

*ITC 3154 A1:* **COGNITIVE COMPUTING**

*Instructor:* **Dr. Ioannis Bartsiokas**

*Academic Term:* **Fall 2024**

*Project Type:* **Group Project (Teams of 2)**

Dimitrios Kalligaridis (320738), Evangelos Passas (315237)

**Final Group Project:**

**Data Exploration and Machine Learning Model Optimization**

December 11th, 2024

**Main Part**

**Abstract:**

This research project focuses on financial forecasting through the use of machine learning models, investigating the accuracy we can achieve. The study utilizes a dataset obtained through the yFinance Python library, which provided daily stock information for JPMorgan Chase & Co. Two LSTM models with different architectures were analyzed, a 4-hidden layer LSTM with the Tanh activation, and a 6-hidden layer LSTM with the ReLU activation function. By employing the Bayesian hyperparameter tuning algorithm, the LSTM with 4 hidden layers and the Tanh activation function outperformed the 6-hidden layer model by a significant margin, achieving a MAE of 2.42. Demonstrating the effectiveness of LSTM models in capturing temporal patterns in stock price data and highlighting the importance of model architecture in achieving optimal performance.

**Introduction:**

In recent years, the integration of machine learning models into financial forecasting has gained considerable attention, with the goal of improving the accuracy and reliability of stock price predictions. We proceeded with utilizing the yFinance library to obtain daily market moves for JPMorgan Chase & Co from January 1st of 2010 to January 1st of 2023. The dataset contains 3,272 rows of daily stock data, with 4 columns representing the attributes: Open, High, Low, and Close. The objective of our analysis was to evaluate how accurately we could predict stock prices for the test dataset, while additionally aiming to assess its performance in forecasting current market data, such as the closing price for a 60 day period in 2024. The findings and accuracy of this model could provide valuable insights for financial institutions, supporting investment strategies, improving decision-making processes, and enabling more efficient risk assessments.

**Exploratory Data Analysis (EDA):**

To begin, it's important to highlight that the data we analyzed is a time series data set spanning 13 years of data, containing four key features: Open, High, Low, and Close, derived from Yahoo Finance. Upon calculating the descriptive statistics for all our features, it became evident, as expected, that the data points for a single day were very close to each other. However, we decided to retain them in our model training.

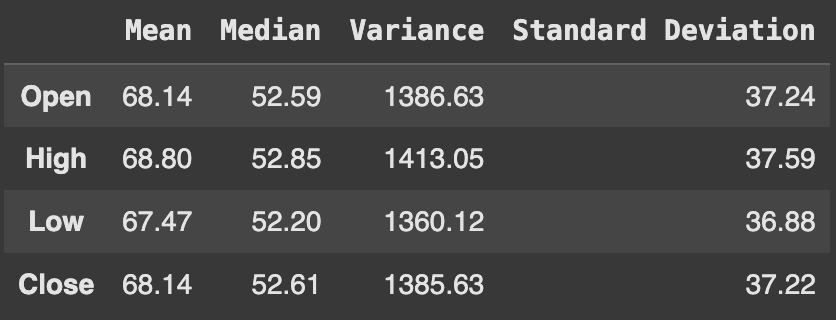


Figure 1. Table presenting descriptive statistics for the dataset's features

From the table above, we observe that for most features, the mean, median, variance, and standard deviation are relatively close in value. The most notable difference is seen in the "High" feature of the dataset, which, as expected, exhibits higher values across all its descriptive statistics.

Below is a line plot visualizing the trend of JPMorgan's closing stock price from 2010 to 2023, measured in USD. The graph clearly depicts an upward trend over this period, indicating strong momentum, with a 336.15% return on investment from the initial price to the final price in the dataset.



Figure 2. JPMorgan Chase & Co closing price trend

To further explore our dataset, it is essential to visualize the distribution of the features. We selected the closing price and the high price, as displaying all four would be redundant due to their similar values. The visualization below illustrates the distribution of the stock's closing and high prices, revealing positive (right) skewness. This observation is further supported by our descriptive statistics, which show the mean being greater than the median for both features, indicating asymmetry and data concentration on the left.

