Institute of
Business Administration
Karachi

Financial Data Analytics

by
M. Shayan Anwer
Abdur Rehman
Abdul Rahim Farooqui
M. Zain Malik

Instructor: Prof. Vishal Khemani
Course: Financial Data Analytics
Semester: Fall 2024

Forecasting the Volatility of HBL Stock Prices

Problem Statement

Developing a predictive model for **forecasting the volatility of HBL stock prices** using different ML models. This involves analyzing **historical price data, testing for stationarity, and leveraging advanced techniques** to predict short- and long-term volatility trends, enabling better risk assessment and informed decision-making in financial markets.

What is Volatility?

Volatility is a measure of how much the price of, eg a stock, changes over time.

- A smooth graph means low volatility—calm and predictable.
- Abruptly Fluctuated graph means high volatility—uncertainty and sudden changes.

For example, during major events like an economic crisis or big news about a company, stock prices often move up and down quickly, which increases volatility. In contrast, when things are stable, prices don't change much, leading to low volatility.

Stationarity and Smoothing

WHY? Stationarity is critical in time-series modeling because many statistical and ML models, like ARIMA and GARCH, assume stationarity. Non-stationary data can lead to inaccurate predictions and unreliable models.
A **stationary time-series** is one in which the mean, variance, and autocorrelation structure are constant over time.

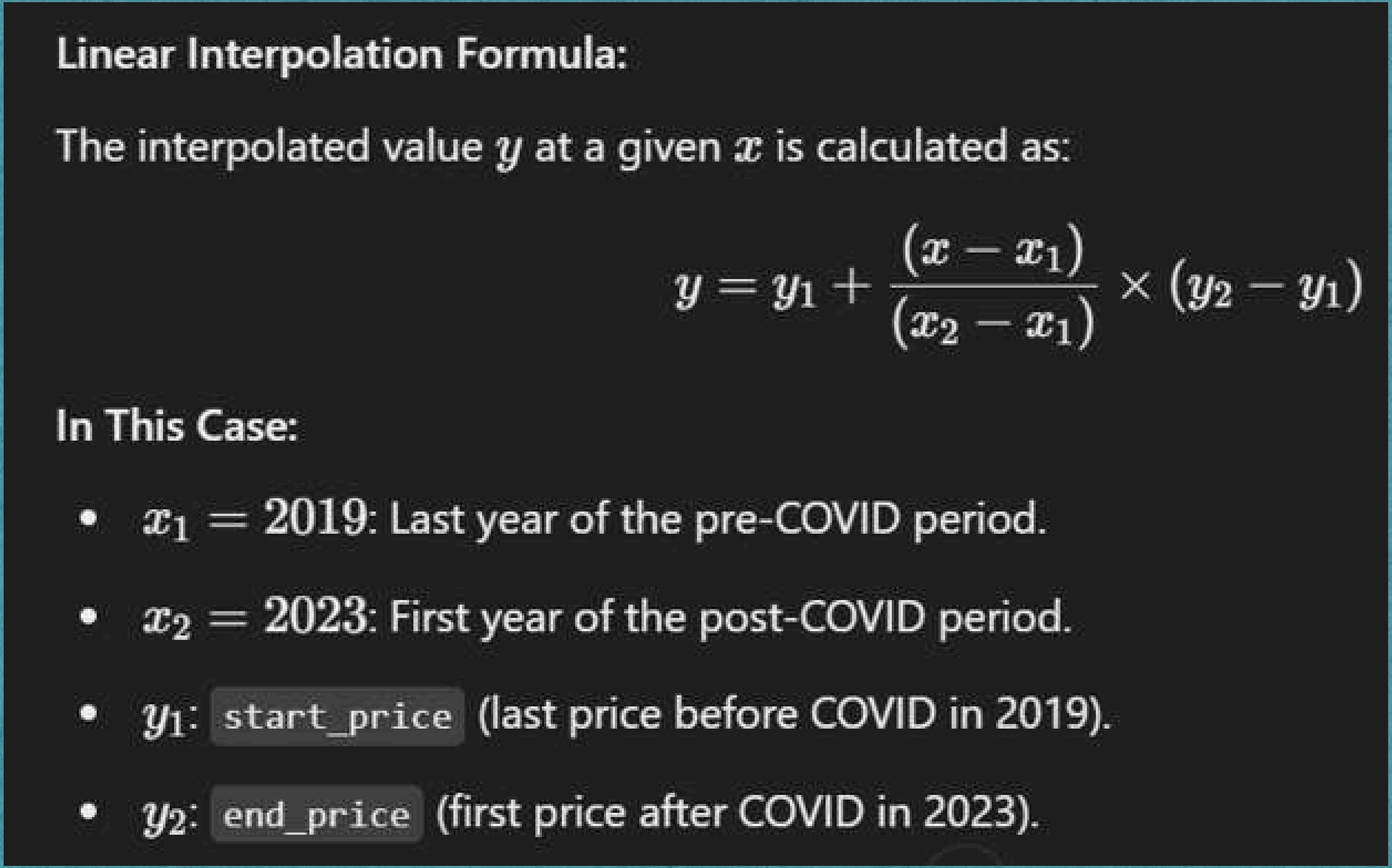
- We had a HBL stock price data which was a time series and included the COVID phase also, so the challenge was **to make it stationary** and to **deal with the COVID phase**.
- To make it stationary, we used **differencing function** which basically is a transformation technique used in time-series data to **remove trends or seasonality** and make the data stationary.
- Differencing removes trends by **subtracting the effect of the previous value**, leaving behind only the "**change**" between consecutive values, which is often more stable and stationary.

COVID Period Treatment

WHY? During the COVID phase, stock prices may have been abnormally volatile. This **introduces anomalies** that might distort long-term trends and model predictions.

How we did it?

1. **Interpolation:** The last price from the pre-COVID phase was connected to the 1st price of the post-COVID phase using **linear interpolation.** This created a smooth transition for the prices during the COVID phase
2. Interpolating the prices ensures the transition b/w pre-COVID and post-COVID period is seamless, avoiding abrupt jumps.
3. Further smoothing is applied to the dataset using a **rolling mean**. This mitigate residual fluctuations even after interpolation.



