DEEP LEARNING & REINFORCEMENT LEARNING

LSTM VS. GRU NEURAL NETWORK PERFORMANCE FOR STOCK MARKET PREDICTION

FEDERICO A. GORRINI

GitHub repository - *https://github.com/Federico-Gorrini-Ilz/IBM*

I. INTRODUCTION

I. INTRODUCTION

OBJECTIVE

This report focuses on comparing the performance of LSTM and GRU neural networks in the context of stock market prediction. By evaluating their predictive accuracy, and training efficiency, this study aims to determine which model offers the best balance of performance and practicality for stock price forecasting tasks. Through this comparison, we will explore how deep learning can contribute to better financial decision-making and the advancement of quantitative trading strategies.

For the current study, the stock share price for the company *Walmart* (WMT) on the *Nasdaq* market is used, although the code can be used for any other stock price.

The dataset consists of the stock's closing prices and corresponding dates, for the timeframe 2013-2022, sourced from *Yahoo Finance* using the *yfinance* library.

I. INTRODUCTION

MACHINE LEARNING FOR STOCK MARKET PREDICTION

The stock market, with its inherent volatility and complexity, has long been a subject of interest for researchers and financial professionals seeking to develop reliable methods for predicting price movements. Accurate stock price prediction is critical for investment strategies, risk management, and trading decisions, making it a prime area of exploration for machine learning (ML).

Over the past few decades, machine learning techniques have increasingly gained popularity due to their ability to uncover patterns and relationships in historical data that may not be immediately apparent to human analysts.

In particular, deep learning methods – such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) networks – have shown great promise in time series forecasting tasks like stock price prediction. Both LSTM and GRU are types of recurrent neural networks (RNNs) specifically designed to handle sequential data and capture temporal dependencies, which are crucial when analyzing stock prices that evolve over time.

LSTM networks, introduced in the late 1990s, are well-suited for tasks where long-term dependencies are important, thanks to their unique architecture that helps mitigate the vanishing gradient problem encountered in traditional RNNs. GRUs, a more recent variation, aim to achieve similar performance with a simpler design, making them computationally more efficient. Despite their differences, both architectures have been widely applied to financial forecasting, as they can learn intricate patterns from historical stock price data, technical indicators, and even news sentiment.

I. INTRODUCTION

INTRODUCTION TO RNN, LSTM, AND GRU

Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) are two popular variants of Recurrent Neural Networks (RNN), each designed to handle long-term dependencies in sequential data. These models are particularly well-suited for tasks such as time series prediction, natural language processing (NLP), and speech recognition, where the relationships between data points span across time. Unlike traditional artificial neural networks (ANN), which are typically designed to process static, non-sequential data, RNNs, LSTMs, and GRUs are specifically engineered to capture the temporal dependencies in sequence data.

In sequence-based tasks, like language modeling or stock price prediction, the input data is ordered, and each data point is often dependent on its predecessors. For example, in a sentence, the meaning of a word is influenced by the words before it, and this context is crucial for accurate understanding. Similarly, in time series data, future values depend on the historical sequence. Traditional feedforward ANNs cannot maintain memory of past inputs, which limits their performance on such tasks. However, RNNs were designed to address this by introducing feedback loops, allowing the model to use its own previous outputs as part of the input at each time step.

