CS 583 - Deep Learning - Tesla Stock Price Forecasting

Siddharth Paraag Nilakhe
Dept. of Data Science
Stevens Institute of Technology
Hoboken, US
snilakhe@stevens.edu

Preeti Reddy Koppolu
Dept. of Data Science
Stevens Institute of Technology
Hoboken, US
pkoppolu@stevens.edu

***Abstract*—The project aims to forecast Tesla's stock price using deep learning techniques. The main focus is on implementing and evaluating Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU), due to their effectiveness in handling time series data. The approach involves preprocessing historical stock price data, applying data normalization, and creating a time-stepped dataset to train the deep learning models. The project's results have shown promising directions in the methodology and model architecture.**

Github:

I. Introduction

Tesla Inc., renowned for its significant contributions to the electric vehicle (EV) and clean energy sectors, has seen its stock price undergo notable fluctuations, making it a subject of interest for investors and market analysts. This project explores the use of deep learning techniques, specifically RNNs and LSTMs, for predicting Tesla's stock price. The motivation behind this approach stems from the capabilities of these models to capture temporal dependencies and patterns in time-series data, which is critical for accurate stock price forecasting.

A. Objective

The primary objective is to develop a reliable predictive model that can forecast Tesla's stock price with a reasonable degree of accuracy, aiding in better investment decisions and market analysis.

B. Significance

Given the volatile nature of stock markets, a model that can provide accurate predictions can be invaluable for investors and analysts. Additionally, the success of this model can set a precedent for applying similar techniques to other stocks or financial time series data.

II. Data

The dataset has been collected from TIINGO [1] through API GET call. The dataset contains records from 01-01-2018 to current date.

Shape of train set: (816, 1)

Shape of test set: (440, 1)

	symbol	date	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash	splitFactor
0	TSLA	2018-11-16 00:00:00+00:00	354.31	355.70	345.12	345.190	7206191	23.620667	23.713333	23.008000	23.012667	108092865	0.0	1.0
1	TSLA	2018-11-19 00:00:00+00:00	353.47	366.75	352.88	356.340	9708871	23.564667	24.450000	23.525333	23.756000	145633065	0.0	1.0
2	TSLA	2018-11-20 00:00:00+00:00	347.49	349.80	333.55	341.750	8004709	23.166000	23.320000	22.236667	22.783333	120070635	0.0	1.0
3	TSLA	2018-11-21 00:00:00+00:00	338.19	353.10	337.40	352.000	4686808	22.546000	23.540000	22.493333	23.466667	70302120	0.0	1.0
4	TSLA	2018-11-23 00:00:00+00:00	325.83	337.50	325.55	334.345	4202642	21.722000	22.500000	21.703333	22.289667	63039630	0.0	1.0

Fig. 1. Dataset Overview

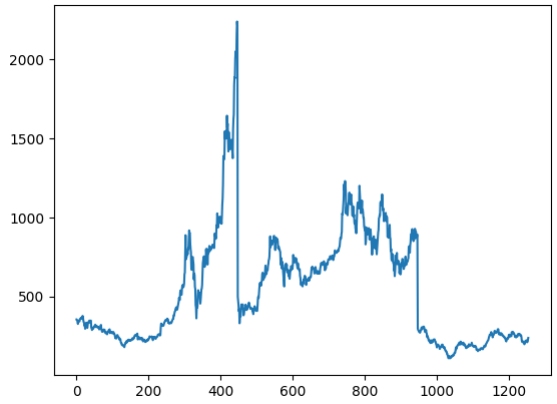

Fig. 2. Time Series Plot

III. Contributions

A. Siddharth Paraag Nilakhe

- Data cleaning and preparation.
- RNN and LSTM model training.
- Hyperparameter tuning for RNN and LSTM.

B. Preeti Reddy Koppolu

- Data cleaning and preparation.
- Stacked RNN and GRU model training.
- Hyperparameter tuning for stacked RNN and GRU.