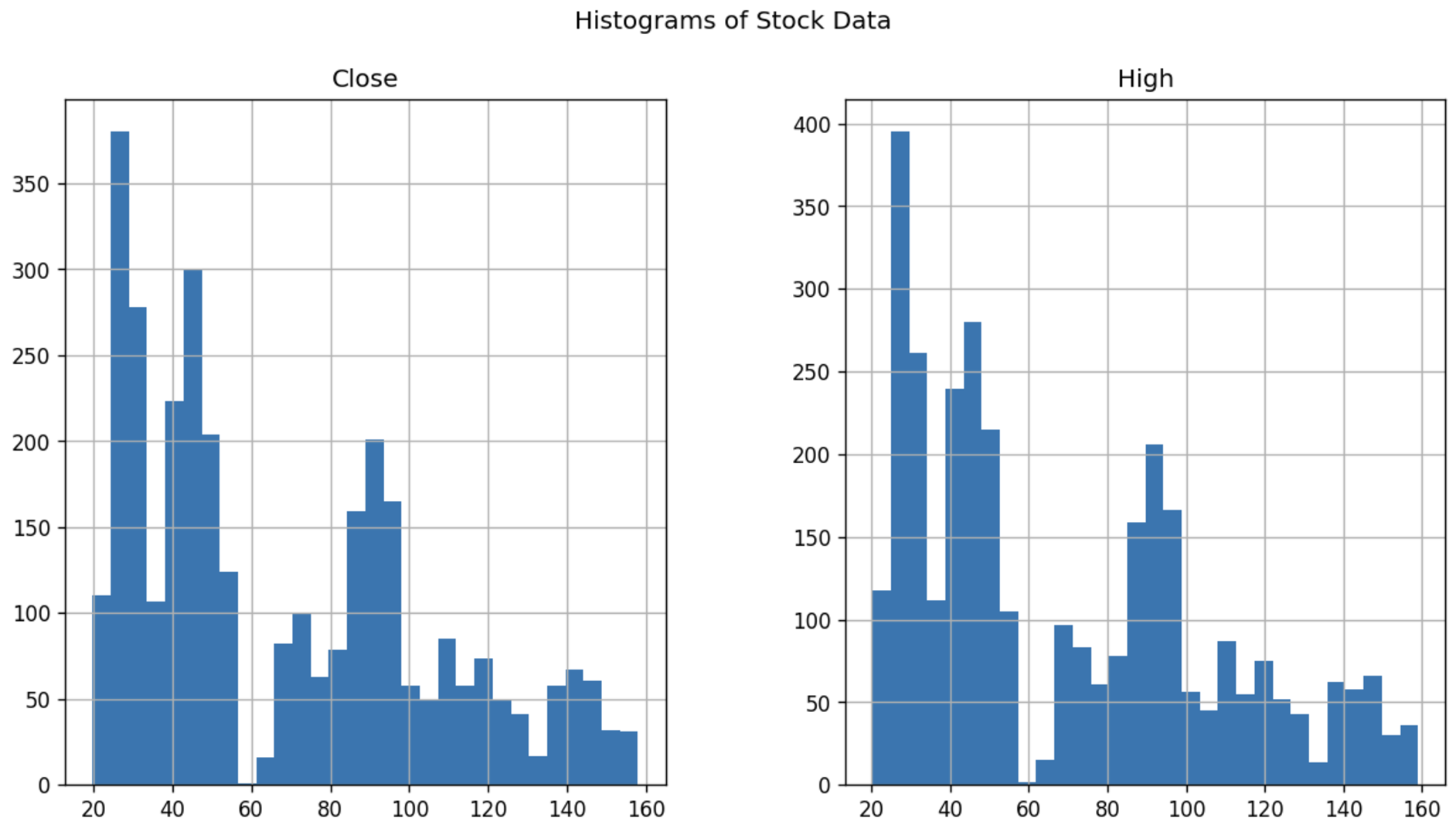


Figure 3. Histograms showing the distribution of the "Close" and "High" features

From that point, we hypothesized that the model would perform better on data prior to 2020, as the distribution of values during that period more closely aligns with the training data. While as we approach the task of predicting current market data, the model may underperform when forecasting 2024 prices which would be due to the fact that it is primarily trained on historical data from earlier years (e.g. lower values), during which the upward trend and market momentum were less pronounced.

In our dataset, we confirmed the absence of missing values by performing the necessary checks using Pandas' isnull() function on our dataframe.

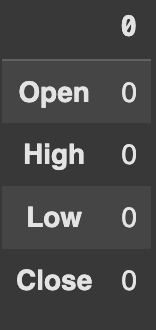


Figure 4. Verification of the absence of missing values in the dataset

In our effort to identify anomalies in the stock price data, we decomposed the time series into its trend, seasonal, and residual components using the seasonal\_decompose function from the statsmodels Python library. As expected, during 2020 caused by the COVID-19 pandemic, the market volatility increased resulting in an anomaly within the dataset. However, we decided to retain this period in the training dataset, as it did not introduce a distortion that would negatively affect the model's predictive ability.