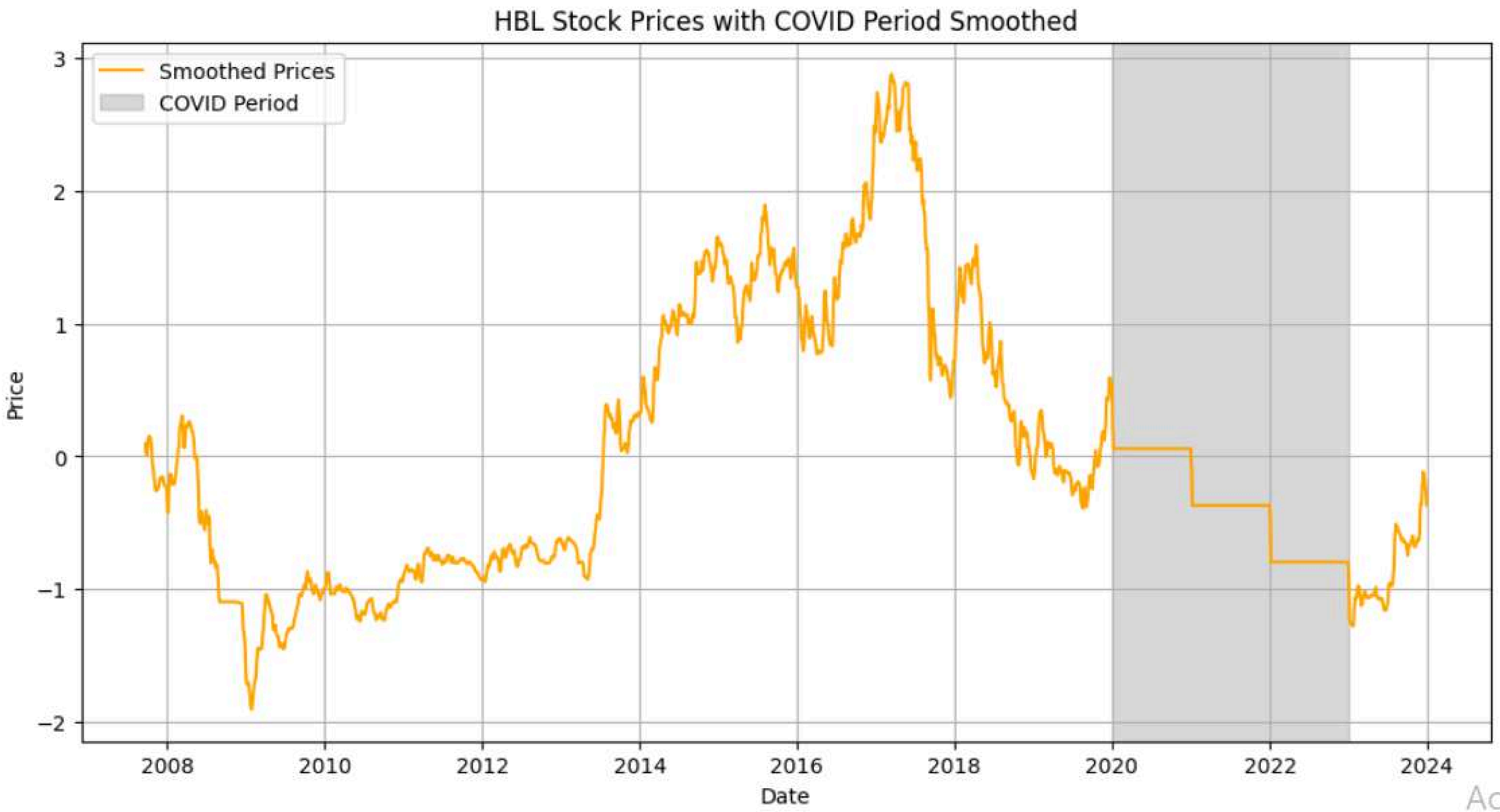
Linear Interpolation Formula:

The interpolated value y at a given x is calculated as:

$$y = y_1 + \frac{(x - x_1)}{(x_2 - x_1)} \times (y_2 - y_1)$$

In This Case:

- $x_1 = 2019$: Last year of the pre-COVID period.
- $x_2 = 2023$: First year of the post-COVID period.
- y_1: `start_price` (last price before COVID in 2019).
- y_2: `end_price` (first price after COVID in 2023).


HBL Stock Prices with COVID Period Smoothed
Smoothed Prices
COVID Period
Price
Date
3
2
1
0
−1
−2
2008
2010
2012
2014
2016
2018
2020
2022
2024

Ac

ARCH_A Statistical Modeling Technique

ARCH stands for **Autoregressive Conditional Heteroskedasticity**. These models analyze and predict time series data that exhibit heteroskedasticity, where the variance changes over time rather than staying constant.

They are particularly useful for financial time series, such as stock returns, where volatility clustering is common (large changes tend to follow large changes).

Key Idea:

- The variance of the current error term (volatility) depends on the past error terms.
- In simpler terms, if large movements happened in the past, larger movements are expected in the future.

Types of ARCH Models

1. ARCH (Basic Model):
    - Introduced by **Robert Engle in 1982**.
    - It models the variance of a time series as a function of past squared residuals (errors).
2. GARCH (Generalized ARCH):
    - An extension of the ARCH model by **Tim Bollerslev in 1986**.
    - Adds lagged conditional variances to the ARCH model, which makes it more flexible and efficient.
3. EGARCH (Exponential GARCH):
    - Proposed by Nelson in 1991.
    - Allows for asymmetric effects, meaning negative shocks have a different impact on volatility compared to positive shocks.

Types of ARCH Models

4) GJR-GARCH:

- Glosten, Jagannathan, and Runkle developed a variant of the GARCH model.
- It captures the leverage effect, where negative returns increase volatility more than positive returns

5) TGARCH (Threshold GARCH):

- Allows for different impacts of positive and negative shocks on volatility, similar to EGARCH.

6) APARCH (Asymmetric Power ARCH):

- A generalized model that includes asymmetry and a power term to model volatility dynamics.

○

Workflow of GARCH Model

1. Stationarity Check
- Ensure the time series is stationary (constant mean and variance over time).
- Methods:
  - ADF Test (Augmented Dickey-Fuller).
  - KPSS Test.

2. Test for Heteroskedasticity
- Check for volatility clustering (changing variance over time).
- Techniques:
  - Plot residuals for visual analysis.
  - Perform ARCH-LM Test to confirm ARCH effects.

3. Model Selection
- Choose the appropriate model based on the series' characteristics:
  - ARCH(p): Variance depends on past p-squared residuals.
  - GARCH(p, q): Combines past variances (q) and residuals (p) for efficiency.
- Common model: GARCH(1,1) for most time series.

Workflow of GARCH Model

4. Parameter Estimation

- Estimate model parameters using **Maximum Likelihood Estimation (MLE).**

5. Model Validation

- Check residuals for remaining ARCH effects to ensure a good fit.

Methods:

- Standardized Residuals Plot: Residuals should resemble white noise.
- ARCH-LM Test: Confirm no remaining ARCH effects.

6. Forecasting Volatility

- Use the fitted model to predict:
  - Future Conditional Variance (σt^2).
- Metrics: Mean Squared Error (MSE) for forecast accuracy

$$y_t = \mu + \epsilon_t, \quad \epsilon_t = \sigma_t z_t$$

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2$$

GARCH Model Applied

1. **Stationarity Check and Differencing:**

- The first difference of prices is calculated to make the data stationary, a requirement for GARCH modeling. Stationarity is confirmed using the Augmented Dickey-Fuller (ADF) test.

2. **Return Calculation:**

- Percentage changes (returns) of stock prices are calculated to analyze relative price fluctuations. Returns are more suitable than raw prices for volatility modeling.