Report and presentation were prepared in a collaborative fashion with equal contribution.

IV. Tools and Technologies

Software/Library Packages:

Pandas Datareader [2], Pandas [3], Numpy [4], Matplotlib [5], Scikit-learn [6], TensorFlow [7], Keras[8]

Hardware Requirements:

Google Colab: Offers a cloud-based solution with robust computational resources, including high-end GPUs and TPUs. Ideal for handling larger datasets or more complex models without the need for personal high-end hardware. [9]

V. Data Preprocessing

- The dataset is cleaned, and the closing prices are extracted.
- Recognizing the sensitivity of LSTM models to data scales, the project employed MinMaxScaler for normalizing the stock price data. This scaling transformed the data to a specified range (0 to 1), ensuring that the LSTM model receives appropriately scaled inputs.
- A rolling window approach is utilized to transform the series data into a supervised learning problem. For each entry, the model will predict the next time step's closing price based on the previous 100 days' prices.

VI. Methodology

A. Dataset Preparation

A critical aspect of the project is the preparation of the dataset for time-series forecasting. The data is split into training and testing sets, with 65% of the data allocated for training. To capture temporal dependencies, a rolling window approach is adopted, creating sequences of 100 days' worth of data to predict the next day's price.

B. Model Development and Training

The project delves into the utilization of three prominent types of Recurrent Neural Networks (RNNs) - Simple RNNs, Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) - each uniquely suited to handle sequences and time-series data. TensorFlow and Keras, renowned for their versatility and ease of use in deep learning applications, serve as the primary tools for building and training these models.

- **Recurrent Neural Network:** The simplest form of RNNs, these networks are characterized by their ability to maintain a memory of previous inputs through hidden layers. In this project, SimpleRNN layers are used as a baseline to capture the sequential nature of stock price movements. However, due to their relatively basic structure, Simple RNNs may struggle with long-term dependencies commonly present in financial time series.
- **Long Short-Term Memory:** LSTMs are an advanced variant of RNNs designed specifically to overcome the limitations of Simple RNNs in handling long-term dependencies. They achieve this through a complex architecture involving gates (input, output, and forget gates), which regulate the flow of information. These gates effectively allow the network to retain or discard information across long sequences, making LSTMs particularly well-suited for predicting stock prices where past information is crucial for forecasting future trends.
- **Gated Recurrent Units:** GRUs are another variant of RNNs, similar to LSTMs but with a simpler structure. They combine the input and forget gates into a single "update gate" and also merge the cell state and hidden state. This streamlined architecture can lead to faster training times and reduced computational complexity, making GRUs an efficient alternative to LSTMs. GRUs maintain the capability to handle long-term dependencies, albeit with a less complex mechanism compared to LSTMs.

Each of these RNN types offers unique advantages for time-series analysis. Simple RNNs provide a basic yet effective approach, LSTMs offer a more sophisticated mechanism for capturing long-term dependencies, and GRUs present a balance between complexity and efficiency. In this project, the performance of these three models is compared to determine the most effective approach for forecasting Tesla's stock prices, taking into account factors like model accuracy, training time, and the complexity of financial data.

C. Model Evaluation and Forecasting

Post-training, the model's performance is evaluated using the root mean squared error (RMSE) metric, comparing the predicted prices against the actual prices in the training and testing datasets. The model is then used to forecast future stock prices, extending beyond the range of the historical data available. This step is crucial to assess the model's ability to generalize and predict unseen data accurately.

D. Visualization

Throughout the project, Matplotlib, a Python plotting library, is employed to visualize the stock prices and the model's predictions. This visual representation is not only pivotal for analyzing the model's performance but also provides intuitive insights into the trends and patterns in Tesla's stock prices.

E. Hyperparameter tuning

Hyperparameter tuning process is essential to optimize the model's performance by identifying the most effective combination of parameters. TensorFlow and Keras provide the foundation for model construction, while Scikit-learn's RandomizedSearchCV is employed for efficient hyperparameter optimization.