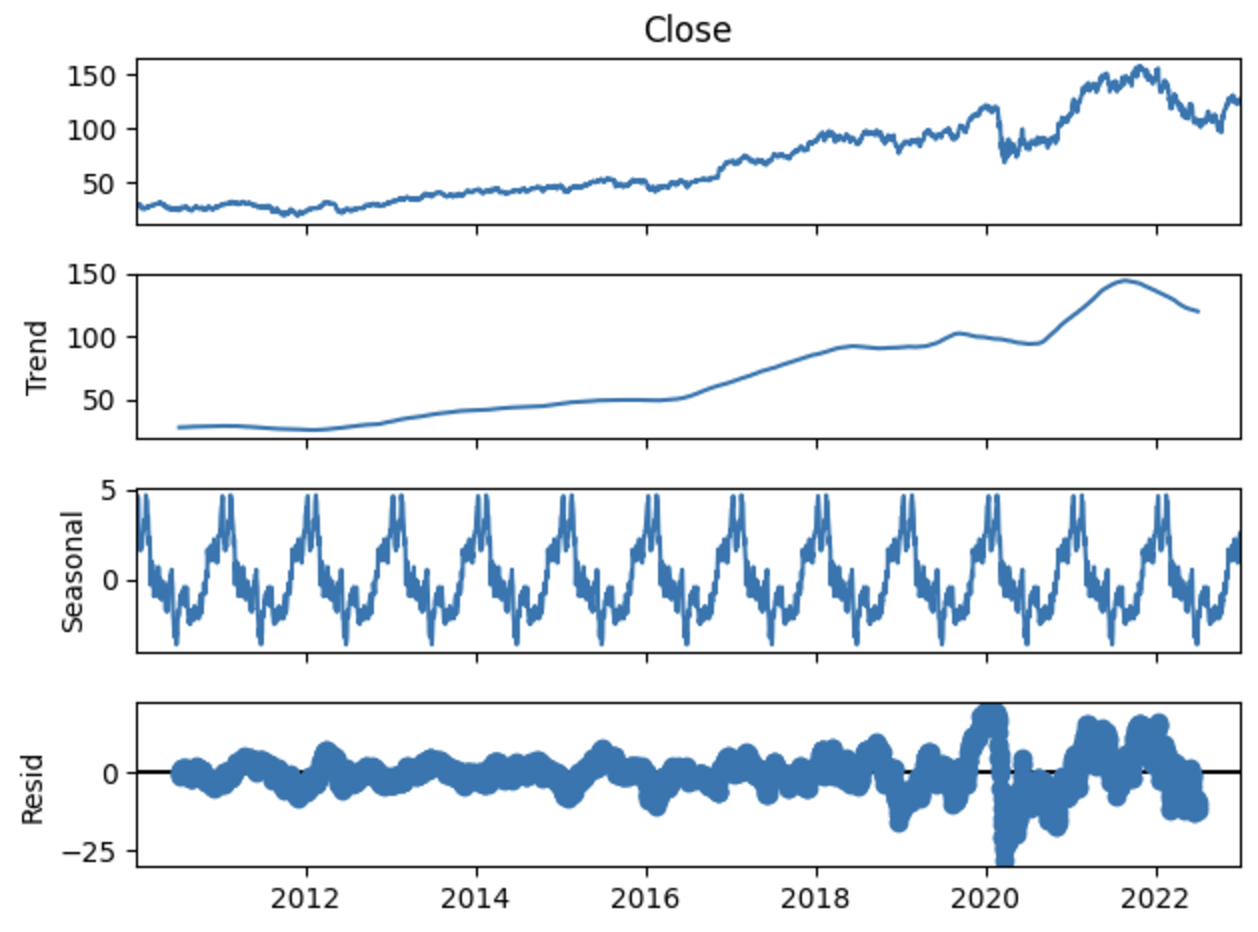


Figure 5. Time series decomposition of the "Close" feature, showing the trend, seasonal, and residual components.

To conclude the exploratory data analysis, it's important to note that the dataset obtained from Yahoo Finance is already clean and free of any missing values. Furthermore, since we are working with time series data, further detailed analysis and visualizations are limited at this stage. Additional visual insights and performance improvements will be explored during the model training and hyperparameter tuning phases. Next, we implemented two LSTM models, one with 4 hidden layers and a Tanh activation function, and another with 6 hidden layers and a ReLU activation function, for comparison, as these models are well-suited for time series forecasting.

**Data Cleaning and Transformation:**

For both LSTM models, the data transformation and preprocessing steps were largely consistent. Specifically, we applied scaling to the training dataset using a Min-Max Scaler from the sklearn.preprocessing library. This function scaled our stock prices to a defined range between 0 and 1, a crucial step as models like LSTM are highly sensitive to the magnitude of input values. By scaling the dataset, we ensure that each feature contributes equally to the model, preventing numerical instability during training, ensuring consistent scale.

The next step in our preprocessing was feature engineering for sequential data. We applied a sliding window approach to transform the scaled data into a structure with 60 timesteps and 1 output suitable for LSTM training. This step was crucial because LSTMs, being a type of RNN, excel at capturing temporal dependencies in sequential data by learning from past observations. This approach was informed by relevant research found on Kaggle, specifically a notebook detailing the process of creating the necessary data structure for LSTM models to interpret effectively (<https://www.kaggle.com/code/nafisur/intro-to-recurrent-neural-networks-using-lstms>).

Finally, we used the reshape function from the NumPy library to format the data into the 3D structure required by LSTM models: samples, timesteps, and features, as this step ensured that the data was properly prepared for training.

**Model Selection and Training:**

After conducting thorough research, we determined that the Long Short-Term Memory (LSTM) model was the most suitable approach for financial forecasting, given its ability to capture long-term dependencies in time series data. We chose to develop and experiment with two different LSTM networks: one with 4 hidden layers and the Tanh activation function, and another with 6 hidden layers and the ReLU activation function. Building on this, we decided to implement both models in our analysis to compare their performance in forecasting stock price trends.

Prior to preparing the dataset, we split the dataset into training and test dataset, one consisted of stock data before 2022 and the other after 2022 respectively. By utilizing 50 nodes in each hidden layer, a training batch size of 32 (the number of data points processed before updating the weights), and 50 epochs in total, we achieved the following results, with an average runtime of 2.5 minutes.

