MTH-443 Project Report

April 23, 2025

Team Members - Devansh Gupta (220345) Rythm Kumar (220925)

Yash Bihany (221216) Subham Anand (221093)

1 Introduction

In the ever-evolving landscape of financial markets, accurate modeling and prediction of time series data such as stock prices, returns, and volatility are essential for risk management, portfolio optimization, and strategic decision-making. This project focuses on the analysis and forecasting of financial time series using models such as the traditional Generalized Auto-regressive Conditional Heteroskedasticity (GARCH) and a variety of neural network models like RNN, LSTM, GRU and attention mechanism. We further assess these models and compare their results.

2 Data Description

We study the daily Nifty-50 stock price over the past ten years. We chose the daily **high** price of a stock to be the target variable and the remaining quantities such as **open, close, low, shares traded and turnover** as independent features variables. We predict the next day's **high** price using features of the previous **10 days**.

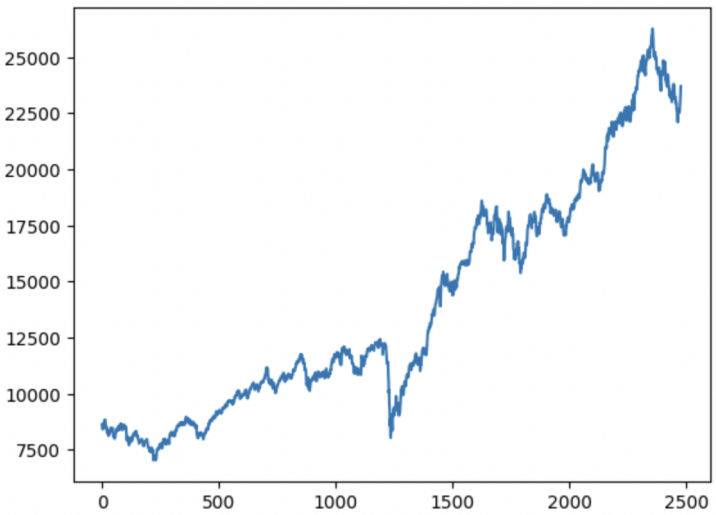


Figure 1: Daily High price of Nifty-50 stock price

3 Hypothesis Testing for ARCH effects

We first test the series for Auto-regressive conditional heteroskedasticity affects. In other words, we test whether the variance conditioned on previous values is constant or not. Usually, financial

data are volatile in nature and hence traditional models like ARIMA don't fit well. To test this we use **Engle's ARCH test**.

3.1 De-trending

From Figure 1 we can see that there is an obvious trend in the series. Hence, we first de-trend it by fitting two non-linear regression models.

- Exponential growth model: $a \times e^{bt}$
- Polynomial growth model of order two: $at^2 + bt + c$