Parameter space:

- **units:** Varying numbers of neurons in the neural layer (20, 50, 100) to determine the optimal complexity of the model.
- **activation:** Different activation functions ('relu', 'tanh') to explore non-linear transformations within the network.
- **optimizer:** Various optimization algorithms ('adam', 'rmsprop') to find the most effective approach for minimizing loss.
- **batch_size:** Multiple batch sizes (16, 32, 64) to assess the impact on model training and convergence.

- **epochs:** Different training durations (10, 50, 100 epochs) to evaluate the effect on model learning and generalization.

VII. Experiments

A. Recurrent Neural Network

- The model consists of a SimpleRNN layer with 50 neurons followed by a Dense layer.
- The input shape is set according to the time steps (100 days).
- The model is compiled using the Adam optimizer and mean squared error loss function.

B. Stacked Recurrent Neural Network

- The model consists of three SimpleRNN layers with 50 neurons followed by a Dense layer.
- The input shape is set according to the time steps (100 days).
- The model is compiled using the Adam optimizer and mean squared error loss function.

C. Long Short-Term Memory

- The model consists of three LSTM layers with 50 neurons followed by a Dense layer.
- The input shape is set according to the time steps (100 days).
- The model is compiled using the Adam optimizer and mean squared error loss function.

D. Gated Recurrent Unit

- The model consists of three GRU layer with 50 neurons followed by a Dense layer.
- The input shape is set according to the time steps (100 days).
- The model is compiled using the Adam optimizer and mean squared error loss function.

E. Hyperparameter tuning

- Defined an model with variable hyperparameters such as units, epochs, batch size, optimizer.
- The hyperparameters are set for various ranges depending upon the model being tuned.
- RandomizedSearchCV is employed with KerasRegressor model for tuning, running a specified number of iterations (n_iter) across different hyperparameter combinations to determine the most effective model settings based on the training data.

VIII. Results

A. Recurrent Neural Network

Fig.3 shows the rolling window forecasting performed using RNN with window size of 100 days. The train mean squared error obtained through this model is 743.75 and the test mean square error is 334.35.

Fig.4 shows the model's forecast of Tesla's stock closing price for the next 30 days.

