasonality**: Regular patterns or cycles observed in the data over specific time intervals.

#### **Trend:** The long-term progression of the series, which could be upward, downward, or stable.

#### **Noise**: Random variation in the data.

**Time Series Analysis in Finance:**

Time series analysis in finance is evaluating past market data to forecast future price fluctuations.. Stock market data is inherently a time series, with prices recorded at regular intervals (daily, weekly, etc.). This analysis can reveal underlying patterns and trends, crucial for making informed investment decisions.

Techniques for Time Series Analysis:

Moving Averages: Beneficial for emphasizing longer-term trends and mitigating short-term variations.

Autoregressive Models (AR): Dependencies between an observed value and a few lagged data are used in models.

Integrated (I): Using raw observation differencing to make the time series steady.

Moving Average Models (MA): Models that apply to lagged data the dependency between an observation and a residual error from a moving average model.

ARIMA (Autoregressive Integrated Moving Average): Combines AR, I, and MA[4].

**Time Series and Machine Learning:**

Machine learning offers advanced techniques to analyze and predict time series data:

1. Feature Engineering: Extracting and selecting relevant features from the time series data to improve model accuracy.
2. RNN and Its Variants: As previously discussed, RNNs, especially LSTMs and GRUs, are well-suited for modeling time series data due to their ability to capture temporal dependencies.
3. Time Series Forecasting in Stock Market Prediction
4. Forecasting stock market trends is a complex task due to the inherent uncertainty, noise, and the influence of numerous external factors. However, time series models, especially when combined with machine learning techniques like RNNs, can significantly enhance prediction accuracy by learning from historical data patterns.

**Challenges in Time Series Forecasting:**

1. Market Volatility: Stock prices are influenced by a myriad of unpredictable factors, making them highly volatile.
2. Non-Stationarity: Financial time series data often exhibit non-stationary behaviors, where statistical properties change over time.
3. Overfitting: Models might learn noise as a pattern, leading to poor performance on unseen data.

### **Recurrent Neural Networks**

RNNs are a type of artificial neural network that recognizes patterns in data sequences such as time series, audio, text, financial data, and others. RNNs, as opposed to standard feedforward neural networks, have the unusual attribute of retaining a ‘memory' of past inputs in their internal state, which effects the network's output. Because of this, they are particularly well-suited for activities requiring historical background.

**Core Concept and Architecture:**

An RNN's essential concept is its recurrent design, in which connections between units form a directed cycle. This results in the network acquiring an internal state that permits it to display dynamic temporal behavior. RNNs, unlike feedforward neural networks, may handle input sequences using their internal state (memory). This capacity to analyze sequences is what distinguishes RNNs in jobs like stock market trend prediction. A typical RNN unit is made up of an input layer, a hidden layer, and an output layer. The hidden layer activations are affected not only by the current input but also by the state of the previous hidden layer. Mathematically, this is often represented as

*Ht*​=*f*(*Wxh*​*Xt*​+*Whh*​*Ht*−1​+*b*)

where Ht​ is the current hidden state, Xt​ is the input at time t, W are the weights, and b is the bias.

**Limitations:**

The vanishing and exploding gradient problem is one of the most difficult challenges in RNN training. This happens during backpropagation through time (BPTT), when gradients can either vanish (become very little) or explode (become very huge), making efficient network training difficult. This issue is exacerbated when dealing with extended sequences, which is frequent in financial time series data.

To address the drawbacks of conventional RNNs, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) were developed.

LSTM: The long-term dependency issue is precisely what LSTMs are meant to avoid, according to Hochreiter and Schmidhuber (1997). They contain unique parts that can store knowledge for a long time called memory cells. A series of gates controls the information flow into and out of the cell, preventing the vanishing gradient issue.

GRU: Cho et al. (2014) developed GRUs, which are comparable to LSTMs but have a simpler structure. They combine the forget and input gates into a single "update gate" and blend the cell state and hidden state, resulting in a faster learning process.

RNNs in Financial Time Series Prediction:

In the context of financial time series prediction, RNNs offer a significant advantage due to their ability to capture temporal dependencies and patterns over time. Stock market data is inherently sequential and influenced by its own past behavior, which RNNs can model effectively. By utilizing RNNs, financial analysts and investors can potentially forecast future stock prices with greater accuracy, considering not just current market conditions but also historical trends.

**Implementation Considerations:**

When implementing RNNs for stock market prediction, several key considerations need to be addressed:

1. Feature Selection: Choosing relevant features (like 'Close/Last' price) that can provide meaningful insights into future trends.
2. Data Preprocessing: Normalizing data to ensure efficient training.
3. Hyperparameter Tuning: Selecting appropriate parameters like learning rate, number of hidden layers, and units to optimize performance.
4. Regularization: Implementing techniques like dropout to prevent overfitting.