Despite their ability to model sequential data, traditional RNNs face a critical limitation: they struggle with long-term dependencies. This is due to issues related to the vanishing gradient problem, where the gradients can become very small as they are propagated back through time. As a result, RNNs tend to "forget" information from earlier time steps, making them less effective at capturing long-term dependencies in the data. If the gradients are too small, the model cannot learn effectively, while large gradients can cause instability and lead to exploding gradients, further complicating the training process.
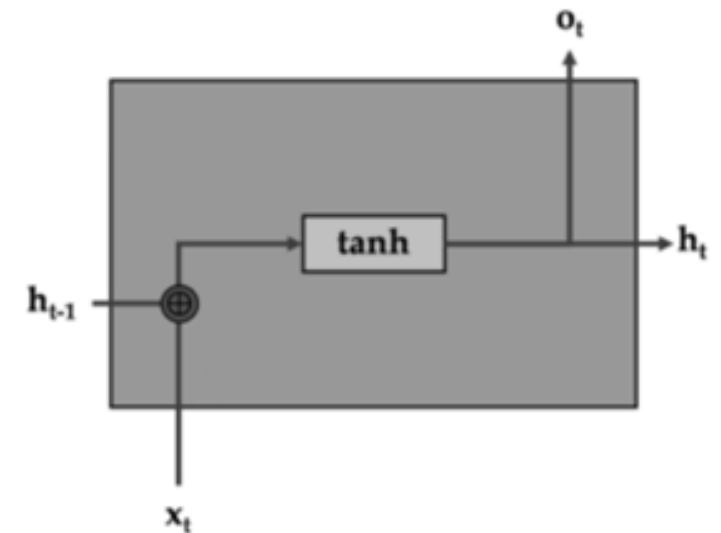
I. INTRODUCTION

RECURRENT NEURAL NETWORKS (RNN)

Recurrent Neural Networks (RNNs) are a class of neural networks specifically designed for sequential data. The key feature of an RNN is that it has a "memory" of previous inputs, allowing it to process sequences of data one element at a time and maintain context across time steps. This is achieved by feeding the output of a previous time step back into the network, which influences the current output.

Key Components

• Hidden State: At each time step, an RNN updates its hidden state based on the current input and the previous hidden state. The hidden state essentially captures the relevant information from the sequence up to that point.

• Weight Sharing: The same set of weights is shared across all time steps, making RNNs efficient for sequence processing.

• Output: The output at each time step is typically a combination of the hidden state and the current input, which is then passed to the next time step in the sequence.

Despite their ability to model sequential data, traditional RNNs struggle with long-term dependencies. This issue arises due to the vanishing gradient problem, where gradients – used for backpropagation during training – become extremely small as they are propagated backwards through many time steps. This causes RNNs to forget long-term dependencies, making them less effective for tasks that require capturing information over long sequences.
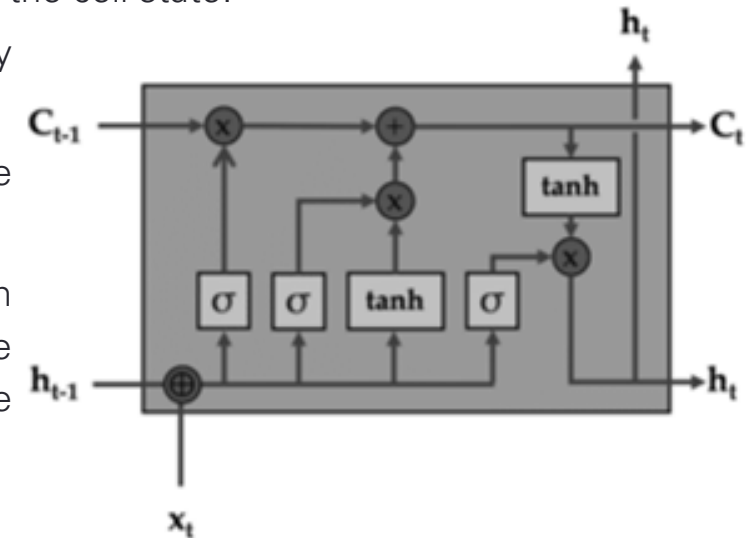
I. INTRODUCTION

LONG SHORT-TERM MEMORY (LSTM)

Long Short-Term Memory (LSTM) networks are an advanced type of RNN designed to solve the vanishing gradient problem and better handle long-term dependencies. LSTMs introduce a more complex structure, including special gating mechanisms that control the flow of information through the network.

Key Components

1. Forget Gate: Decides which information from the previous time step should be discarded from the cell state. It outputs a value between 0 and 1, where 0 means "forget" and 1 means "keep."