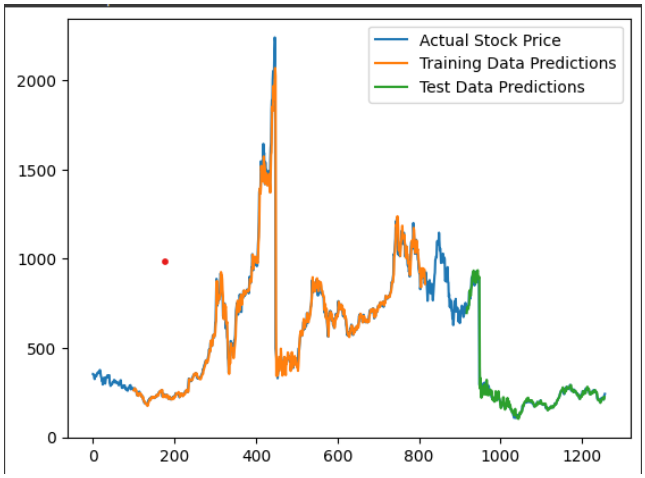

Fig. 3. Rolling window forecast - RNN

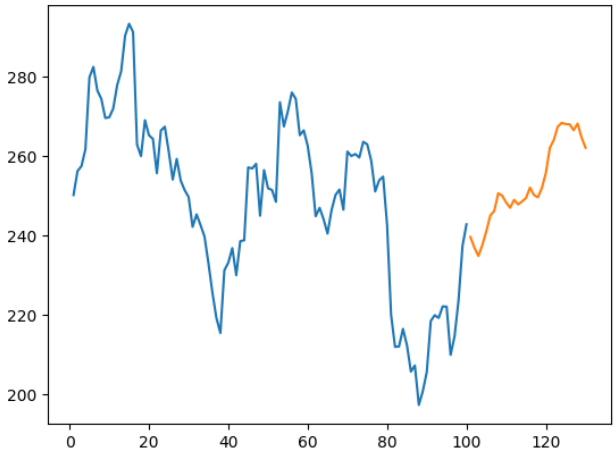

Fig. 4. Next 30 days forecast - RNN

B. Stacked Recurrent Neural Network

Fig.5 shows the rolling window forecasting performed using stacked RNN with window size of 100 days. The train mean squared error obtained through this model is 748.44 and the test mean square error is 325.81. There is an observable decrease in test error by increase in RNN layers within the model

Fig.6 shows the stacked RNN model's forecast of Tesla's stock closing price for the next 30 days.