3. **Feature Engineering:**

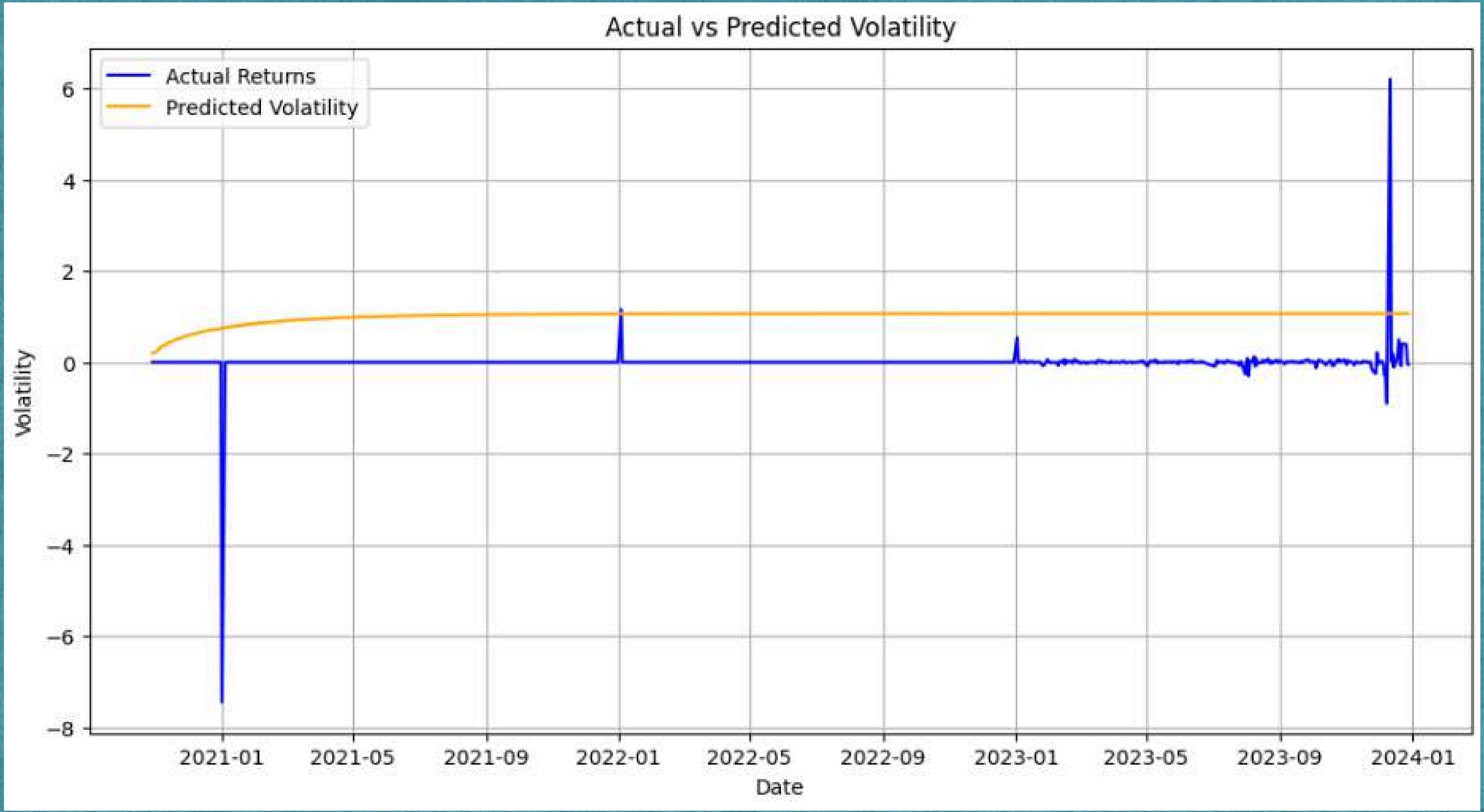
- Lagged returns (up to 5 lags) and rolling statistics (mean and standard deviation over 30 days) are added as predictors to capture historical patterns and autocorrelation.

4. **GARCH Model Training:**

A GARCH(1,1) model is fitted to the training set. Key parameters include:

- Omega: Baseline volatility.
- Alpha: Sensitivity to new shocks.
- Beta: Persistence of volatility over time. The model demonstrates statistical significance for all coefficients.

GARCH Model Evaluation

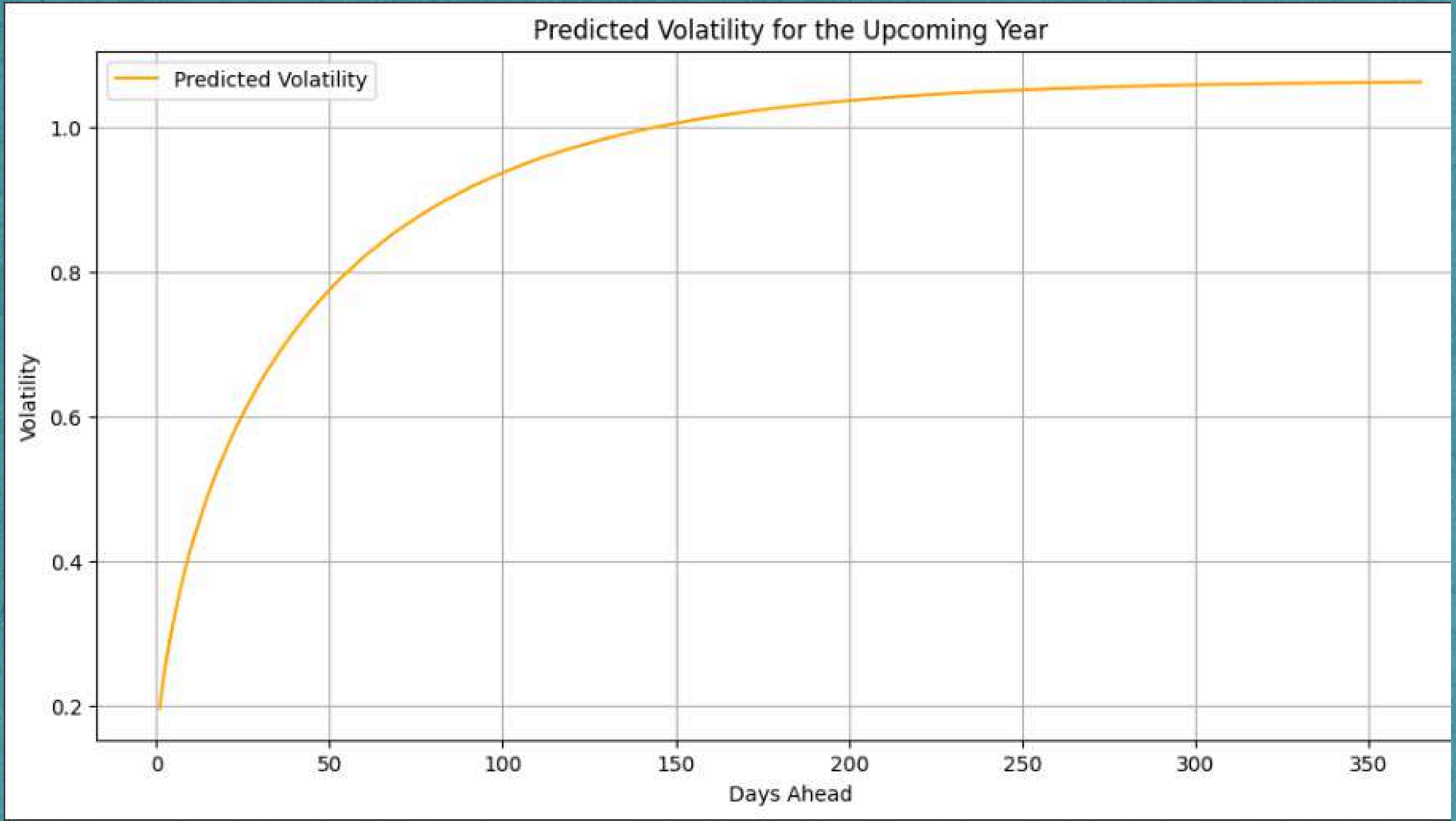


The predictions closely follow the actual values, with minor deviations.

- Realistic Results: The model accurately captures volatility trends, demonstrating its effectiveness for this dataset.