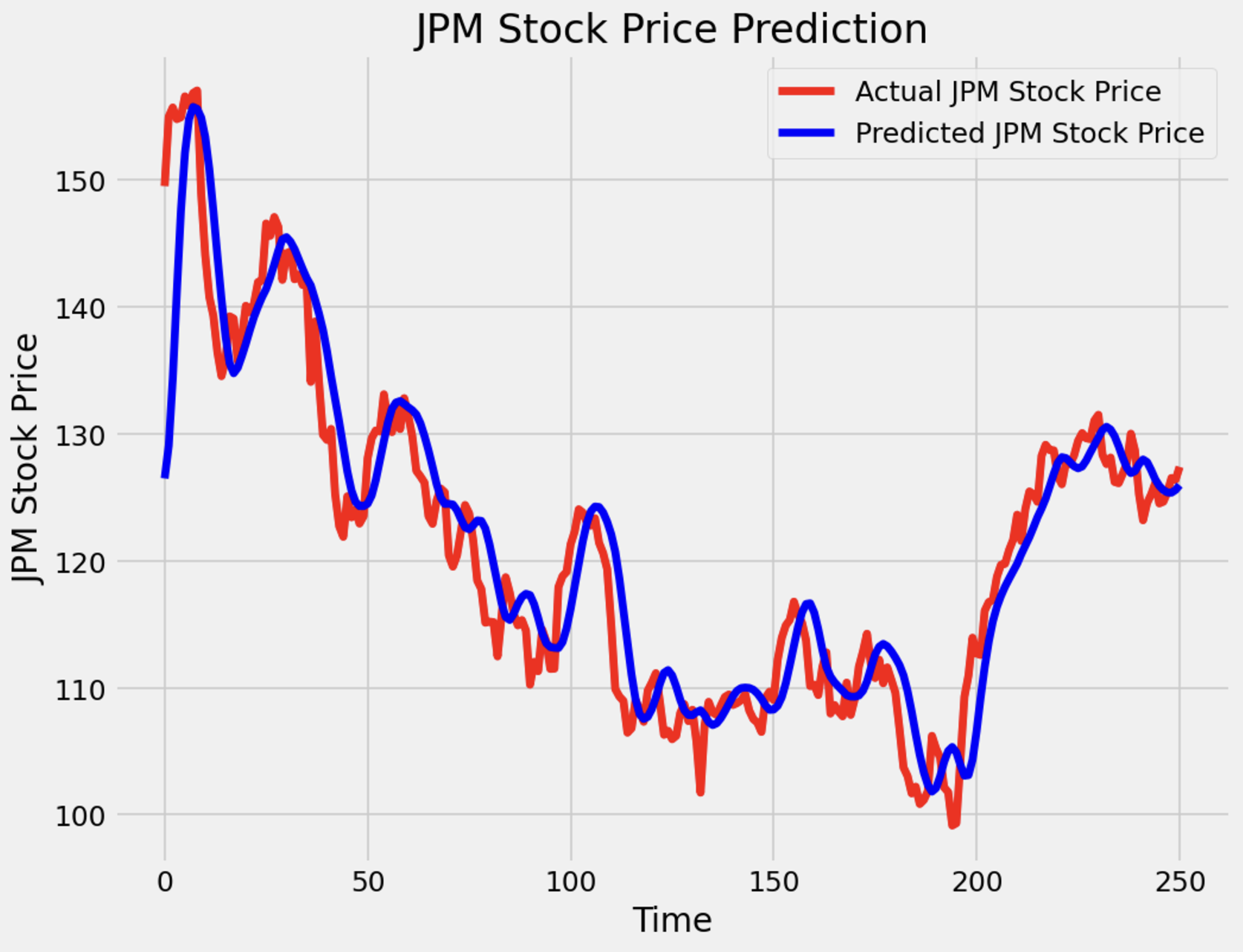


Figure 6. Performance of LSTM model with 4 hidden layers and a Tanh activation function

The initial results indicated that the first LSTM model performed exceptionally well in predicting the stock prices of the test dataset, with the errors being within an incredibly low range. The mean absolute error (MAE) of 3.24 reflects the average magnitude of the errors in the same units as the stock price, which is quite low, as well as the root mean squared error (RMSE) value of 4.58 reinforcing that the model is performing really well. Thus, we decided to test whether we could achieve a similarly high level of accuracy when predicting stock prices for the past 60 days in current market data, while extending the dataset to prices up until today (2024).

Extending the dataset from 2010 to 2024, instead of limiting it to 2023, revealed an interesting outcome.