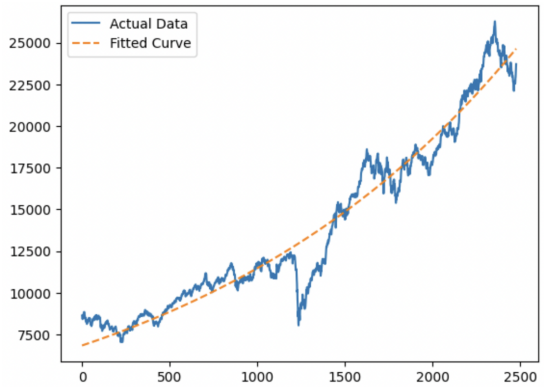


Figure 2: Fitted exponential trend

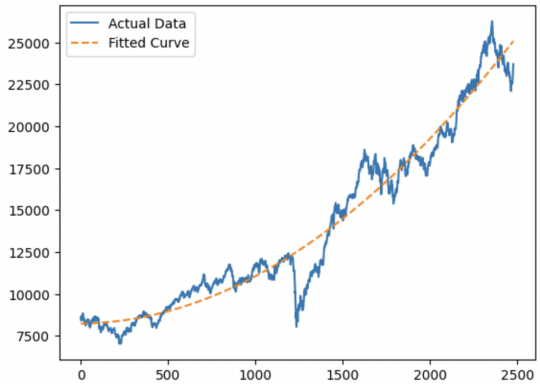


Figure 3: Fitted polynomial trend

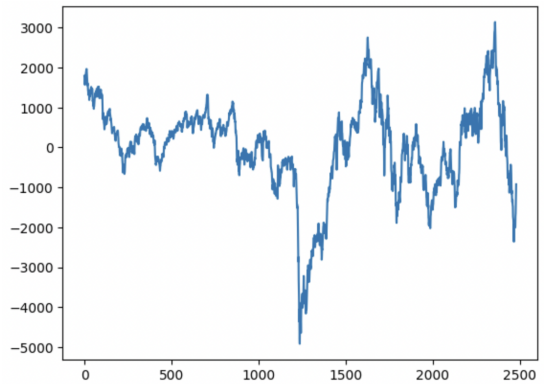


Figure 4: Residual plot of exponential trend

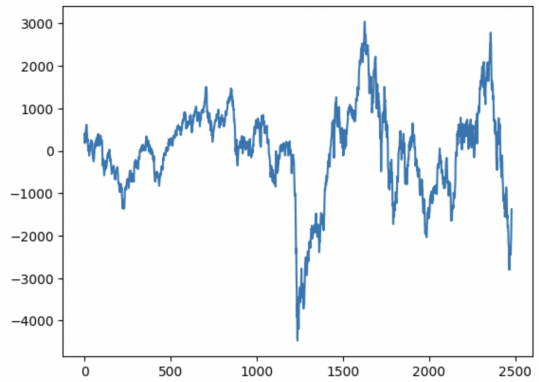


Figure 5: Residual plot of polynomial trend

To assess the fitted non-linear models, we calculate the coefficient of determination R^2 and Root Mean Squared Error (RMSE).

Model	R^2	RMSE
Exponential Model	0.9487	1163.8046
Polynomial Model	0.9556	1082.6240

Table 1: Metrics for the fitted non-linear models

From Table 2, we can see that both the models fit well on the data with polynomial growth model fitting marginally better. Hence, in our further analysis we consider the polynomial growth model.

3.2 Preliminary Analysis

On observing the residual plots Figure 4 and Figure 5 we suspect heteroskedasticity. Let's first analyze the residuals at hand.