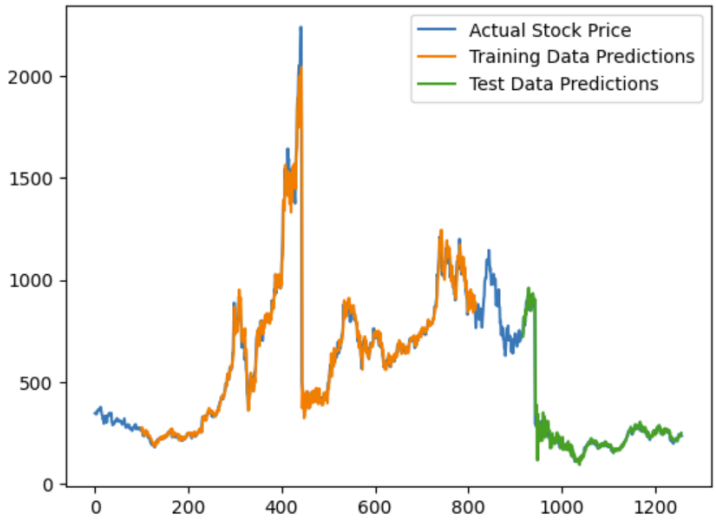

Fig. 5. Rolling window forecast - Stacked RNN

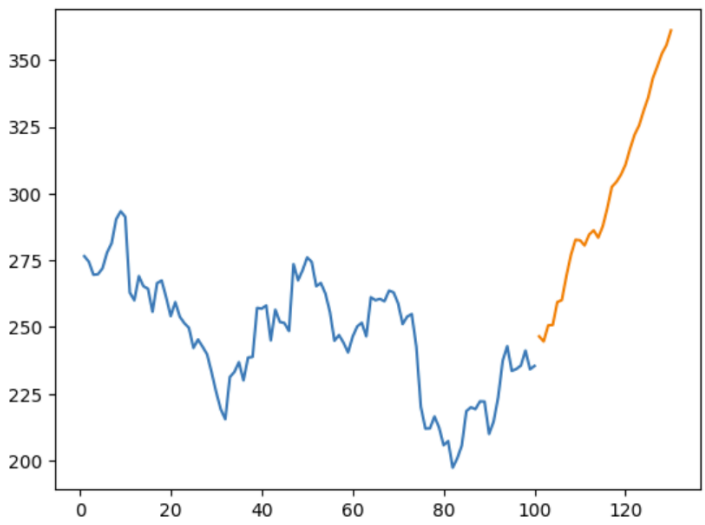

Fig. 6. Next 30 days forecast - Stacked RNN

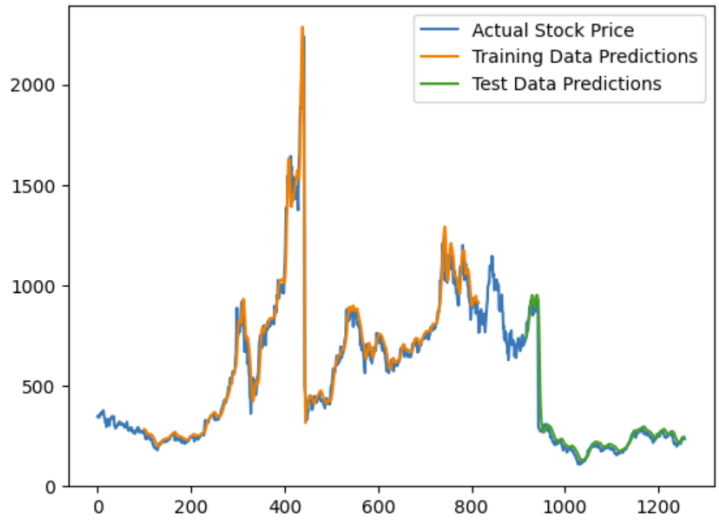

Fig. 7. Rolling window forecast - LSTM

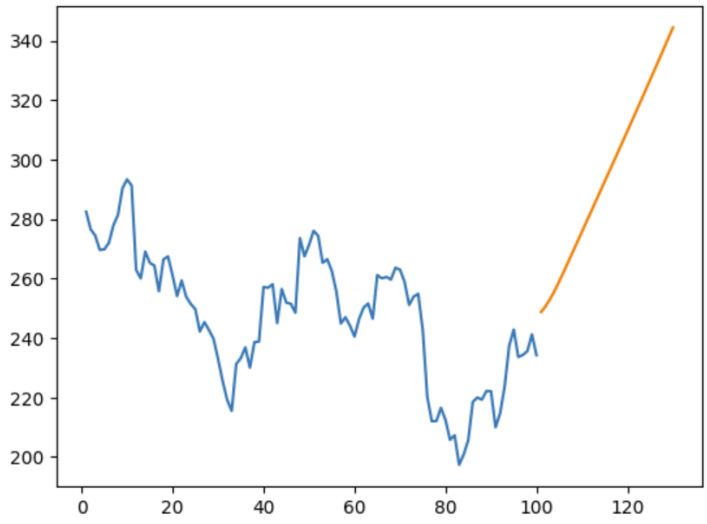

Fig. 8. Next 30 days forecast - LSTM

C. Long Short-Term Memory

Fig.7 shows the rolling window forecasting performed using LSTM with window size of 100 days. The train mean squared error obtained through this model is 764.48 and the test mean square error is 341.67. It can bee seen that LSTM does not perform any better than simple RNN. Infact, the test error is higher for LSTM compared to simple RNN model.

Fig.8 shows the stacked LSTM model's forecast of Tesla's stock closing price for the next 30 days.

D. Gated Recurrent Unit

Fig.9 shows the rolling window forecasting performed using GRU with window size of 100 days. The train mean squared error obtained through this model is 751.85 and the test mean square error is 327.99. This model performs the best compared to the previous models which can be seen through it's lowest test mean squared error amongst all the 4 models.

Fig.10 shows the GRU model's forecast of Tesla's stock closing price for the next 30 days.

E. Hyperparameter Tuning for Recurrent Neural Network

Best parameters chosen by RandomizedSearchCV are 'units': 50, 'optimizer': 'rmsprop', 'epochs': 150, 'batch_size': 64

On rerunning the model with above parameters:

Train mean squared error - 751.47

Test mean squared error - 328.38

Fig.11 shows rolling window forecast and Fig.12 shows forecast for next 30 days.