Recurrent Neural Networks, with their unique ability to process sequential data and capture temporal dependencies, are well-suited for predicting stock market trends. The evolution from basic RNNs to more advanced forms like LSTMs and GRUs has opened new possibilities in accurately modeling financial time series data. As machine learning continues to advance, RNNs will likely play an increasingly vital role in financial analytics, offering more refined and sophisticated tools for market prediction.

# Model Implementation

#### Dataset:

1. The dataset used in this project is pivotal for training the Recurrent Neural Network (RNN) model to predict stock market trends. consists of historical stock market data, specifically daily US stock market data from 2018 to 2023.The dataset includes 1,258 instances, providing a comprehensive view of market behavior over this period.
2. The primary feature of interest is the 'Close/Last' price, which the model will use to forecast future trends. To enhance the model's accuracy, other stock market indicators, such as Volume, Open, High, and Low prices, are also included. The data undergoes preprocessing, where the 'Close/Last' price is extracted and normalized. Normalization is a critical step in preparing the data for machine learning models, as it scales the data within a specific range (in this case, between 0 and 1), making the training process more stable and efficient.

A graph showing a line of green lines

Description automatically generated with medium confidence

Fig. 1. Time Series Data for Stock Price

#### Implementaion:

**Data Preprocessing:** The initial phase of the implementation involved preprocessing the stock market dataset. The 'Close/Last' price, extracted from the dataset, was normalized to facilitate effective model training. The normalization process adjusted these values to a common scale, aiding in stabilizing and expediting the training process. This step is crucial for handling the inherent volatility and varying scales within financial data.

1. Sequence Creation: Post-normalization, the data was structured into sequences. Each sequence, of a predefined length (seq\_length), comprised a series of data points followed by a corresponding target value. This transformation was essential for the RNN to learn from the temporal dependencies inherent in time-series data, a characteristic feature of stock market trends.
2. Dataset Split: For training and assessment, the dataset was split into two subsets: testing and training. Eighty percent of the data comprised the training set, which was used to train the model. The remaining twenty percent of the data was utilized as the testing set, which evaluated the model's performance and capacity for generalization.
3. Model Development: The core of the implementation was the development of a basic RNN model using NumPy, which has shown promising results in similar studies [5], [6]. This model featured a simple yet customizable architecture, allowing for the adjustment of parameters such as the number of hidden units and the learning rate. These parameters were pivotal in experimenting with the model’s complexity and learning dynamics.
4. Training Process: During the training process, the RNN model was ran through numerous epochs, using forward and backward passes to change its weights and biases. This repeated method aims to reduce the loss function, improving the model's capacity to reliably predict future stock values.

#### Evaluation:

Three key metrics were employed to assess the model’s performance:

1. Mean Absolute Percentage Error (MAPE): Representing accuracy as a percentage, MAPE was particularly useful for comparing performance across different datasets or models.
2. Root Mean Square Error (RMSE): This metric evaluated the model's accuracy by determining the average magnitude of the differences between predicted and actual values.
3. Mean Absolute Error (MAE): MAE calculated the average size of errors in a series of predictions without taking direction into account.

#### Hyperparameter Tuning and Analysis:

The model’s performance was evaluated under varying conditions by adjusting hyperparameters such as the number of hidden units, learning rate, and epochs. The impact of these hyperparameters on the model’s accuracy was visually represented through a series of plots, facilitating an in-depth analysis of the model’s behavior and effectiveness.

V. RESULTS

The analysis presented in Figures 1-6 illustrates various performance metrics for the model under study. Figure 1 depicts the close price of a given asset over time, which correlates with findings from recent studies [1], [7]. where a significant peak is observed in early 2021 before a subsequent decline. The volatility in the market is evident, with notable fluctuations occurring throughout the observed period.

A diagram of a bar graph

Description automatically generated

Fig.2. Hidden units vs Test RMSE

Figure 2 examines the effect of different numbers of hidden units in a neural network on the Test Root Mean Square Error (RMSE). It is observed that increasing the number of hidden units from 10 to 20 reduces the median test RMSE, while further increase to 30 hidden units does not lead to a significant change.

A graph of a graph

Description automatically generated with medium confidence

Fig. 3. RMSE Train Vs Test

In Figure 3, we observe the RMSE analysis across different epochs for both training and test datasets. The training data RMSE remains consistently lower than the test data RMSE, indicating a possible overfitting scenario as the model learns.