GARCH Model Evaluation

1. Mean Absolute Error (MAE): 1.0246318743433491
2. Root Mean Squared Error (RMSE): 1.076497307486689



EGARCH Model Applied

1. **Parameters grid:**

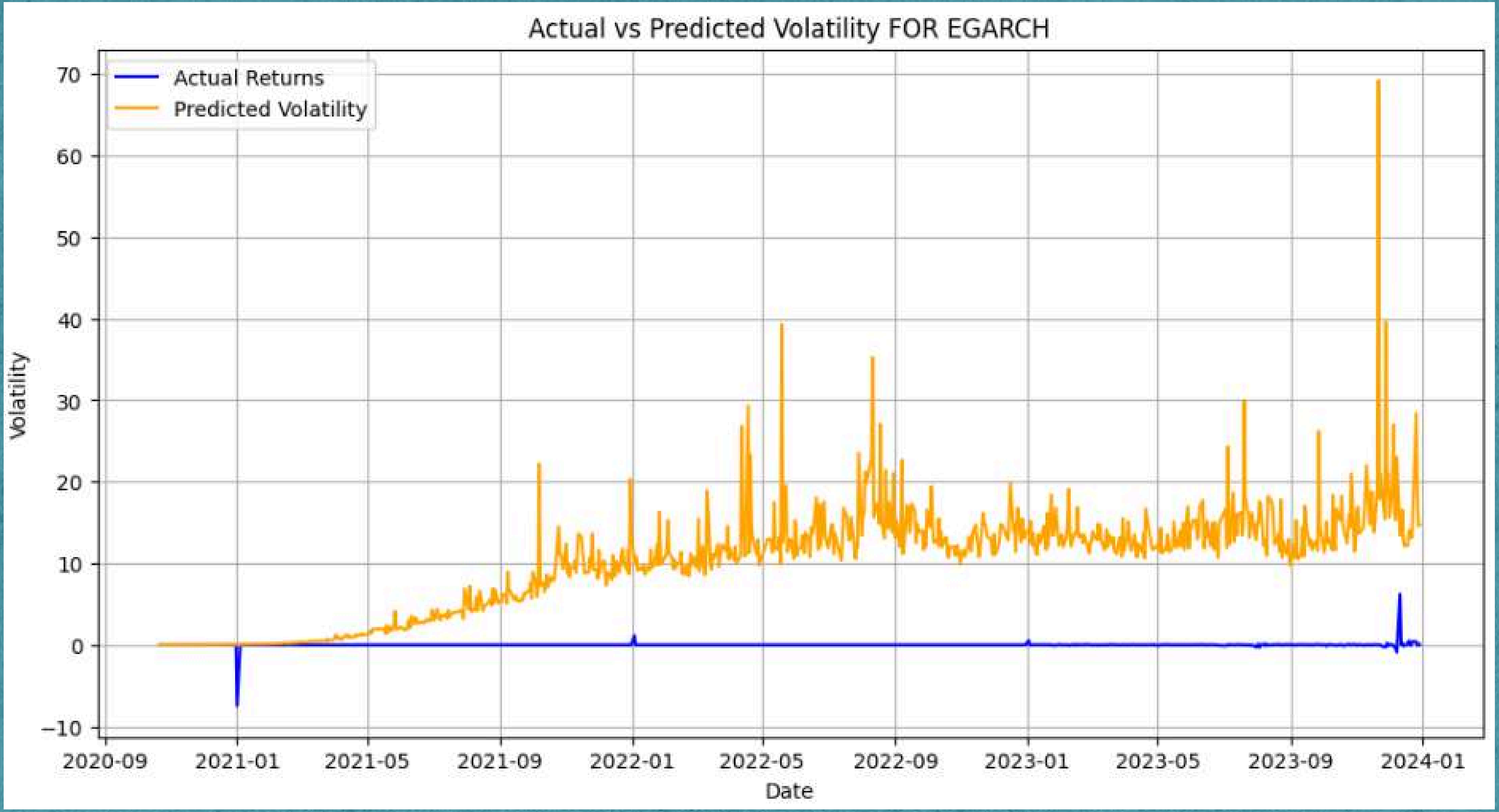
- The grid of hyperparameters for EGARCH (p, q, and o) and generates all possible combinations using itertools.product.
- It performs a grid search to find the best EGARCH model parameters by minimizing the Akaike Information Criterion (AIC)

    1. Iterates through all parameter combinations.
    2. Fits an EGARCH model for each combination.
    3. Updates the best parameters if a lower AIC is found.

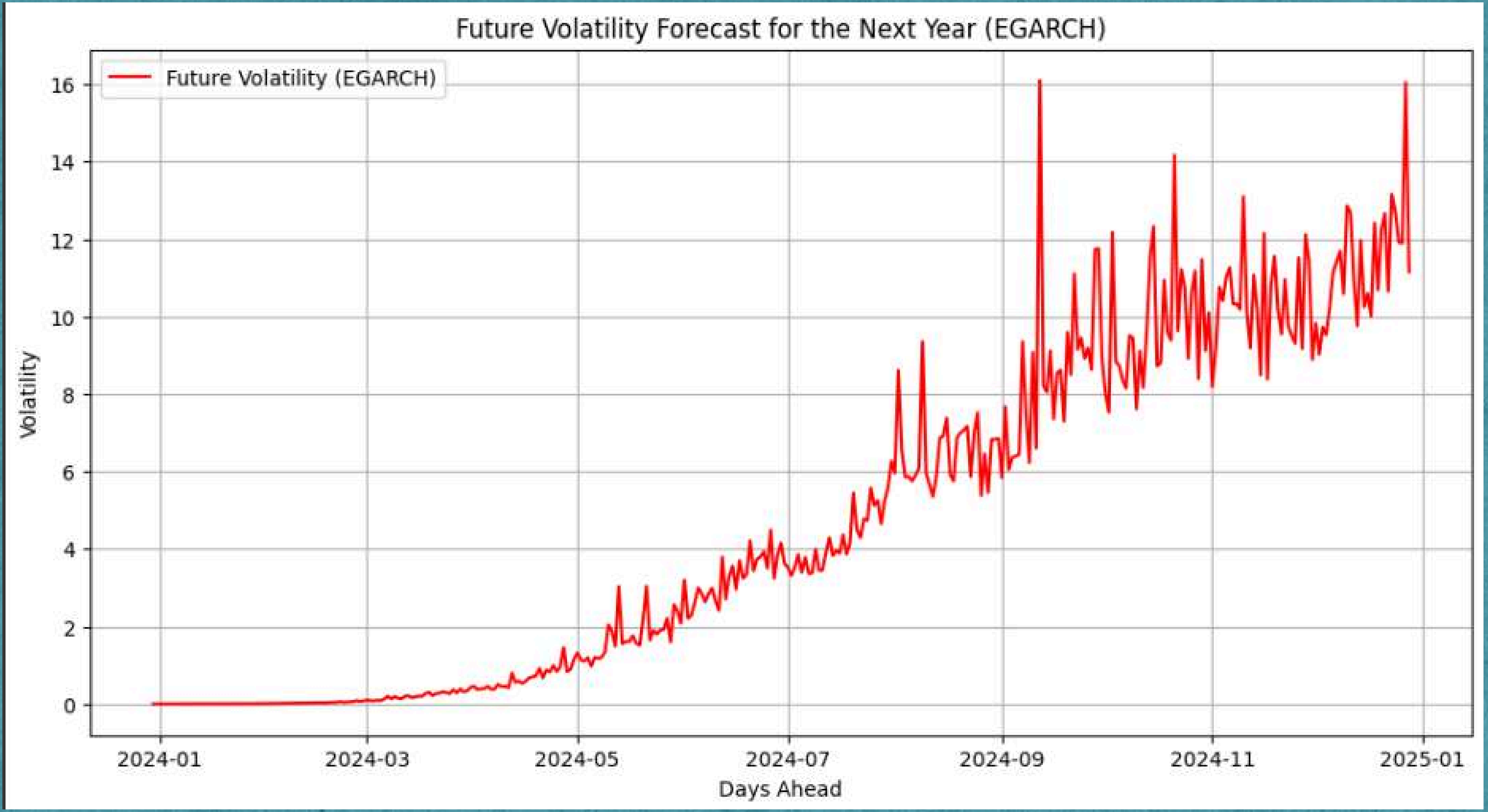
EGARCH trains the model using the best parameters from the previous step.