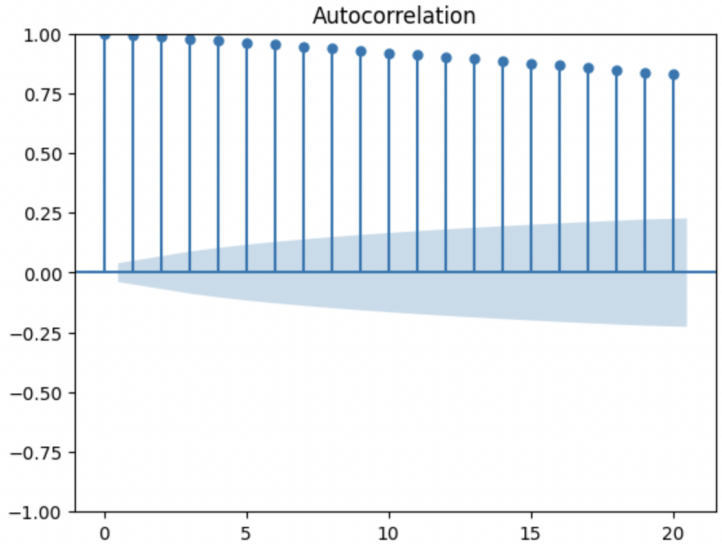


Figure 6: ACF plot of polynomial residuals

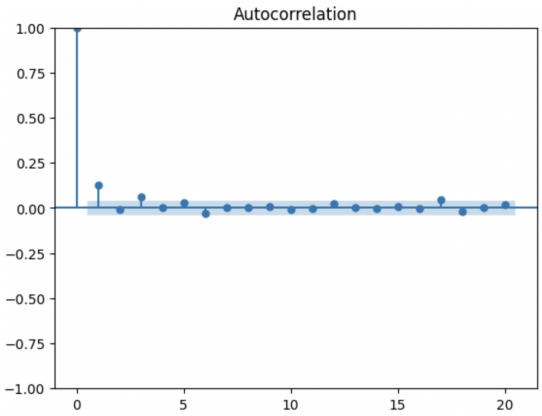


Figure 7: ACF plot of differenced residuals

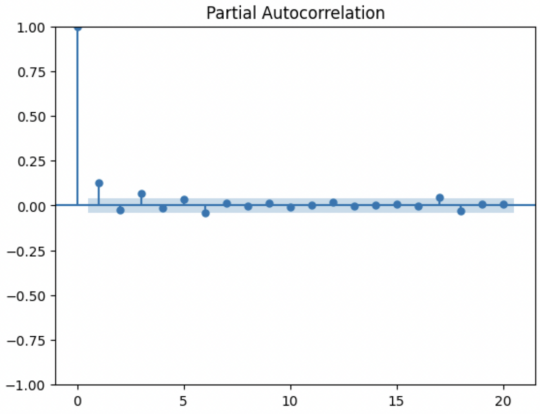


Figure 8: PACF plot of differenced residuals

From the Figure 6, we can see that there is significant auto-correlation in the residuals obtained after de-trending. Hence, we consider differencing the residuals. Figure 7 and Figure 8 show the ACF and PACF plots of the differenced series respectively. From these plots we can see both ACF and PACF cut off at lag 1. This prompts us to fit an ARMA(1,1) model on the differenced residuals.

Figures Figure 9 and Figure 10 show the ACF and PACF plots of ARMA residuals on the difference residuals (or ARIMA(1,1,1) on the residuals obtained after differencing) respectively. These plots suggest that ARMA(1,1) is a nice fit on the residuals after differencing.

3.3 Testing and results

Now that we have obtained the residuals from a base model i.e ARIMA(1,1,1), we perform the Engle's ARCH test for heteroskedasticity.

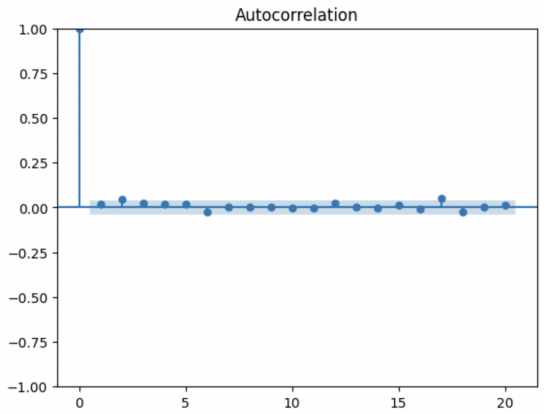


Figure 9: ACF plot of ARMA residuals

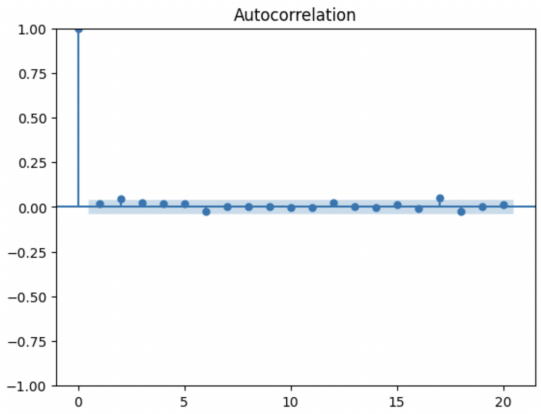


Figure 10: PACF plot of ARMA residuals

Null Hypothesis $\mathscr{H}_0$: The series is homoskedastic.
Alternate Hypothesis $\mathscr{H}_1$: The series is heteroskedastic.

LM Statistic	P-value
314.16840	$8.99 \times e^{-66}$

Table 2: Metrics for the fitted non-linear models

Based on the P-values calculated we reject the null hypothesis and conclude that the given series is heteroskedastic in nature.