2. Input Gate: Determines which new information should be stored in the cell state. It uses two functions: a sigmoid to decide which parts of the input are relevant and a tanh function to generate new candidate values for the cell state.

3. Cell State: The cell state acts as the long-term memory of the network. It is modified by the forget and input gates to carry important information across time steps.

4. Output Gate: Decides what the next hidden state (output) should be, based on the updated cell state and the current input.

The key advantage of LSTMs over traditional RNNs is their ability to capture long-term dependencies and retain information for longer periods, making them effective for time series analysis, language modeling, and other tasks that require memory across many time steps.

I. INTRODUCTION

GATED RECURRENT UNITS (GRU)

Gated Recurrent Units (GRUs) are another variation of RNNs designed to address the limitations of traditional RNNs and improve upon them, similar to LSTMs. GRUs, however, are structurally simpler than LSTMs, combining the forget and input gates into a single update gate. Despite their simplicity, GRUs often perform comparably to LSTMs in many tasks and are computationally more efficient.

Key Components

1. Update Gate: This gate combines the roles of the forget and input gates in LSTM. It decides how much of the previous hidden state should be carried forward and how much of the current input should be incorporated. The update gate is a sigmoid function, outputting a value between 0 and 1.

2. Reset Gate: The reset gate determines how much of the past information should be discarded when computing the current hidden state. It is also a sigmoid function, producing values between 0 and 1.In GRUs, the memory of past inputs is managed by the update gate, and the reset gate determines how much information from the past should influence the current hidden state. The absence of a separate cell state (like in LSTMs) reduces the model's complexity and computational requirements, making GRUs more efficient for certain applications.

Despite their simpler architecture, GRUs can still capture long-range dependencies and tend to perform well when training time or computational resources are limited.
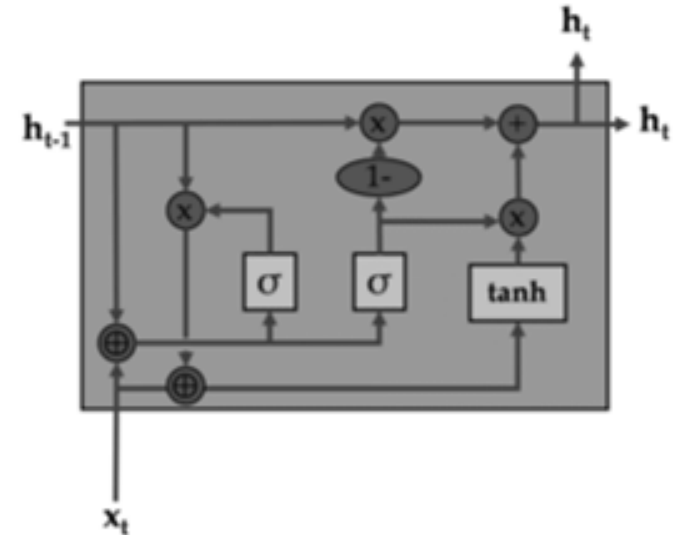
II. LSTM MODEL

II. LSTM MODEL

COMPANY STOCK PRICE HISTORY

The stock price is inherently highly unpredictable and subject to significant fluctuations. While the overall trend may show a gradual upward movement over time, the path is rarely linear. Stock prices often exhibit frequent highs and lows, which create a volatile and noisy environment. These fluctuations are influenced by a wide variety of factors, including market sentiment, economic conditions, company performance, geopolitical events, and investor behavior, among others.

Moreover, stock price movements are not driven by a single factor but rather respond to multiple inputs simultaneously. These inputs include:

• Historical prices: Past price trends provide important context for future price movements, although they are not always predictive.

• Market indicators: Economic data, interest rates, inflation, and other financial metrics can heavily influence stock prices.

• Sentiment analysis: Investor sentiment, driven by news, social media, and market psychology, can cause sharp price movements in short periods.

• External factors: Political events, regulations, or global crises (e.g., pandemics) can dramatically impact stock prices, adding to the noise and unpredictability.