F. Hyperparameter Tuning for Long Short-Term Memory

Best parameters chosen by RandomizedSearchCV are 'units': 100, 'optimizer': 'adam', 'epochs': 150, 'batch_size': 16

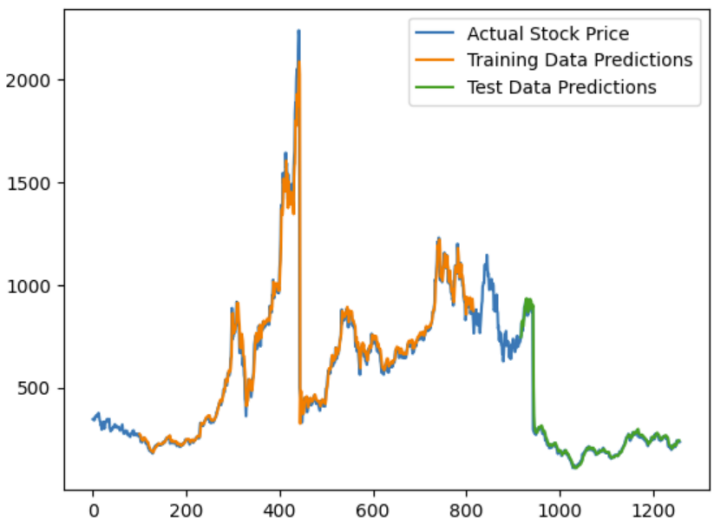

Fig. 9. Rolling window forecast - GRU

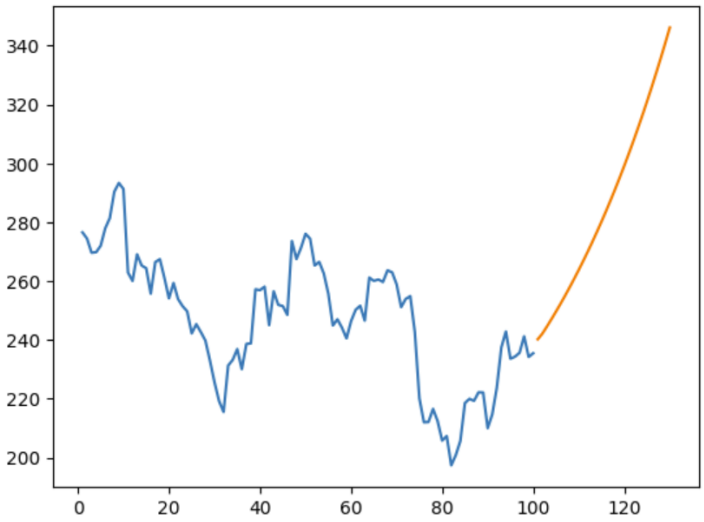

Fig. 10. Next 30 days forecast - GRU

On rerunning the model with above parameters:

Train mean squared error - 762.35

Test mean squared error - 327.32

Fig.13 shows rolling window forecast and Fig.14 shows forecast for next 30 days.

G. Hyperparameter Tuning for Gated Recurrent Unit

Best parameters chosen by RandomizedSearchCV are 'units': 50, 'optimizer': 'rmsprop', 'epochs': 100, 'batch_size': 16

On rerunning the model with above parameters:

Train mean squared error - 747.56

Test mean squared error - 323.67

Fig.15 shows rolling window forecast and Fig.16 shows forecast for next 30 days.