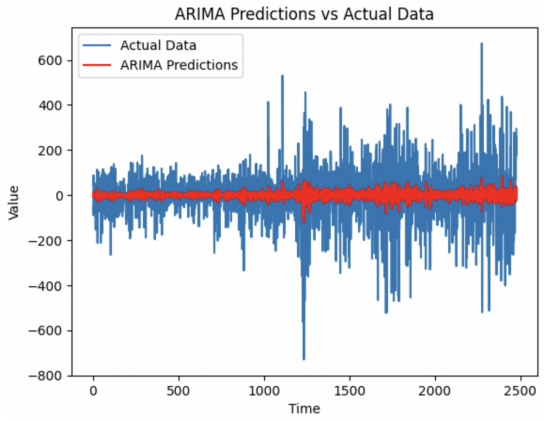


Figure 11: ARIMA predictions on residuals

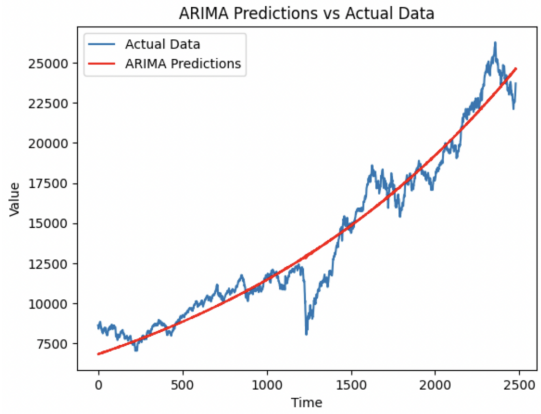


Figure 12: ARIMA predictions on the series

Figure 11 and Figure 12 show predictions by the ARIMA(1,1,1) model. The predictions obtained are too smooth, suggesting that the ARIMA model isn't able to capture the volatility in the data.

4 GARCH

To analyze the volatility in the financial return series, we first examined the ACF and PACF plots of the returns, shown below in Figure 13. These plots indicate that autocorrelations beyond lag 1 are minimal, suggesting that higher-order lag terms are unnecessary. However, due to some uncertainty at lag 1, we initially fitted a GARCH(2,2) model as a conservative choice.

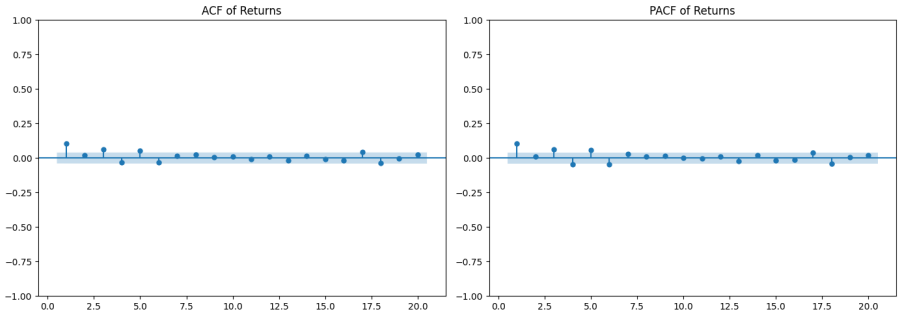


Figure 13: Combined ACF and PACF Plot of Return Series

Upon estimation, the p-values for all GARCH(2,2) parameters were greater than 0.05, indicating a lack of statistical significance. Consequently, we rejected the GARCH(2,2) model.

We then fitted a GARCH(1,1) model. The estimated parameters in this model all had p-values below 0.05, suggesting statistical significance and indicating that the GARCH(1,1) model is more appropriate for modeling the return volatility.

The p-values for the coefficients of both GARCH(2,2) and GARCH(1,1) models are shown side by side in Figure 14.

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	0.0244	0.188	0.129	0.897	[-0.345, 0.393]
alpha[1]	0.1106	0.322	0.344	0.731	[-0.520, 0.742]
alpha[2]	0.0000	1.099	0.000	1.000	[-2.154, 2.154]
beta[1]	0.5264	7.463	7.053e-02	0.944	[-14.102, 15.154]
beta[2]	0.3260	6.409	5.087e-02	0.959	[-12.235, 12.887]

(a) GARCH(2,2) Coefficient P-values

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	0.0201	6.740e-03	2.979	2.895e-03	[6.865e-03,3.328e-02]
alpha[1]	0.0857	1.696e-02	5.050	4.422e-07	[5.241e-02, 0.119]
beta[1]	0.8837	2.410e-02	36.666	2.536e-294	[0.836, 0.931]

(b) GARCH(1,1) Coefficient P-values

Figure 14: Comparison of p-values for GARCH(2,2) and GARCH(1,1) model coefficients

The predicted volatility from the GARCH(1,1) model is shown in Figure 15, along with the actual volatility. This visualization helps assess how well the model captures time-varying volatility.