Because of this inherent complexity, predicting stock prices remains a challenging task. Neural networks like LSTM and GRU, which are well-suited for time series data, can help capture patterns in the price movement. However, due to the presence of noise and the influence of numerous external factors, even sophisticated models can only provide approximate predictions rather than precise forecasts.

II. LSTM MODEL

## COMPANY STOCK PRICE HISTORY

In this context, the aim of using models like LSTM and GRU is not to predict the exact future price, but rather to capture underlying trends and make reasonably accurate forecasts of price direction over short time horizons, such as predicting the next day's closing price or providing a 30-day forecast. These models can help investors make informed decisions by identifying potential patterns amidst the noise, though they cannot fully account for the unpredictable nature of the market.
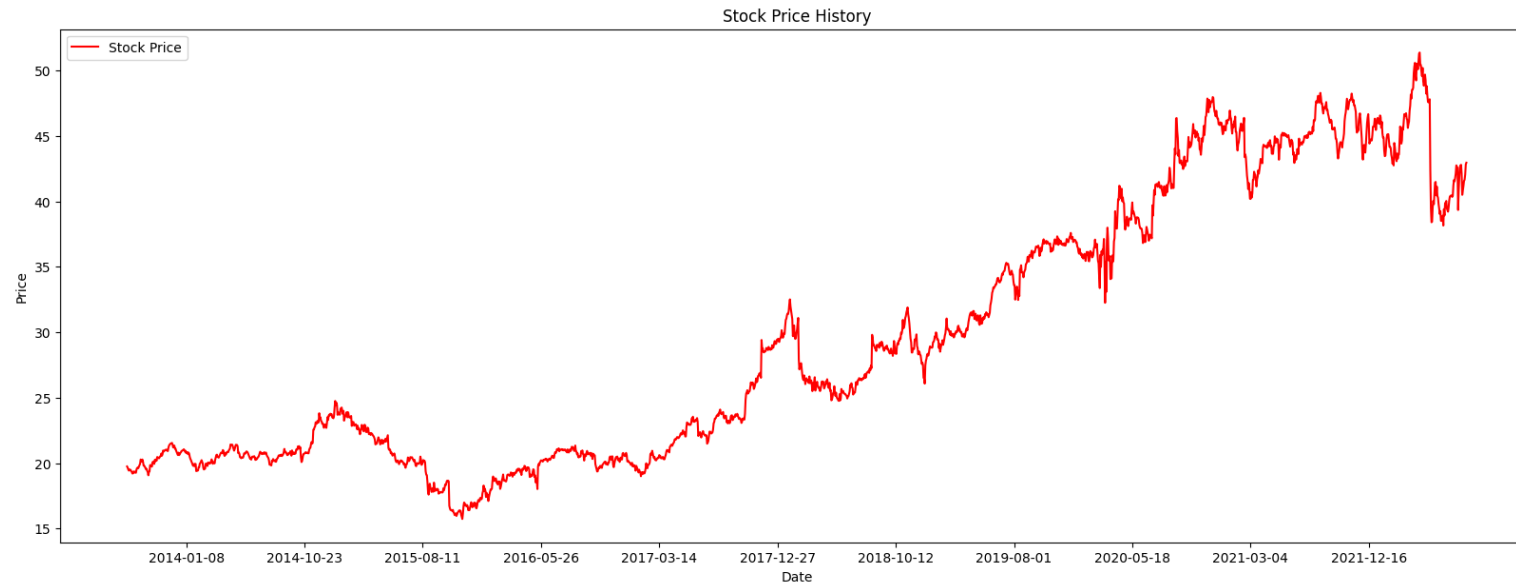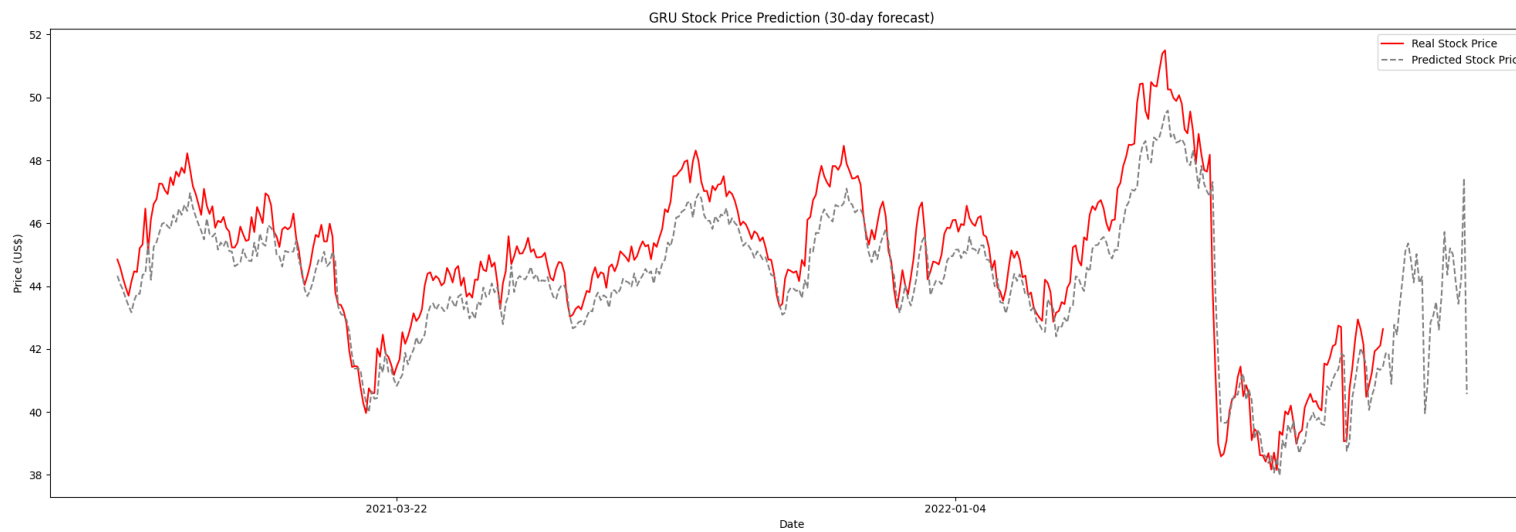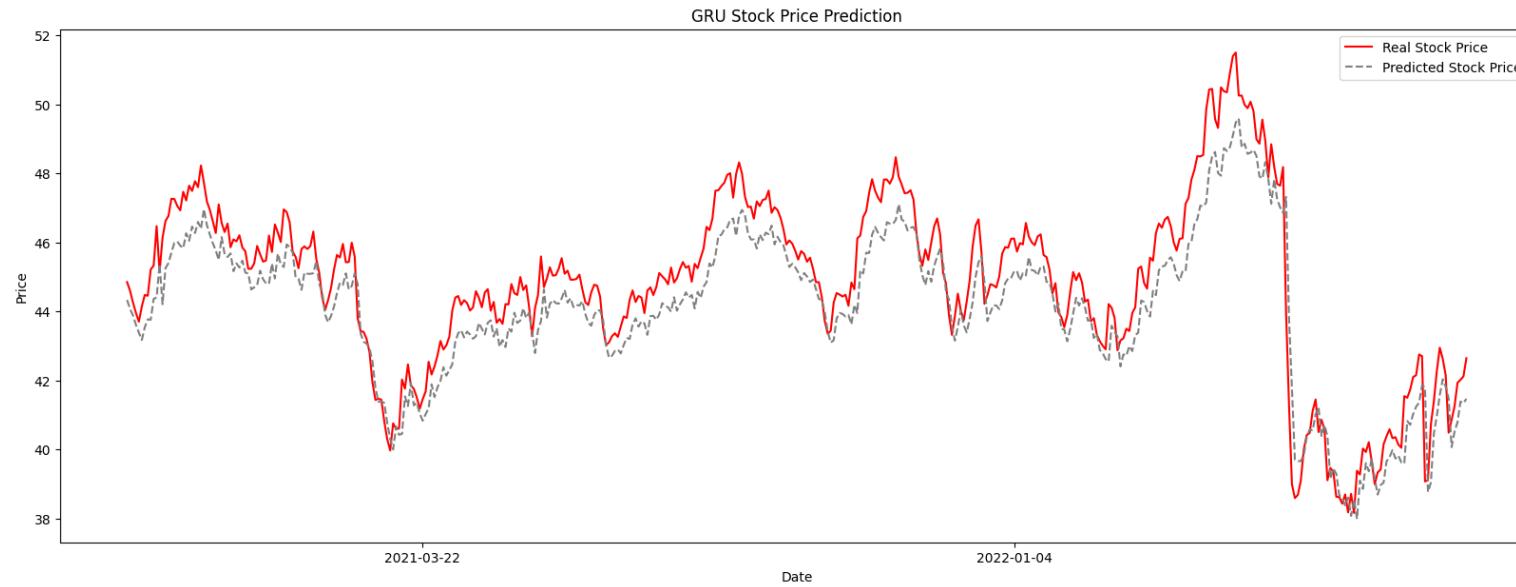
III. LSTM VS GRU MODEL