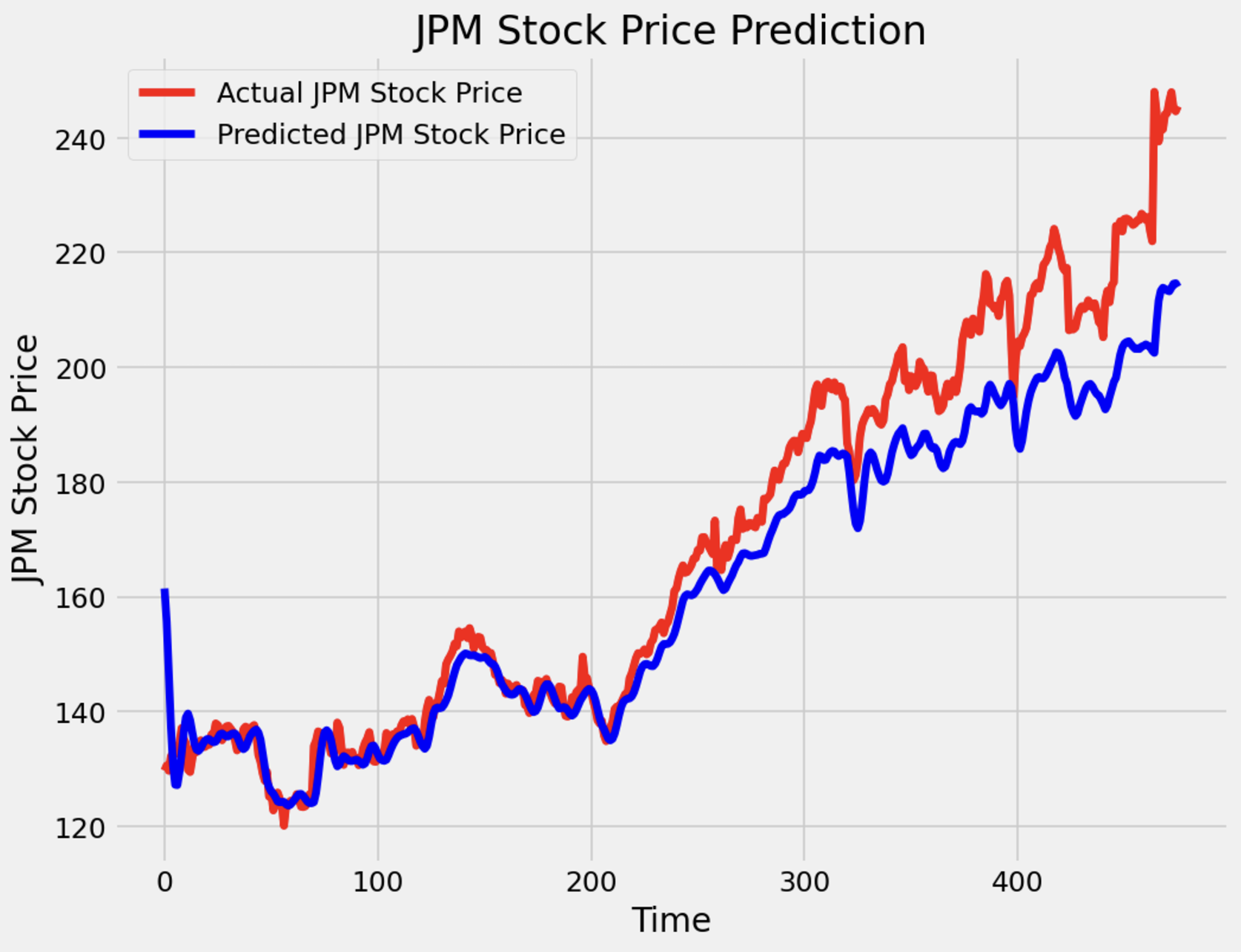


Figure 7. Performance of LSTM model with 4 hidden layers and a Tanh activation function on current market data (2024)

While the model performed well overall, it struggled to fully capture the recent upward trend in the U.S. economy, especially within the financial sector. This decline in accuracy may be attributed to the fact that, for a significant portion of the dataset, stock prices did not exhibit substantial spikes or variability in daily price changes, limiting the model's ability to generalize to more volatile trends. In this extended analysis, we observed a root mean squared error (RMSE) of 10.99 and a mean absolute error (MAE) of 7.92. While these values indicate a reasonably good performance, they fall short of the results achieved in the previous run using data from 2010 to 2023. The difference highlights the challenges of accurately modeling a market that is currently in a clear upward trend.

The transition to the second LSTM model, which utilized 6 hidden layers and a ReLU activation function, required adjustments to the code and retraining the model with the original dataset sourced from Yahoo Finance. During this process, the unoptimized parameters were kept similar to those used in the 4-hidden layer Tanh activation function LSTM model.

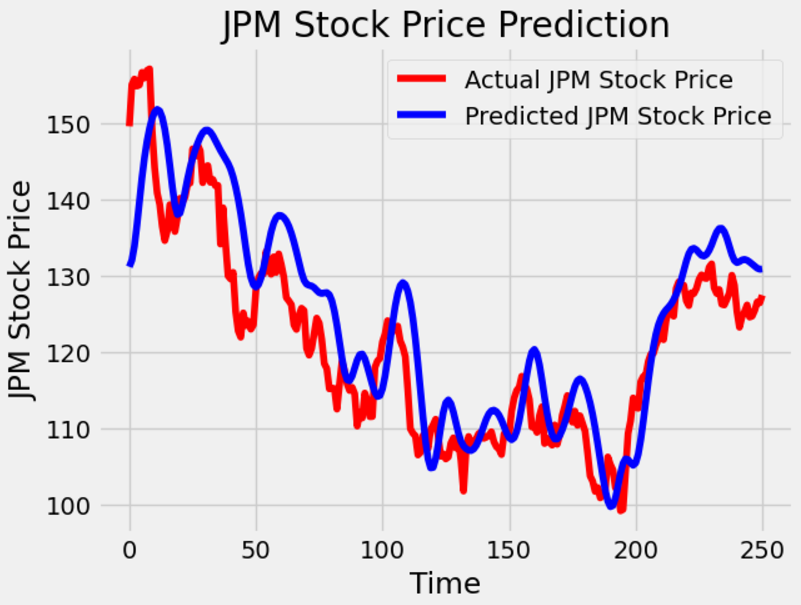


Figure 8. Performance of LSTM model with 6 hidden layers and a ReLU activation function

With this model, we achieved a strong initial performance, completing the unoptimized version in approximately 3.5 minutes. While its predicted accuracy did not surpass that of the initial 4 hidden layers LSTM with the Tanh activation function, it was still deemed satisfactory. The calculated root mean squared error (RMSE) and mean absolute error (MAE) were 7.02 and 5.56, respectively.

Concluding our initial model training before applying hyperparameter tuning algorithms, we observed that the Tanh activation function with 4 hidden layers outperformed the ReLU-based LSTM model, potentially due to the vanishing gradient problem associated with ReLU activation function.

**Hyperparameter Tuning:**

To perform hyperparameter tuning on our already initiated Long Short-Term Memory (LSTM) networks, we conducted the necessary research and concluded that we should first utilize Bayesian hyperparameter optimization. It works in an efficient way that “reduces the search time and improves the model’s performance by finding a better set of hyperparameters” (Saxena, 2024), by performing intelligent choices based on previous outcomes. The hyperparameters we selected for optimization included the number of hidden layer units, batch size, number of epochs, learning rate, and dropout rate, as these factors have a substantial impact on the performance and efficacy of the models.

Starting with the LSTM model consisting of 4 hidden layers and the Tanh activation function, the Bayesian optimization algorithm conducted 20 trial runs to identify the optimal parameters. Through this process, the optimal hyperparameters were determined to be 200 units per hidden layer, a dropout rate of 0.3, and a learning rate of approximately 0.0095, while the ideal number of epochs and batch size were confirmed to be 50 and 32, respectively. Upon training the model and visualizing the results, it became evident that the predicted stock prices aligned more closely with the actual values, with only minimal discrepancies. This improvement in accuracy demonstrated that the Bayesian optimization effectively fine-tuned the model, enhancing its ability to capture temporal patterns in the data.