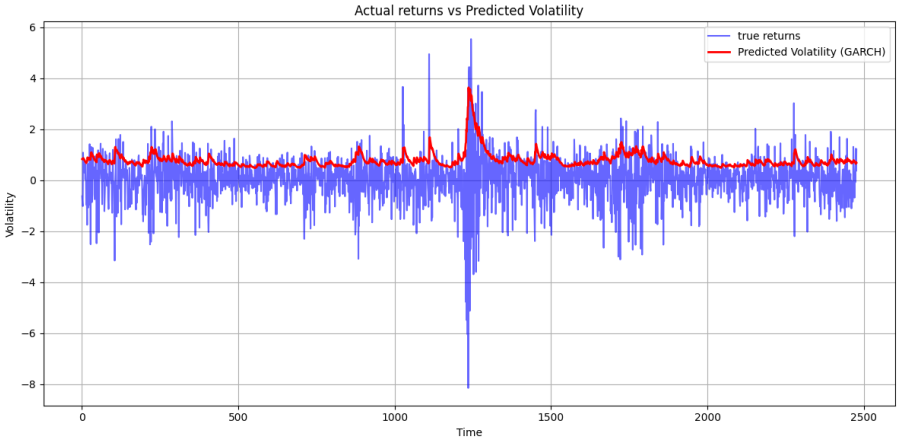
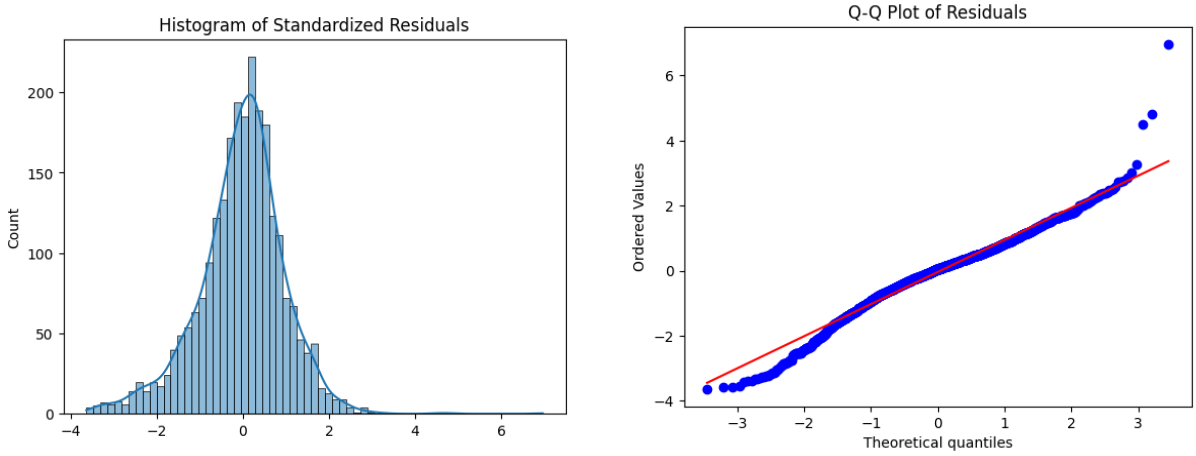


Figure 15: Predicted vs Actual Volatility (GARCH(1,1))

To evaluate the residuals of the model, we plotted the histogram of standardized residuals and the Q-Q plot side by side in Figure 16. The histogram suggests that the residuals are

approximately normally distributed, which is further supported by the Q-Q plot that shows the residuals closely following the 45-degree reference line.



(a) Histogram of Standardized Residuals

(b) Q-Q Plot of Standardized Residuals

Figure 16: Residual Analysis: Histogram and Q-Q Plot of Standardized Residuals

Finally, the mean squared error (MSE) of the volatility prediction is **4.409**, reflecting the predictive performance of the GARCH(1,1) model.

5 Neural Network Models

We fit several neural network models for sequential data on the given series. These models use the feature variables(Open, Close, Low, Shares traded and Turnover) of the previous days to predict the target variable(High). We set the **window as 10** [1] and **batch size as 32** for each of the following models.