III. LSTM VS GRU MODEL – LSTM TRAINING
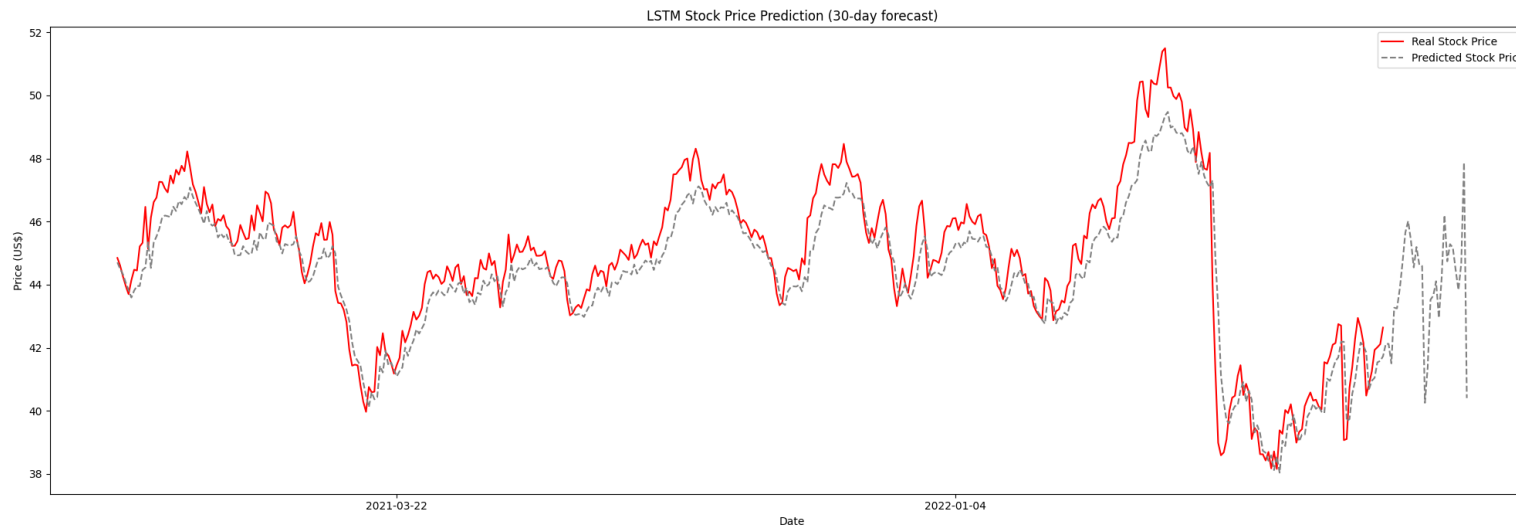