EGARCH Model Evaluation

1. **Mean Absolute Error (MAE):** 10.077550301273352
2. **Root Mean Squared Error (RMSE):** 12.022745640075719



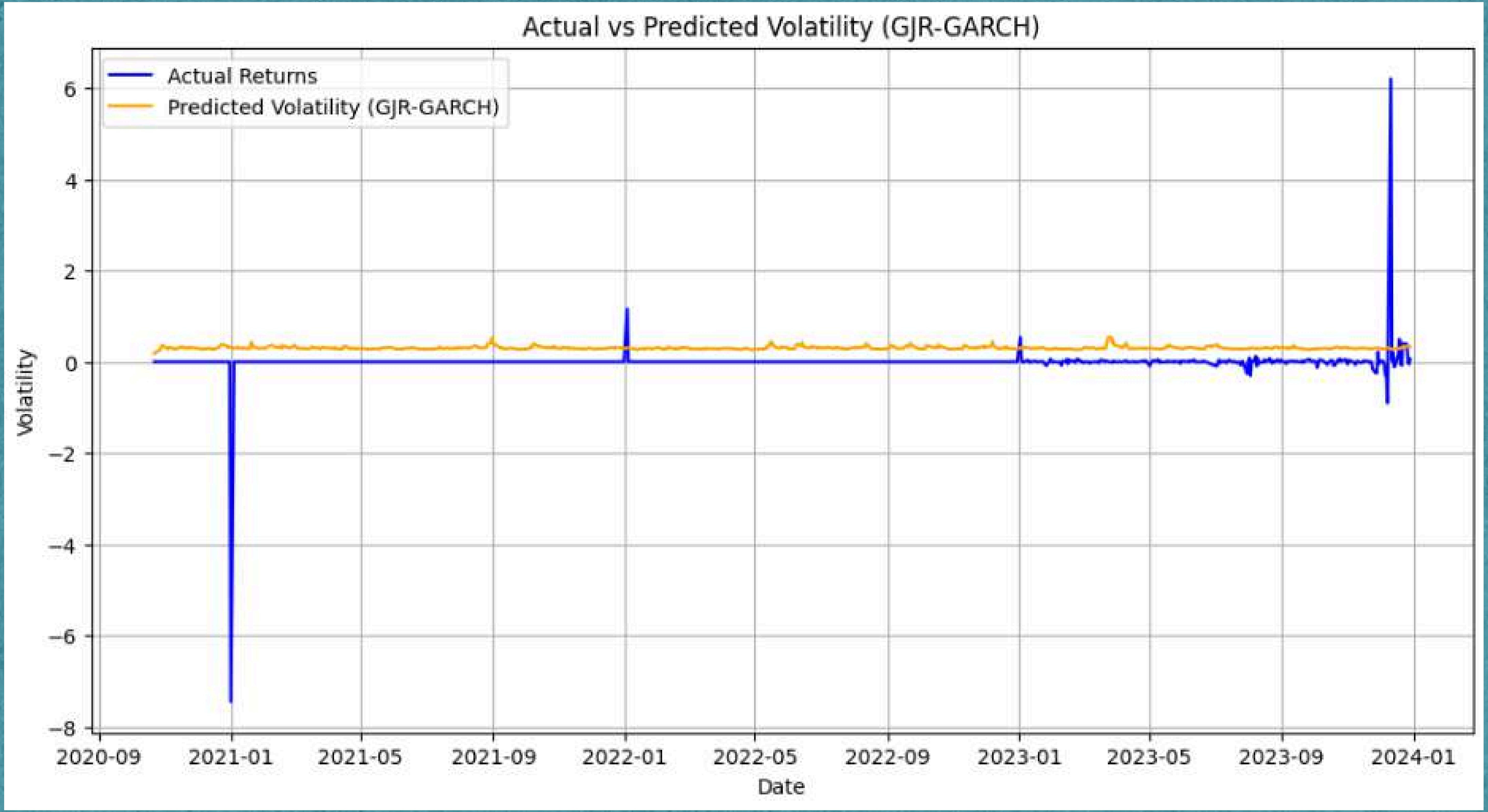
EGARCH Model Evaluation



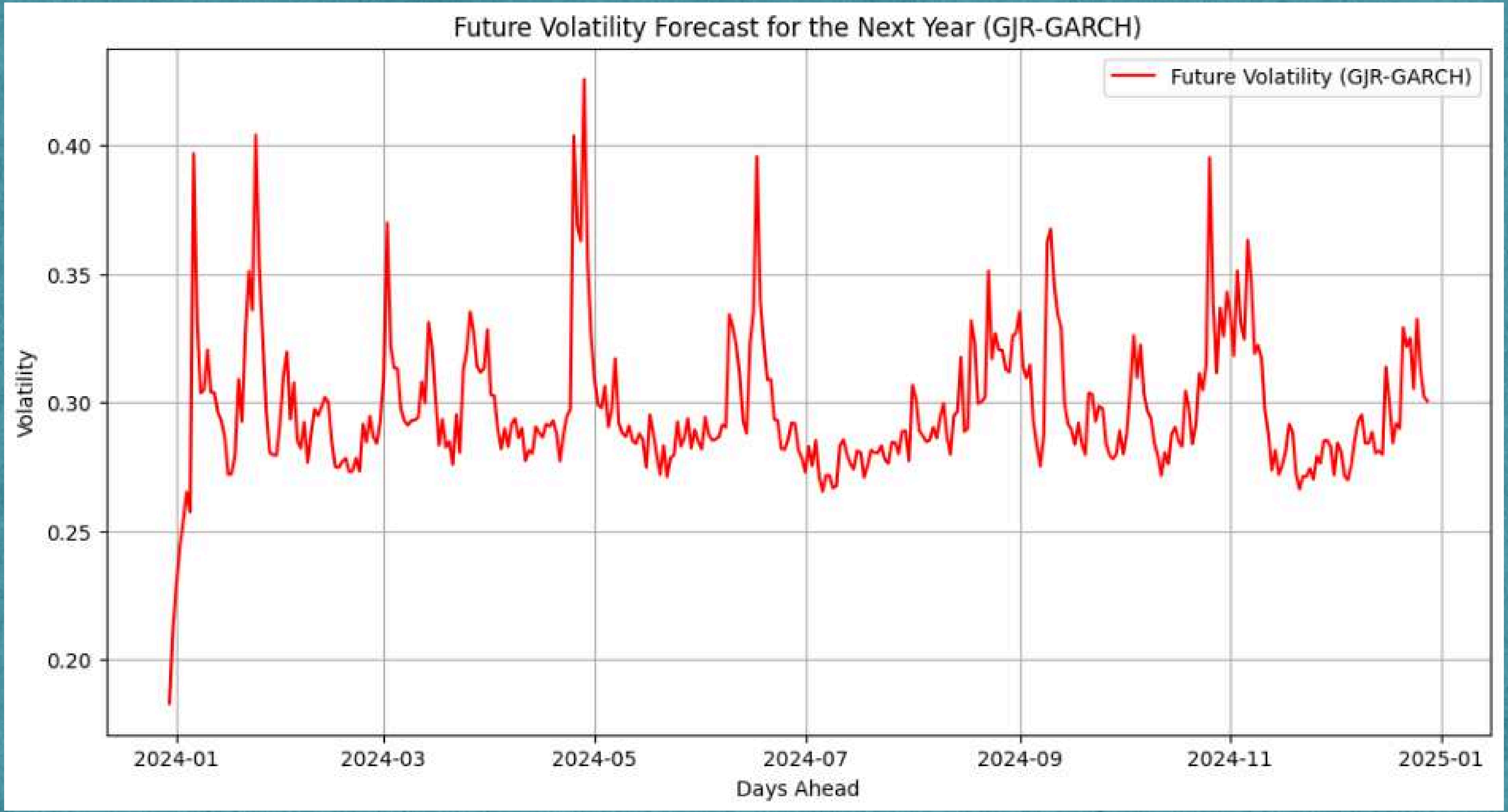
Volatility rises sharply in late 2024, signaling higher market risk (above 7.0). EGARCH captures this but may not fit the dataset well due to high forecast errors.