- Simple RNN
- LSTM
- GRU
- Attention + LSTM
- Attention + GRU

5.1 Simple RNN

We begin by training a simple **Seq2Vec** RNN model with a single hidden layer of size 50. The model is optimized using the Adam optimizer, and the loss is measured using Mean Squared Error (MSE). To ensure training stability, we apply gradient clipping with a maximum norm of 5 during backpropagation. The model is trained for 50 epochs on the training dataset.

5.2 LSTM

We implement a stacked LSTM model with two layers and a hidden state size of 64 to process sequential input data. The model takes input sequences with multiple features and produces

[1] Window is the number of previous observations used to predict the target at a given time point.

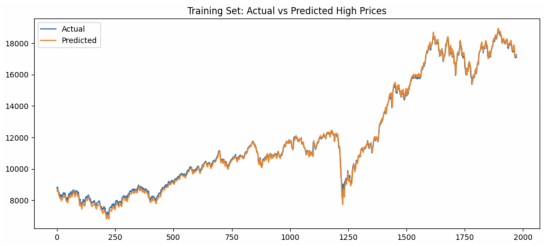


Figure 17: Prediction on training dataset by Simple RNN model

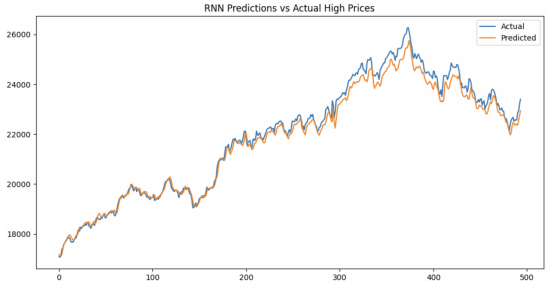


Figure 18: Prediction on test dataset by Simple RNN model

a single output using a fully connected layer after the final LSTM output. Dropout is applied between the LSTM layers to help prevent overfitting.

The model is trained using the Adam optimization algorithm with a learning rate of 0.001, and Mean Squared Error (MSE) is used as the loss function. Training is carried out for 100 epochs on the training data.

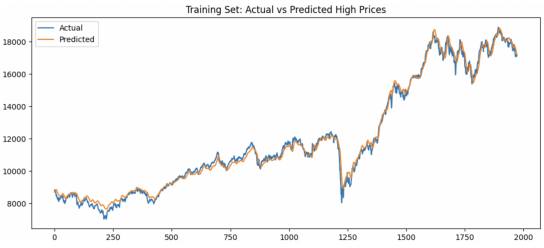


Figure 19: Prediction on training dataset by LSTM model

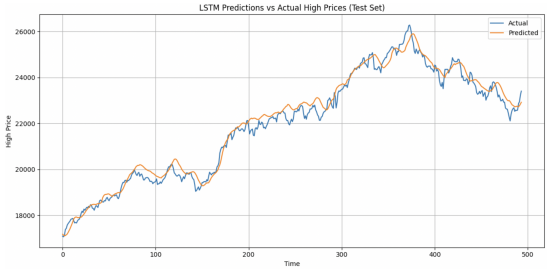


Figure 20: Prediction on test dataset by LSTM model

5.3 GRU

To model the temporal dependencies in the time series data, we implemented a Gated Recurrent Unit (GRU) neural network. The input to the model consisted of sequences of the previous 10 days of actual values, each day represented by four features: `[high, low, close, open]`. This sliding window approach allowed the GRU to learn from short-term historical patterns.

The model was trained to predict the **normalized high price** for the next day. The normalization was done to stabilize training and improve convergence. The GRU architecture consisted of two layers with a hidden size of 64 units, followed by a fully connected linear layer to project the hidden representation to the desired output size. The model was trained using the Mean Squared Error (MSE) loss function and optimized using the Adam optimizer with a learning rate of 0.001 for 50 epochs.

The GRU was trained on the training set and then used to generate predictions on both the training and test data. Figure 21 shows a comparison between the actual and predicted values of the normalized high price for both training and test periods. The model demonstrates its ability to capture the underlying trend in the data and performs reasonably well in forecasting future values.