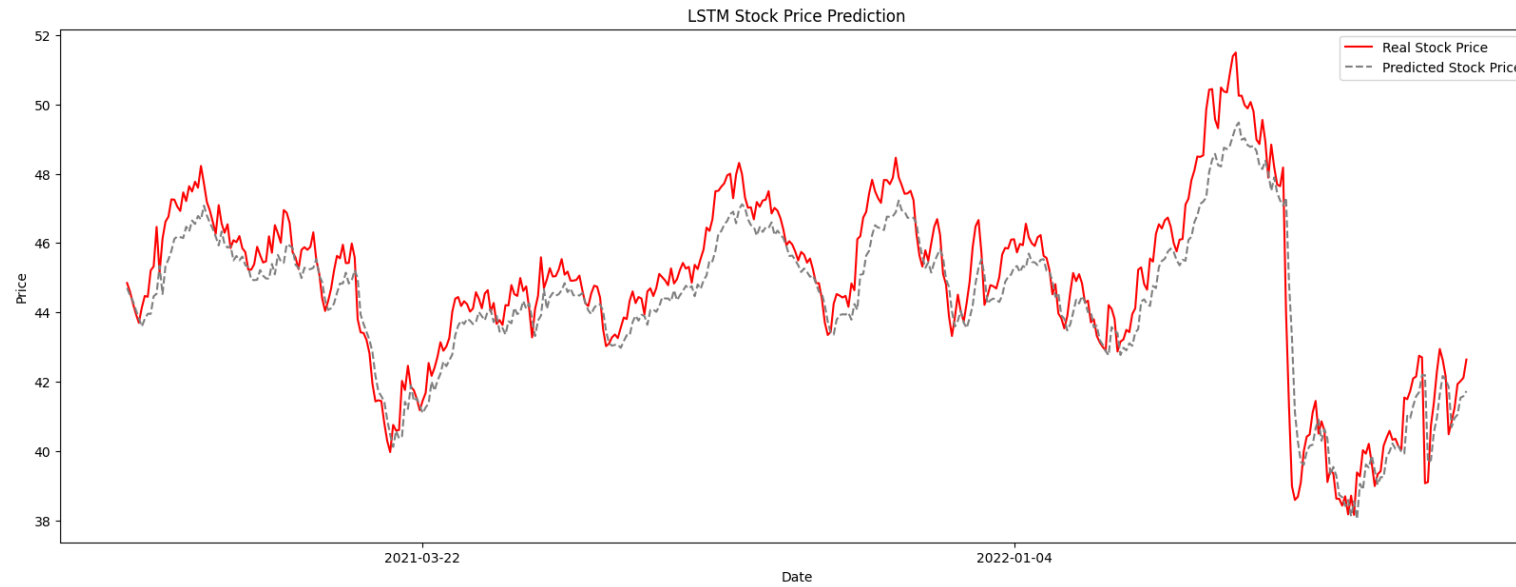
III. LSTM VS GRU MODEL – LSTM TRAINING