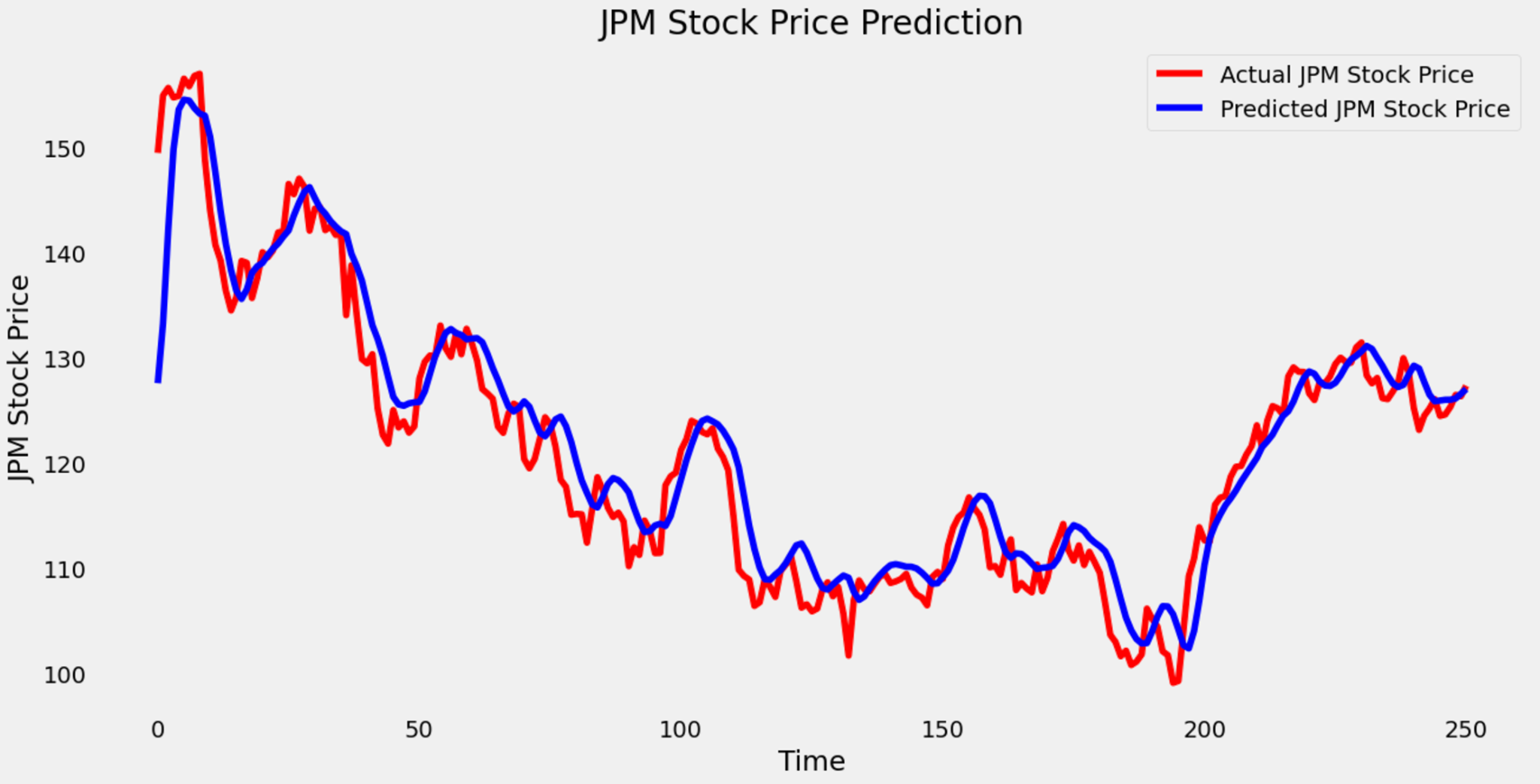


Figure 9. Performance of the Bayesian-optimized LSTM model with 4 hidden layers and a Tanh activation function

After calculating the root mean squared error (RMSE) and mean absolute error (MAE) for comparison with the previous unoptimized 4 hidden layer model, the scores showed significant improvements, reaching new lows of 3.49 and 2.42, respectively, down from the earlier scores of 4.58 and 3.24.

Moving on to the Bayesian hyperparameter optimization for the LSTM model with 6 hidden layers and the ReLU activation function, the algorithm determined that the ideal hyperparameters were similar to those of the 4 hidden layer model with the Tanh activation function. These included 200 nodes per hidden layer, a dropout rate of 0.2, a learning rate of approximately 0.004, and unchanged batch size and epoch values of 32 and 50, respectively.