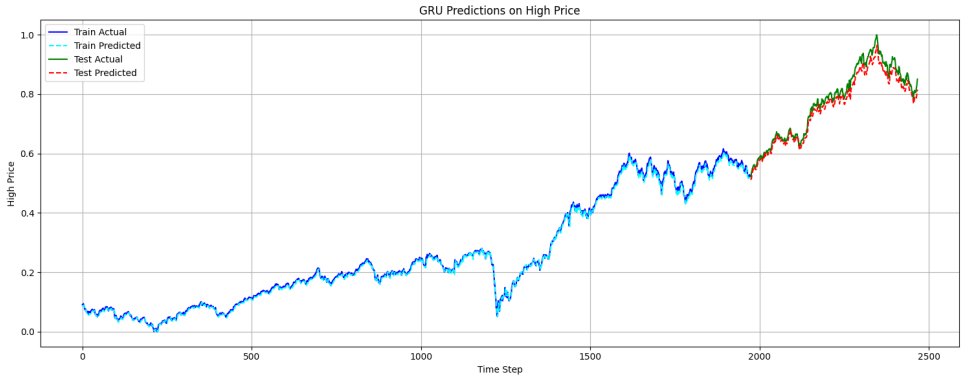


Figure 21: GRU model predictions vs actual values (normalized high price) on train and test sets

The GRU's performance highlights its effectiveness in modeling sequential patterns and learning from past observations in the time series. However, further tuning or regularization may be explored to reduce any observed prediction errors or overfitting tendencies.

5.4 Attention + LSTM

We develop an LSTM model enhanced with an attention mechanism to capture important time steps within the input sequence. The model consists of two LSTM layers with a hidden size of 64, followed by an attention layer that computes a weighted representation (context vector) of the sequence. This context vector is then passed through a fully connected layer to generate the final output.

The model is trained for 100 epochs using the Adam optimizer with a learning rate of 0.001. The training objective is to minimize the Mean Squared Error (MSE) loss between the predicted and actual values. A dropout rate of 0.2 is applied between LSTM layers to reduce the risk of overfitting.

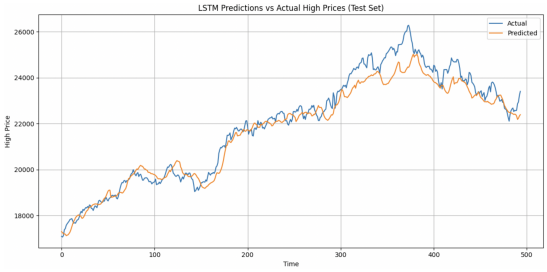


Figure 22: Prediction on test dataset by Attention + LSTM model

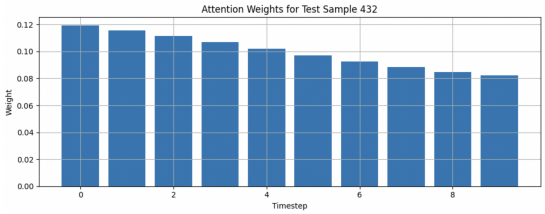


Figure 23: Attention weights of the inputs for a prediction chosen at random

The plot Figure 23 shows the weights given to the inputs while making a prediction. The model uses the features of the past 10 days to predict the high price of the next day. The highest attention weight is given to the previous day and then gradually decreases as we move further into the past.

5.5 GRU with Attention

To enhance the interpretability and performance of our model, we extended the GRU architecture by incorporating an attention mechanism. Unlike standard GRU models, where each time step in the sequence has equal importance, the attention mechanism allows the model to assign

varying levels of importance to different time steps in the sequence, thereby enabling it to focus on the most relevant past observations.

In this model, the input consisted of sequences of the previous 10 days of normalized `[high, low, close, open]` values. The attention mechanism was applied first to compute the attention weights for each day in the input sequence. These weights reflect the importance of each time step in predicting the next day's high price. After the attention weights were computed, they were used to generate a weighted input sequence, which was then fed into a GRU layer for further processing.

The GRU layer processed the weighted input and produced a context vector, which was passed through a fully connected layer to predict the normalized high price for the next day. The model was trained using Mean Squared Error (MSE) loss and optimized with the Adam optimizer over 500 epochs.

The results of the model on both the training and test sets are shown in Figure 24, where we compare the predicted values to the actual normalized high prices.