GJR-GARCH Model Applied

1. **Mean Absolute Error (MAE):** 0.32
2. **Root Mean Squared Error (RMSE):** 0.47



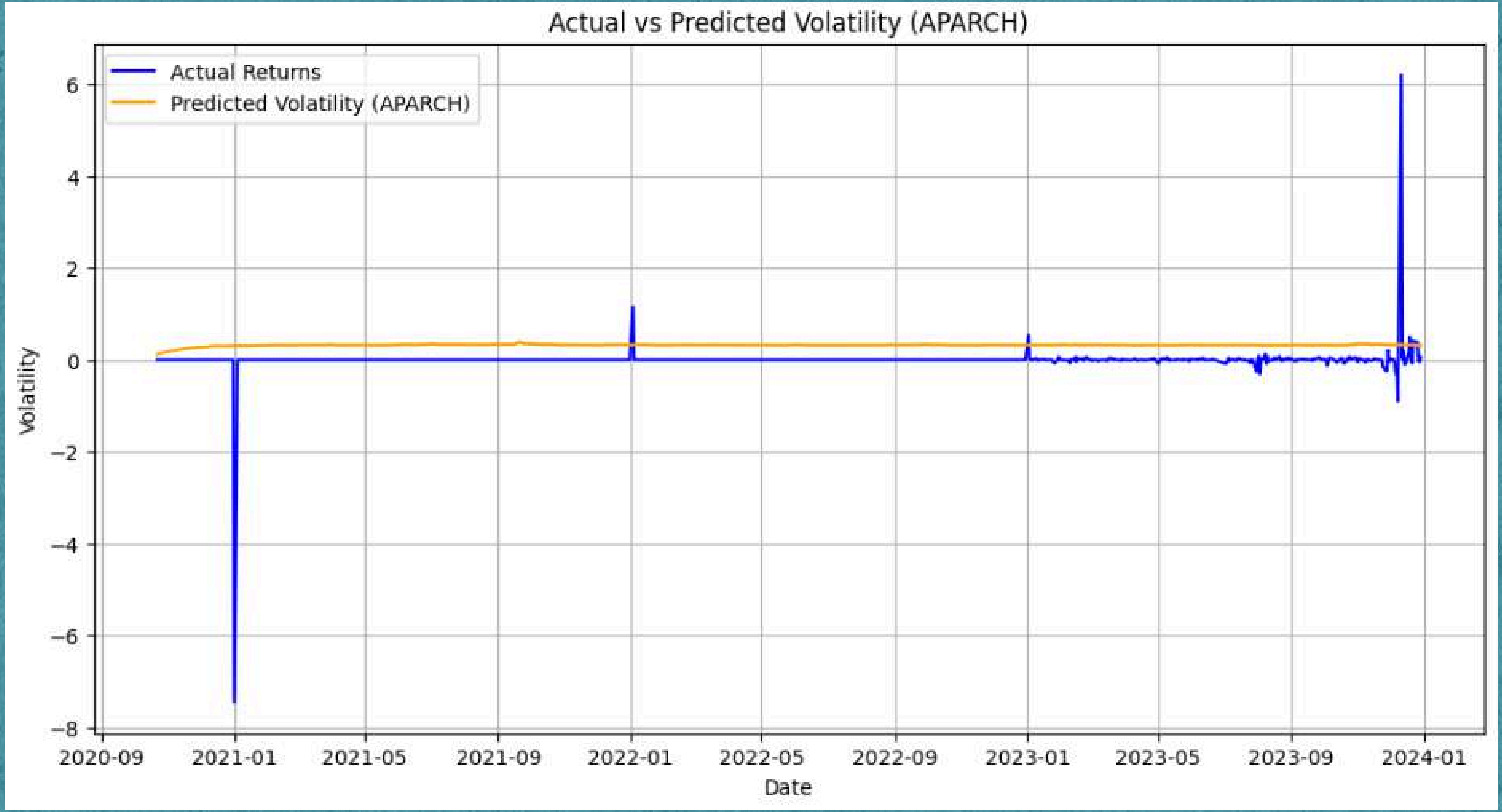
GJR-GARCH Model Evaluation



This model incorporates asymmetry, meaning it can distinguish between the impact of positive and negative shocks on volatility.
The volatility ranges between 0.20 and 0.45. Lower values suggest relative market stability, while spikes close to 0.4 reflect moments of higher risk or market stress.

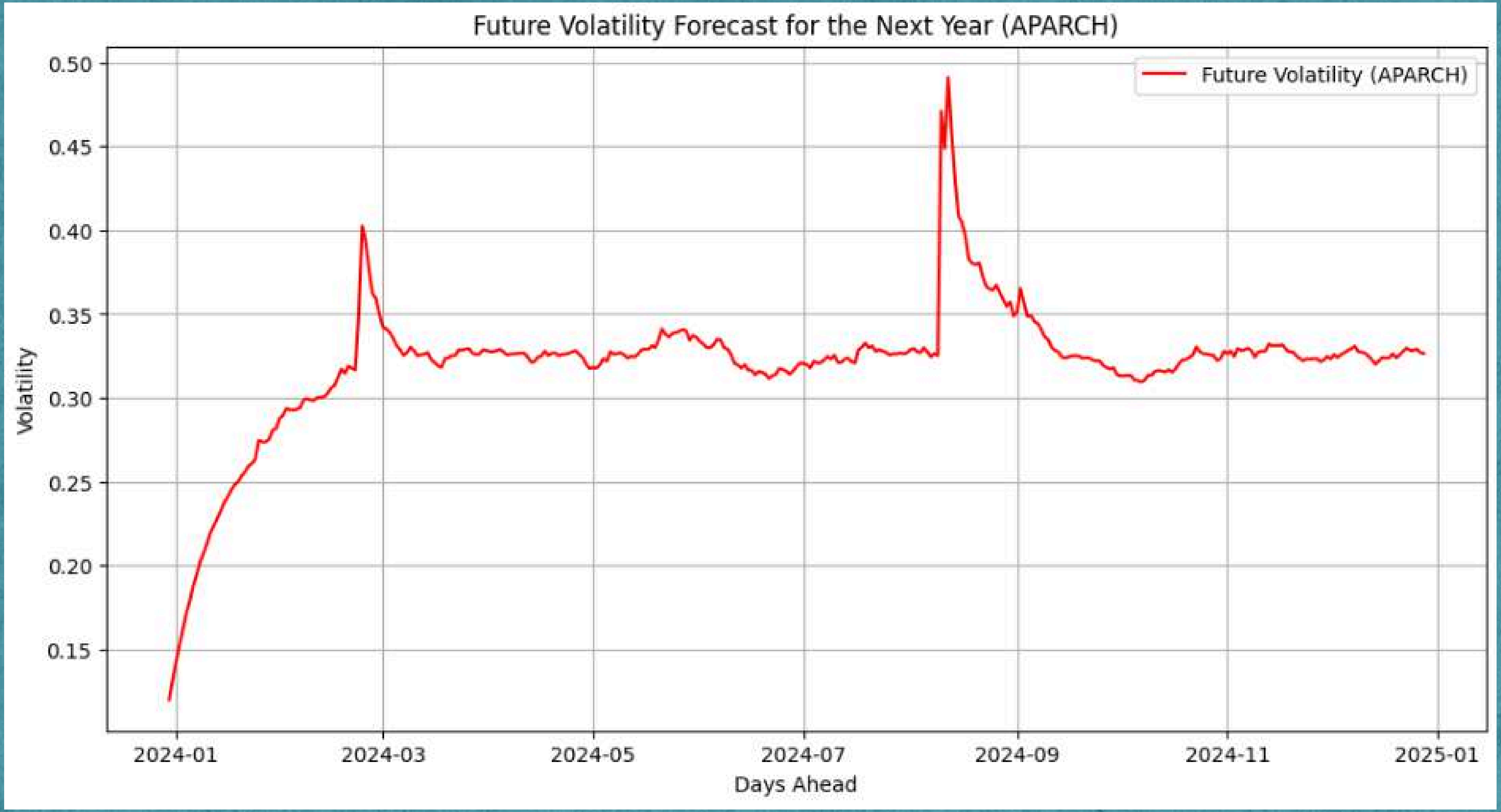
APARCH Model Applied

1. Mean Absolute Error (MAE): 0.34
2. Root Mean Squared Error (RMSE): 0.48



The APARCH model performs well, with slightly higher errors than GJR-GARCH.