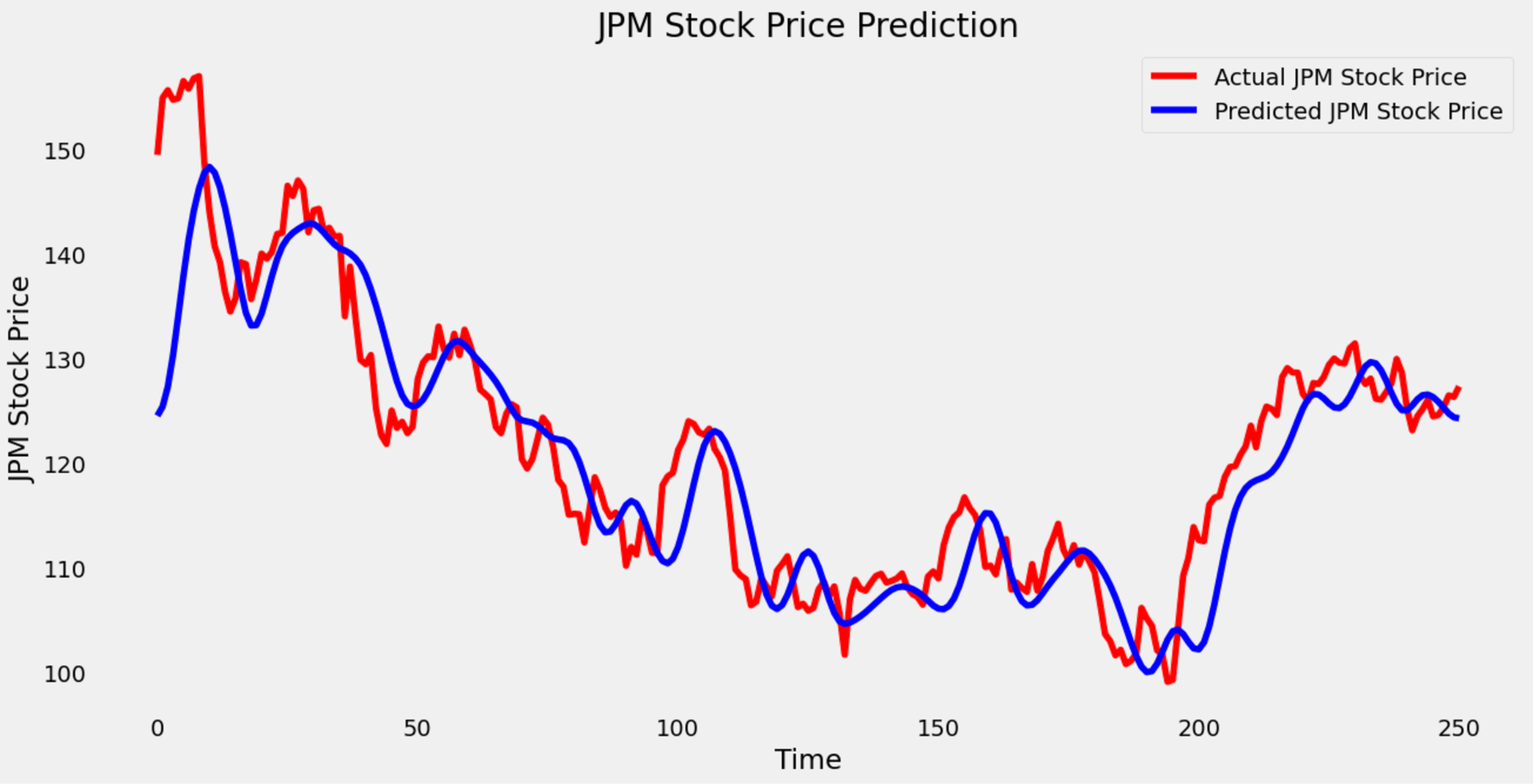


Figure 10. Performance of the Bayesian-optimized LSTM model with 6 hidden layers and a ReLU activation function

After training the now optimized model, it demonstrated significantly improved accuracy compared to the unoptimized version, achieving an root mean squared error (RMSE) of 4.32 and an mean absolute error (MAE) of 3.58, notably lower than the previous scores of 7.02 and 5.56. However, these results still fell short of surpassing the Bayesian optimized 4-hidden-layer LSTM model with the Tanh activation function, which achieved superior scores of 3.49 and 2.42.

While the optimized parameters obtained through Bayesian optimization significantly improved the performance of both LSTM models, we explored whether further refinement was possible. We decided to attempt employing the Grid Search Cross Validation algorithm making use of, “all the combinations of the values passed in the dictionary and evaluate the model for each combination using the Cross Validation method.” This approach allows us to identify the best-performing hyperparameter configuration, which we can then use to train and test the final model.

Upon the completion of the algorithm, it determined that the optimal parameters were 100 hidden layer units, a dropout rate of 0.3 (similar to Bayesian optimization), a batch size of 64, and 100 epochs. Through the line graph of the model optimized using the Grid Search Cross Validation algorithm, it is noticeable that in some instances, the model predicts slightly higher values, as indicated by the actual stock price (red line) being positioned below the predicted stock price (blue line).



Figure 11. Performance of the Grid Search CV LSTM model with 4 hidden layers and a Tanh activation function

Comparing the root mean squared error (RMSE) and mean absolute error (MAE) the values came to 3.83 and 2.72 respectively. Coming out just slightly higher from the model optimized using the Bayesian approach which marked a RMSE of 3.49 and a MAE of 2.42.

Proceeding then to applying the Grid Search Cross Validation algorithm to the 6-hidden-layer LSTM model with the ReLU activation function, the final trial identified the optimal hyperparameters as 100 units per hidden layer, a dropout rate of 0.2, and a batch size of 64, while reaffirming that the ideal number of epochs remains at 50.