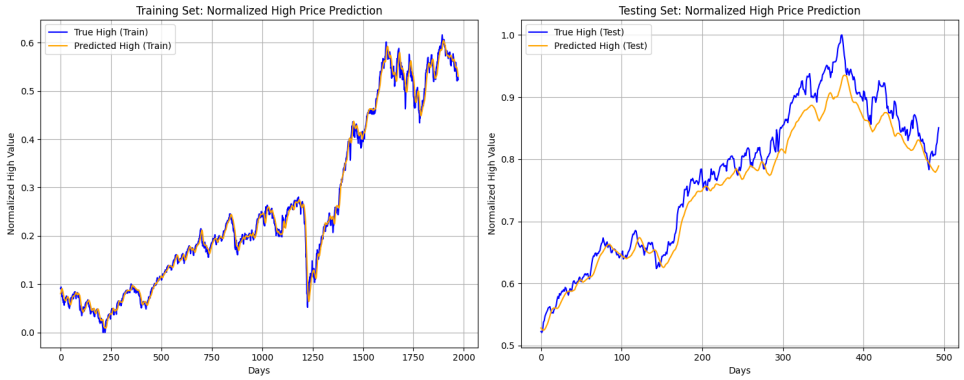


Figure 24: GRU with Attention model predictions vs actual values (normalized high price) on train and test sets

To better understand the model's behavior, we visualize the attention weights for five test samples in Figure 25. This single image displays the attention weights for the 10-day input sequence for each of the five test samples. The attention weights indicate how the model assigns importance to different days in the input sequence when making predictions. As seen in the image, the model focuses more on certain days, reflecting its ability to learn which past observations are most relevant for predicting future high prices.

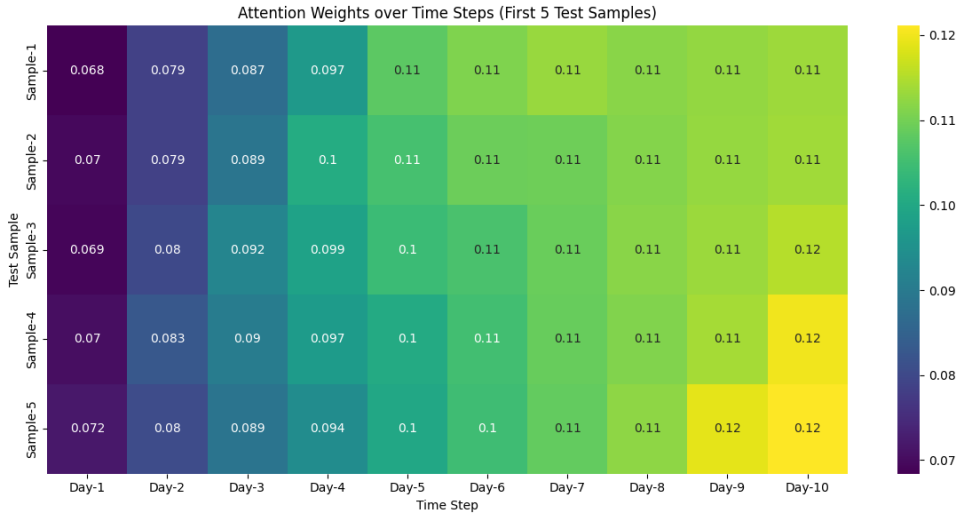


Figure 25: Attention weights for five different test samples (over the 10-day input window)

The attention weights provide valuable insights into how the model prioritizes specific days in the input sequence, offering interpretability regarding the temporal patterns the model uses to make its predictions. This mechanism allows for more transparent decision-making, particularly when evaluating which past data points are most influential for the model's forecasts.

6 Evaluations

To evaluate the performance of various models we use the Mean Absolute Percentage Error(MAPE) defined as follows.

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

The following are the results.

Models	MAPE(Train)	MAPE(Test)
Simple RNN	0.010%	0.014%
LSTM	0.013%	0.015%
GRU	0.0075%	0.0064%
Attention + LSTM	0.015%	0.017%
Attention + GRU	0.014%	0.023%

Table 3: MAPE of different Models

7 Conclusion

It this project we analyzed and assessed the performance of traditional and various neural-network based models on financial time series data. We observed that GRU performs the best as it's predictions had the least Mean Absolute Percentage Error(MAPE). We also observed that the attention weights assigned to inputs were in decreased as the lag increased denoting that the previous day's prices were the most important.