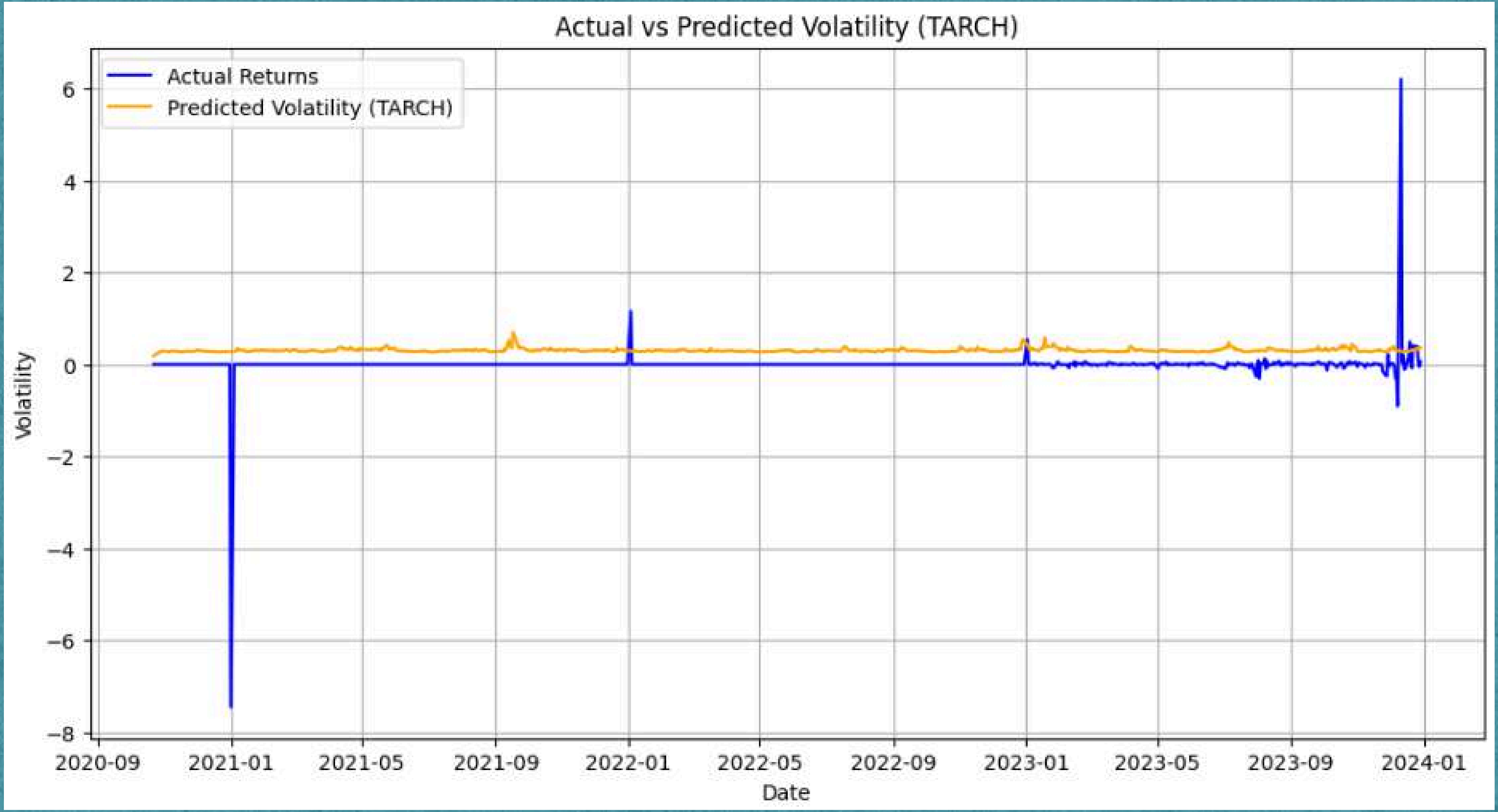
APARCH Model Evaluation



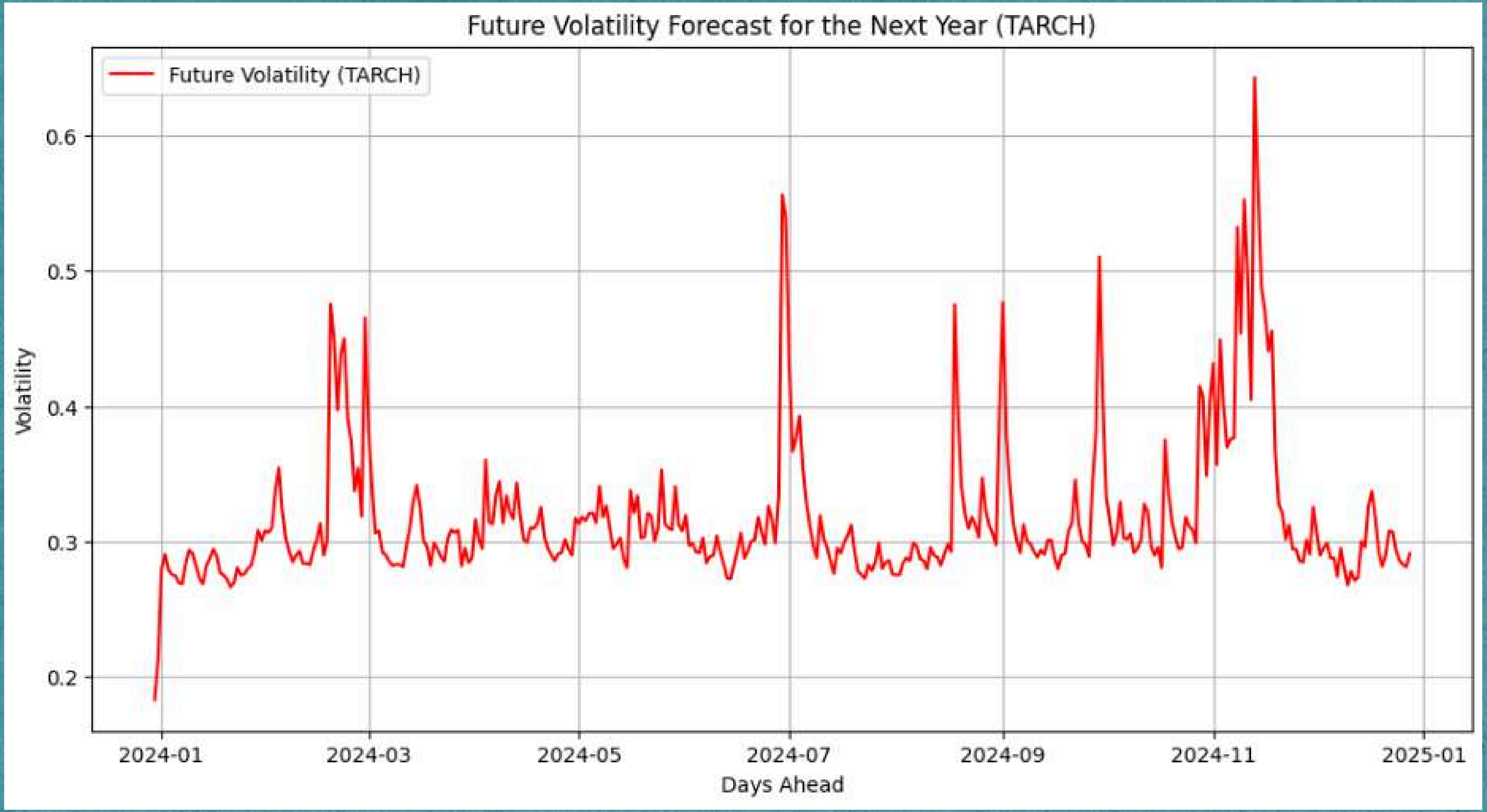
Volatility values between 0.15 and 0.50 are seen. A consistent decline to about 0.3 reflects predicted market stability over time, but spikes indicate potential external shocks or high-risk scenarios.