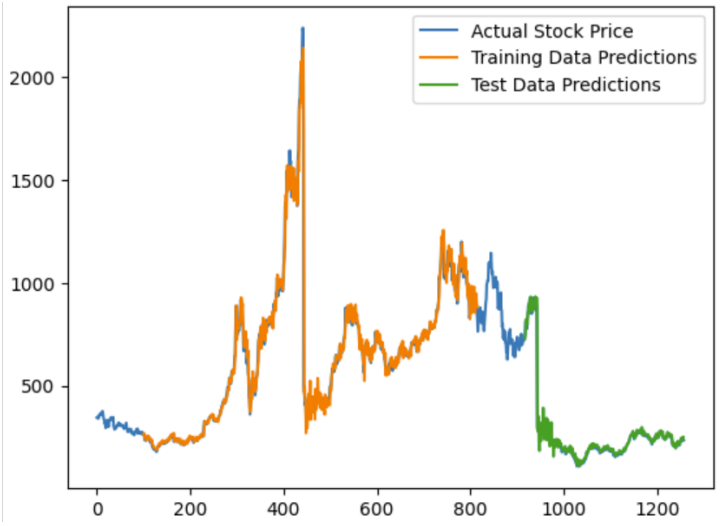

Fig. 11. Rolling window forecast with hyperparameter tuning - RNN

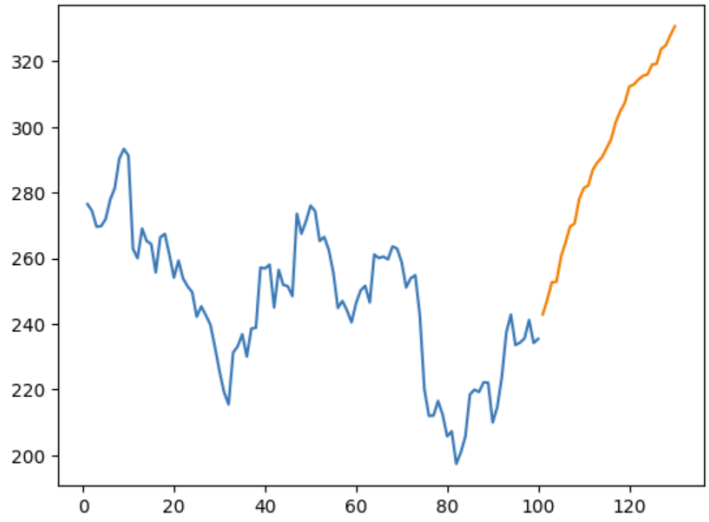

Fig. 12. Next 30 days forecast with hyperparameter tuning - RNN

IX. Conclusion

By employing and comparing various RNN architectures, including Simple RNN, LSTM, and GRU, the study provided valuable insights into the application of these models in predicting stock prices. The models, optimized through hyperparameter tuning, showcased varying degrees of efficacy in handling the complexities of time-series data. Gated Recurrent Unit model proved to be the best model with lowest test mean squared error in price forecasting. The project's findings highlight the significance of machine learning in financial analysis, offering a promising tool for investors and analysts.