III. LSTM VS GRU MODEL - COMPARISON

COMPARISON RESULTS

MSE (Mean Squared Error) and MAE (Mean Absolute Error) are both common metrics used to evaluate the performance of regression models (like stock price prediction models) by measuring the difference between predicted and actual values. MSE is more sensitive to large errors because the error is squared. MAE treats all errors equally, whether they are large or small.

As seen below, LSTM achieves better metrics than GRU, which is expected, since LSTM networks are specifically designed to capture long-term dependencies more effectively.

80% of the dataset was used for model training, with the remaining 20% allocated for testing. Once the model was trained, a 30-day forecast was generated, beginning from the final day of the dataset.
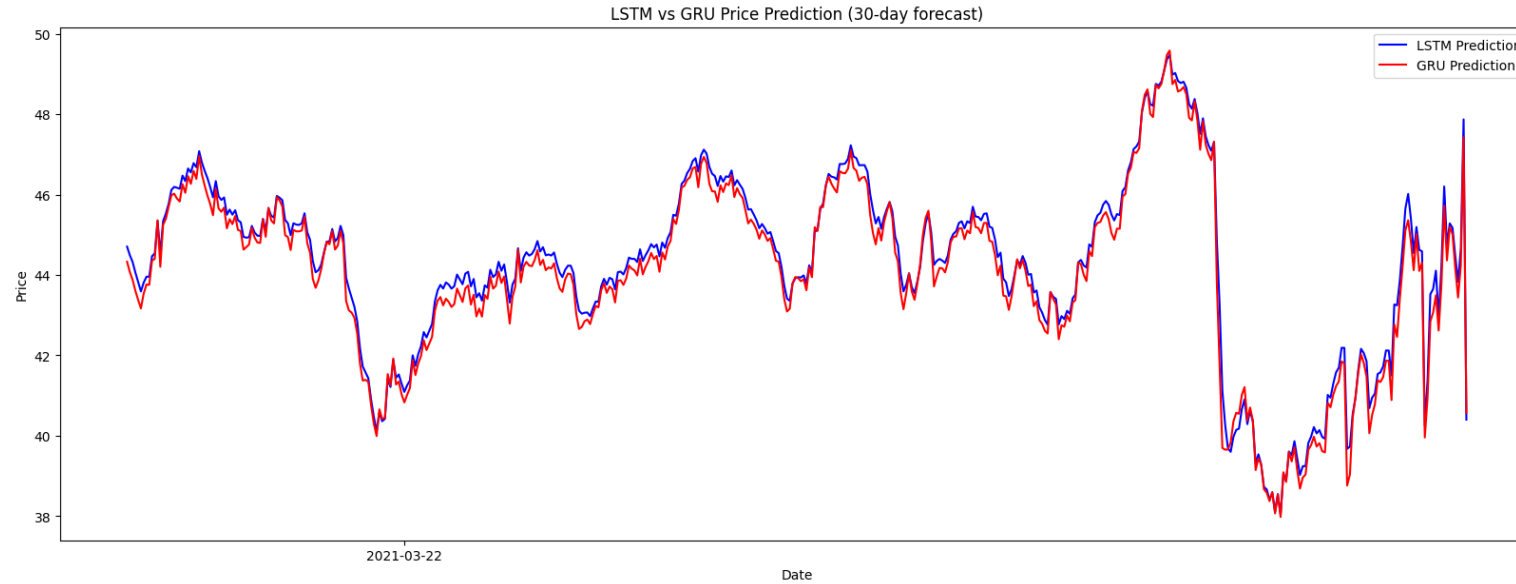
	LSTM	GRU
MSE (Mean Squared Error)	0.8035	1.0086
MAE (Mean Absolute Error)	0.7066	0.8532

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_{\text{true},i} - y_{\text{pred},i})^2$$

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_{\text{true},i} - y_{\text{pred},i}|$$

III. LSTM VS GRU MODEL

IV. CONCLUSIONS

IV. CONCLUSION

Overall, both variants achieved a strong representation of the stock prices, demonstrating their effectiveness in capturing the underlying patterns in the data. While the LSTM model provided slightly better predictive accuracy, the GRU model still delivered competitive results with faster training times and lower computational demands. Both models were able to generalize well to the stock price dynamics, making them valuable tools for time series forecasting in financial applications.

LSTM's architecture, with its specialized gating mechanisms (input, forget, and output gates), allows it to retain information over longer sequences, mitigating the vanishing gradient problem that often affects traditional RNNs and even GRU networks. This makes LSTM particularly well-suited for tasks like stock price prediction, where long-term memory and the ability to remember patterns over time are crucial.

Nonetheless, the GRU-based model achieves similar performance and offers the advantage of being computationally more efficient. While LSTM networks excel at capturing long-term dependencies due to their more complex gating mechanism, GRUs, with their simpler architecture (using only two gates instead of three), are faster to train and require fewer resources. This trade-off between performance and computational efficiency makes GRUs a strong alternative, especially in situations where model speed or resource constraints are important.