A screen shot of a graph

Description automatically generated

Fig .4. Learning Rate vs Test RMSE

The impact of the learning rate on test RMSE is visualized in Figure 4. Different shades represent the number of hidden units. This scatter plot indicates that a lower learning rate, coupled with an increased number of hidden units, tends to result in a lower RMSE.

A graph of a graph

Description automatically generated with medium confidence

Fig. 5. MAPE Train Vs Test

Figure 5 shows us the Mean Absolute Percentage Error (MAPE) across epochs for both training and test datasets. The test data MAPE decreases as the number of epochs increases, suggesting an improvement in the prediction accuracy over time.

A graph of a graph

Description automatically generated with medium confidence

Fig. 6. MAE Train vs Test

Lastly, Figure 6 showcases the Mean Absolute Error (MAE) analysis over epochs. Similar to the MAPE results, the test data MAE decreases as the model is exposed to more epochs, which could imply a better model performance over the training period.

These results collectively suggest that the model's performance is sensitive to the number of hidden units and the learning rate. It also indicates that while the model can improve over time with more training, there is a potential risk of overfitting, as evidenced by the discrepancies between the training and test error metrics.

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Hidden Units | 30.0 |
| Learning Rate | 0.01 |
| Epochs | 10.0 |

Fig.7. Best Parameter Values.

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Train RMSE | 0.328274 |
| Test RMSE | 0.039906 |
| Train MAE | 0.267680 |
| Test MAE | 0.030796 |
| Train MAPE | 85.689025 |
| Test MAPE | 53.193157 |

Fig.8. Best Metrics Table

# VI. Conclusion

The goal of this study was to use recurrent neural networks (RNNs) to anticipate trends in the stock market, focusing on the 'Close/Last' price of stocks. The project successfully demonstrated the capability of RNNs to analyze and forecast stock market behavior, leveraging a rich dataset of daily US stock market data from 2018 to 2023 [8]. The model’s proficiency in capturing the dynamic and temporal aspects of financial time series data is a testament to the power of machine learning in financial analysis.

This study's findings not only reinforce RNNs' place as a vital tool in financial forecasting, but they also contribute to a broader knowledge of applying advanced machine learning techniques to complicated, real-world issues. In order to achieve trustworthy predicted performance, the research emphasizes the significance of proper data preparation, feature selection, and model tweaking.

## VII. Future Work

In the future, there are various avenues for additional research and development in this domain:

1. Incorporating Additional Data Sources: Future iterations of the model could benefit from integrating alternative data sources such as economic indicators, news sentiment, or social media trends to capture a more holistic view of the factors influencing stock prices.
2. Experimenting with Advanced Neural Network Architectures: Convolutional Neural Networks (CNNs) and Transformer models are two examples of neural network architectures that may provide better accuracy and efficiency in feature extraction.
3. Real-Time Data Analysis: Adapting the model to work with real-time data feeds could significantly enhance its practical utility for day-to-day trading and investment decisions.
4. Robustness and Generalization: Further research could focus on improving the model's robustness to market volatility and its generalization capabilities across different stock markets and economic conditions.
5. Explainability and Transparency: As machine learning models become more complex, ensuring their explainability and transparency becomes crucial, especially in high-stakes domains like finance.

By continuing to explore these areas, future work can build upon the foundation laid by this project, driving forward the intersection of machine learning and financial analysis. The goal is to develop models that are not only accurate but also adaptable, transparent, and robust, offering valuable tools for investors and analysts in navigating the complexities of the stock market.

##### References

[1] L. Mathanprasad and M. Gunasekaran, "Analysing the Trend of Stock Market and Evaluate the performance of Market Prediction using Machine Learning Approach," Jan 2022.

[2] J. Kumari, V. Sharma, and S. Chauhan, "Prediction of Stock Price using Machine Learning Techniques: A Survey," Dec 2021.

[3] S. Goswami and S. Yadav, "Stock Market Prediction Using Deep Learning LSTM Model," Oct 2021.

[4] S. B. Islam, M. M. Hasan, and M. M. Khan, "Prediction of Stock Market Using Recurrent Neural Network," Oct 2021.

[5] A. Vij, K. Saxena, and A. Rana, "Prediction in Stock Price Using of Python and Machine Learning," Sep 2021.

[6] K. J, E. Hemalatha, M. S. Jacob, and R. Dhanalakshmi, "Stock Price Prediction Based on LSTM Deep Learning Model," Jul 2021.

[7] D. Liu and A. Chen, "Research on Stock Price Prediction Method Based on Deep Learning," Dec 2020.

[8] A. Moghar and M. Hamiche, "Stock Market Prediction Using LSTM Recurrent Neural Network," in Proc. of the International Workshop on Statistical Methods and Artificial Intelligence, Warsaw, Poland, April 6-9, 2020.

**.**