X. References

[1] https://www.tiingo.com/

[2] https://pandas-datareader.readthedocs.io/en/latest/ remote_data.html

[3] https://pandas.pydata.org/

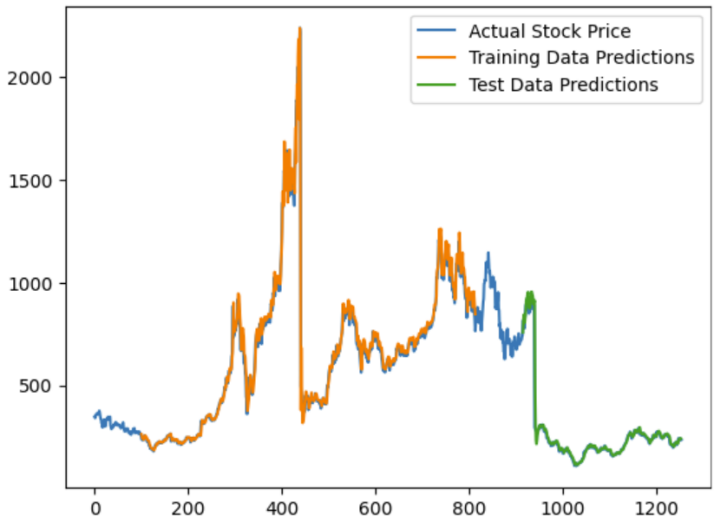

Fig. 13. Rolling window forecast with hyperparameter tuning - LSTM

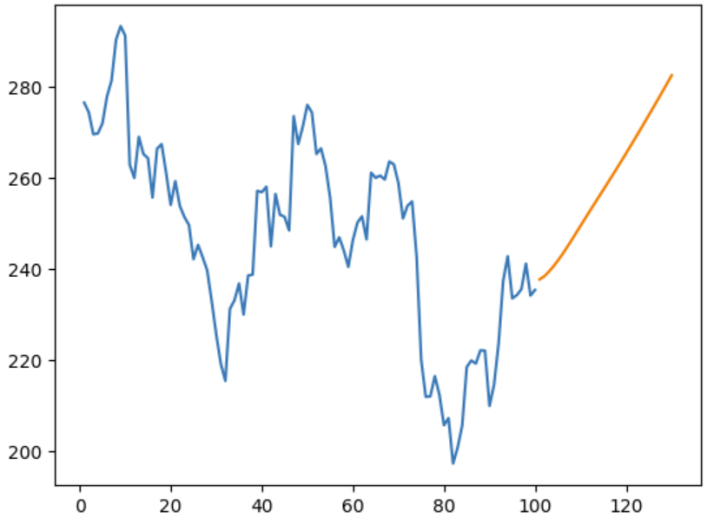

Fig. 14. Next 30 days forecast with hyperparameter tuning - LSTM

[4] https://numpy.org/

[5] https://matplotlib.org/

[6] https://scikit-learn.org/stable/

[7] https://www.tensorflow.org/

[8] https://keras.io/

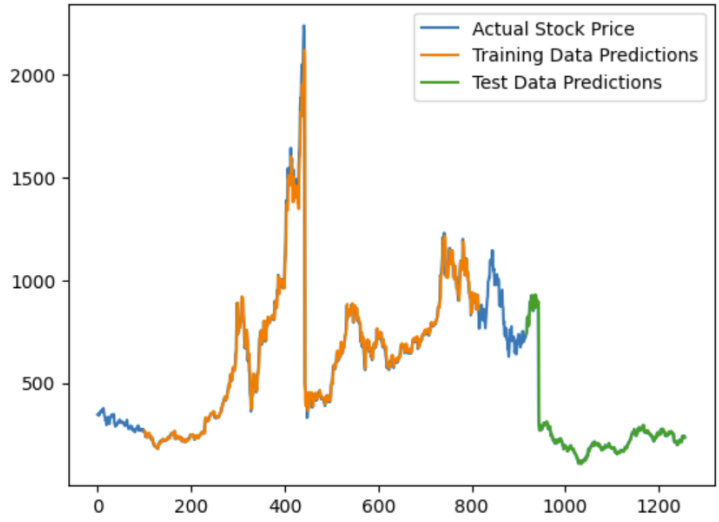

Fig. 15. Rolling window forecast with hyperparameter tuning - GRU

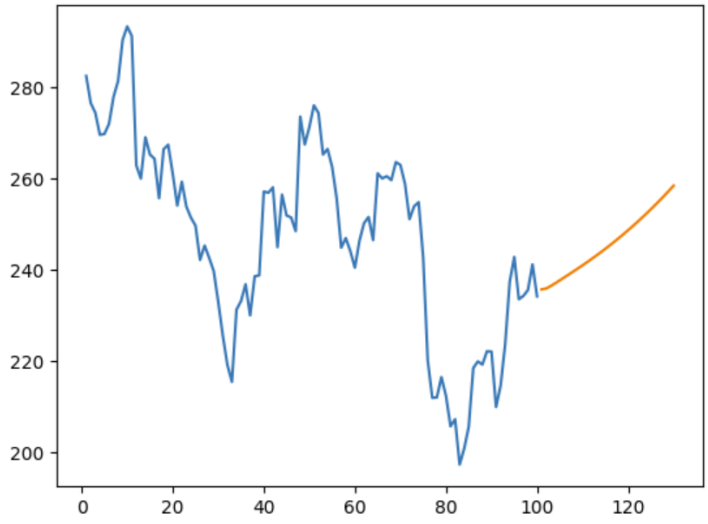

Fig. 16. Next 30 days forecast with hyperparameter tuning - GRU