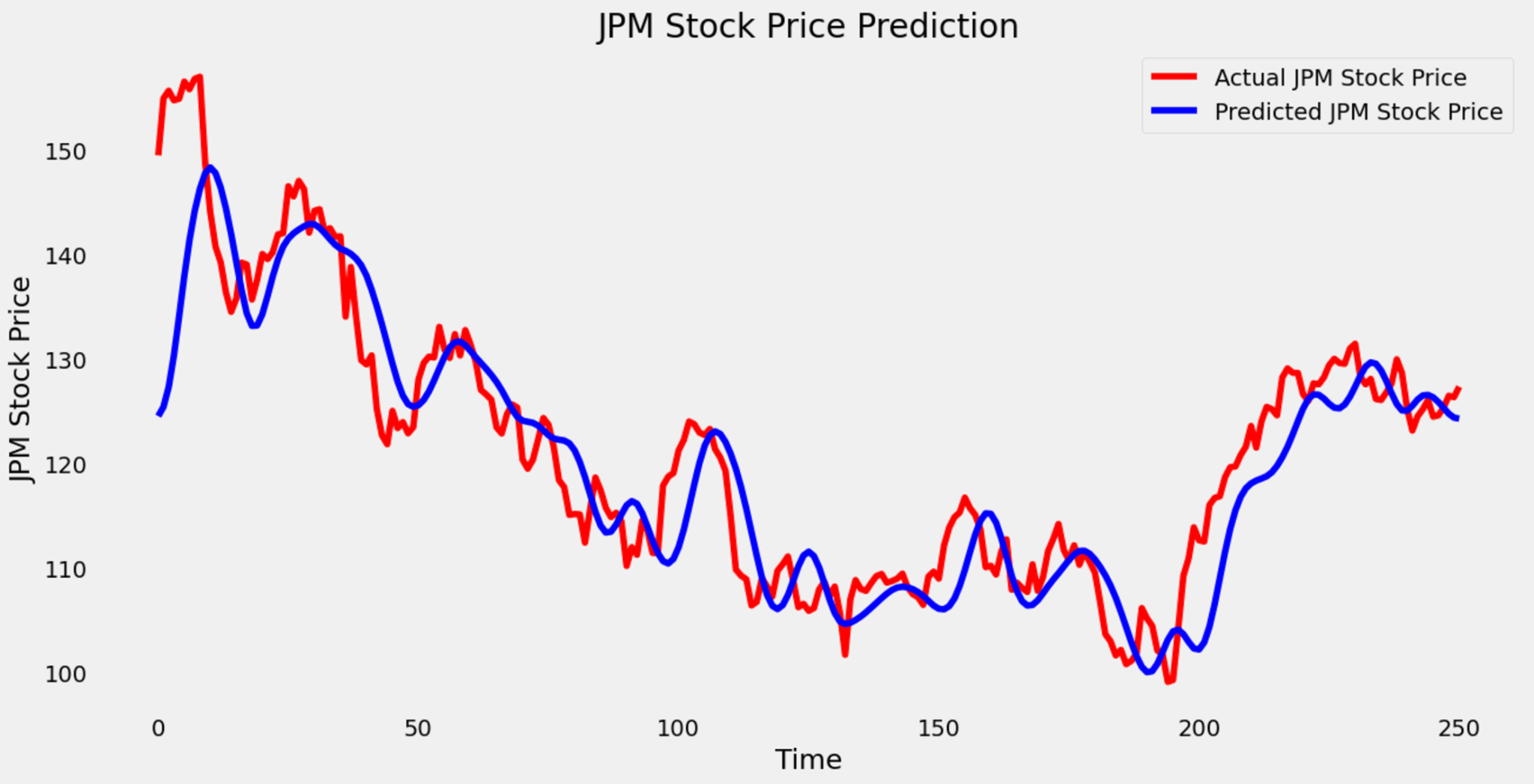


Figure 12. Performance of the Grid Search CV LSTM model with 6 hidden layers and a ReLU activation function

The Grid Search Cross Validation algorithm optimized the 6-hidden-layer LSTM model with the ReLU activation and demonstrated significant performance improvements over the unoptimized version, achieving an RMSE of 5.93 and a MAE of 4.10 compared to the initial RMSE of 7.02 and MAE of 5.56. However, it still fell short of the performance achieved through Bayesian optimization, which yielded a RMSE of 4.32 and an MAE of 3.58. These results highlight the relative effectiveness of Bayesian optimization in fine-tuning hyperparameters for enhanced predictive accuracy.

To conclude, both optimization methods demonstrated the importance of tuning hidden layer units, dropout rates, and other hyperparameters for LSTM models as it significantly improves the accuracy which is evident by the reduced RMSE and MAE. Bayesian optimization produced the best results, showcasing its efficiency in minimizing error rates and improving model predictions with actual stock prices. It is evident that through this approach the 4 hidden layers with the Tanh activation function LSTM model achieved a 23.80% decrease for the root mean squared error (RMSE) and a 25.31% decrease in the mean absolute error (MAE). While the 6 hidden layers LSTM with the ReLU activation function achieved a 38.46% decrease in RMSE and 35.61% decrease in MAE scores.

Lastly, the Grid Search Cross Validation algorithm, while slightly less effective than the Bayesian optimization, validated the consistency of key hyperparameter choices and highlighted alternative configurations.

**Discussion and Insights:**

After extensive experimentation, it is evident that the LSTM model effectively captures the temporal dependencies in stock price movements, demonstrating its strength in handling time-series data. Hyperparameter tuning had a significant impact on the model's performance, yielding improved accuracy and reducing prediction errors. The LSTM's success can be attributed to effective feature engineering and its robust architecture, which is well-suited for sequential data. Limitations that must be acknowledged are its reliance solely on historical data and its inability to account for external factors such as news events, macroeconomic trends, or geopolitical influences, which can heavily impact stock prices.

To enhance the model’s predictive power, future improvements could include integrating sentiment analysis of financial news and social media to capture market sentiment. Additionally, experimenting with transformer-based models, known for their scalability and superior handling of long-term dependencies, could provide a more advanced and comprehensive framework for stock price prediction. These advancements could enable the model to incorporate a broader context, ultimately leading to more accurate and reliable forecasts.

**References:**

Provide a complete list of references used in your project, including:

* <https://pypi.org/project/yfinance/>
* <https://www.comet.com/site/blog/hyperparameter-tuning-with-bayesian-optimization/>
* <https://www.mygreatlearning.com/blog/gridsearchcv/>

**Appendix:**

Provide a link to the Kaggle Notebook(s) that you have implemented to answer these assessment questions.

<https://www.kaggle.com/code/nafisur/intro-to-recurrent-neural-networks-using-lstms>

Code for HyperParameter Tuning utilizing Bayesian optimization and Grid Search CV generated by ChatGPT 4.0