TARCH Model Applied

1. Mean Absolute Error (MAE): 0.32
2. Root Mean Squared Error (RMSE): 0.46



TARCH Model Evaluation



Volatility values from 0.20 to 0.45 suggest alternating periods of market stability and stress. Peaks around 0.45 might be linked to market-moving events, while values closer to 0.2 represent stable conditions.

Long Short-Term Memory

LSTM (Long Short-Term Memory) is a **Recurrent Neural Network (RNN)** type designed to model sequential and time-series data. Unlike traditional neural networks, LSTMs excel at capturing **dependencies** and **patterns** over time by maintaining a memory state

- RNNs are a type of **neural network** designed for sequential data, where the output at a given step depends on previous inputs. It is used for **time-series forecasting**, **Natural language processing** and **sequence-to-sequence prediction (e.g., stock price trends)**

Why LSTM for Time-Series Data

- **Volatility has long-term Dependencies:**
  - A shock (like a financial crisis/Covid) can have lingering effects on the market.

- Handling Sequential Data:
  - LSTMs process stock data sequentially, learning patterns and dependencies over time.

- Capturing Non-Linear Relationships:
  - Stock market data is non-linear and noisy. LSTMs are powerful enough to capture these complexities.

Key Components of LSTM

1. Cell State:
   - Acts as a memory that runs through the entire sequence, storing useful information and discarding irrelevant data.
2. Forget Gate:
   - Decides what information from the past should be forgotten.
3. Input Gate:
   - Determines which new information to store in the cell state.
4. Output Gate:
   - Decide what part of the memory to output at the current step.

Workflow:

1. Take input data at the current time step and the previous memory states.
2. Compute forget, input, and output gates using learned weights.
3 .Update the cell state and compute the output.

LSTM on HBL Stock Price Data

1) Data Normalization:
- Use MinMaxScaler to scale data to the range (0, 1), making it suitable for LSTM models. LSTMs perform better when input data is normalized.

2) Sequence Preparation:
- Create sequences of 60 timesteps (past 60 days of data) for each sample.

3) For each sequence:
- X: Includes the previous 60 days of features (e.g., price and volatility).
- y: Contains the volatility value on the 61st day (prediction target).

4) Split Data:
- Split the sequences into training and testing sets (80% for training, 20% for testing).
- This ensures the model generalizes well on unseen data.

5) Build the LSTM Model:
- First LSTM Layer: Captures temporal dependencies in the data, processes sequences, and outputs intermediate patterns. LSTM layer with 50 units (neurons). These neurons help capture patterns and temporal dependencies in the data.
- Dropout Layer: Prevents overfitting by randomly deactivating 20% of neurons during training.
- Second LSTM Layer: Extracts higher-order temporal features from the previous LSTM layer.
- Dense Layer: Outputs a single value (predicted volatility).

LSTM on HBL Stock Price Data

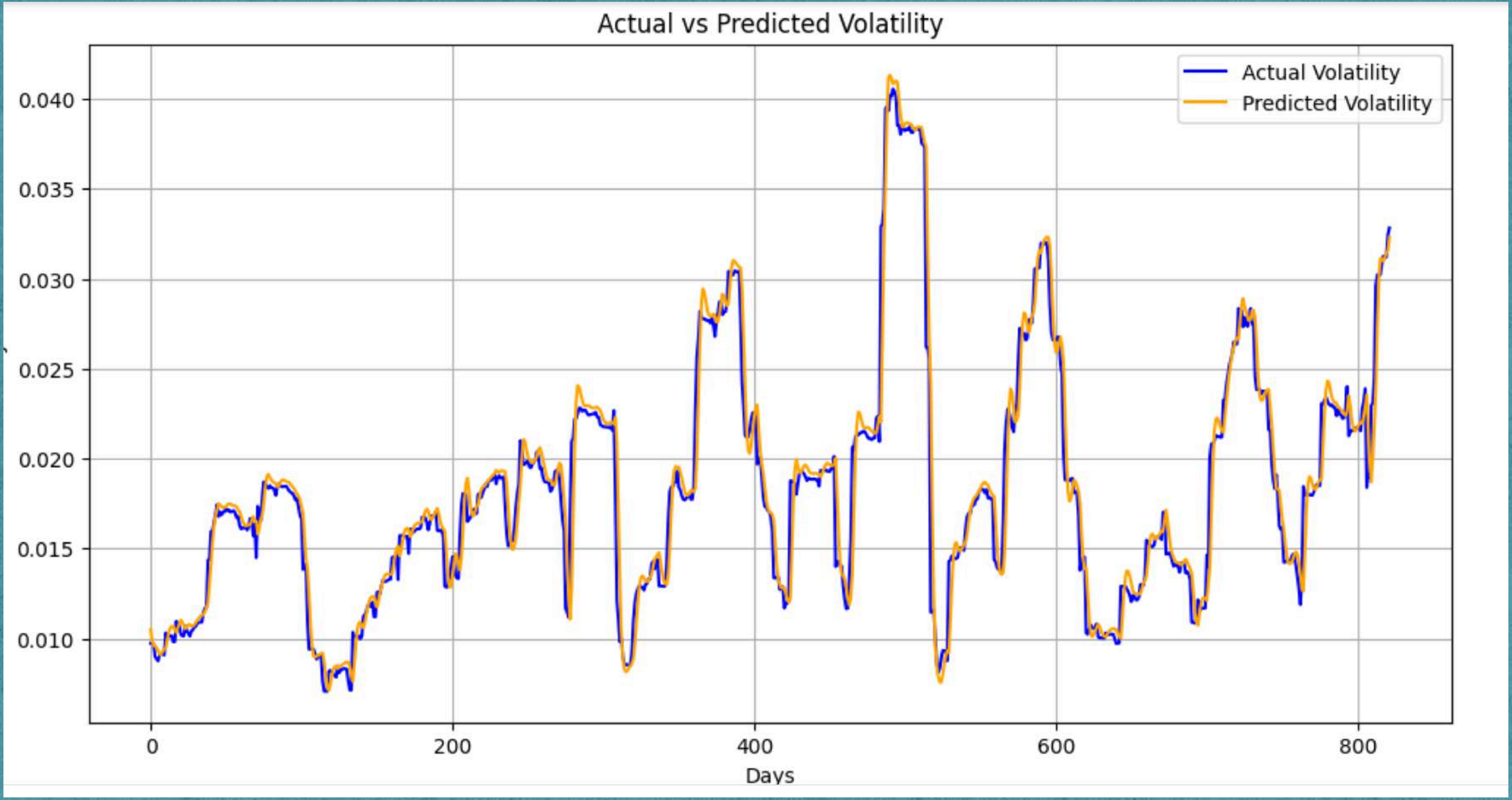
5) Compile the Model:
- Use the Adam optimizer, which dynamically adjusts learning rates for efficient training.
- Choose Mean Squared Error (MSE) as the loss function, suitable for regression tasks.

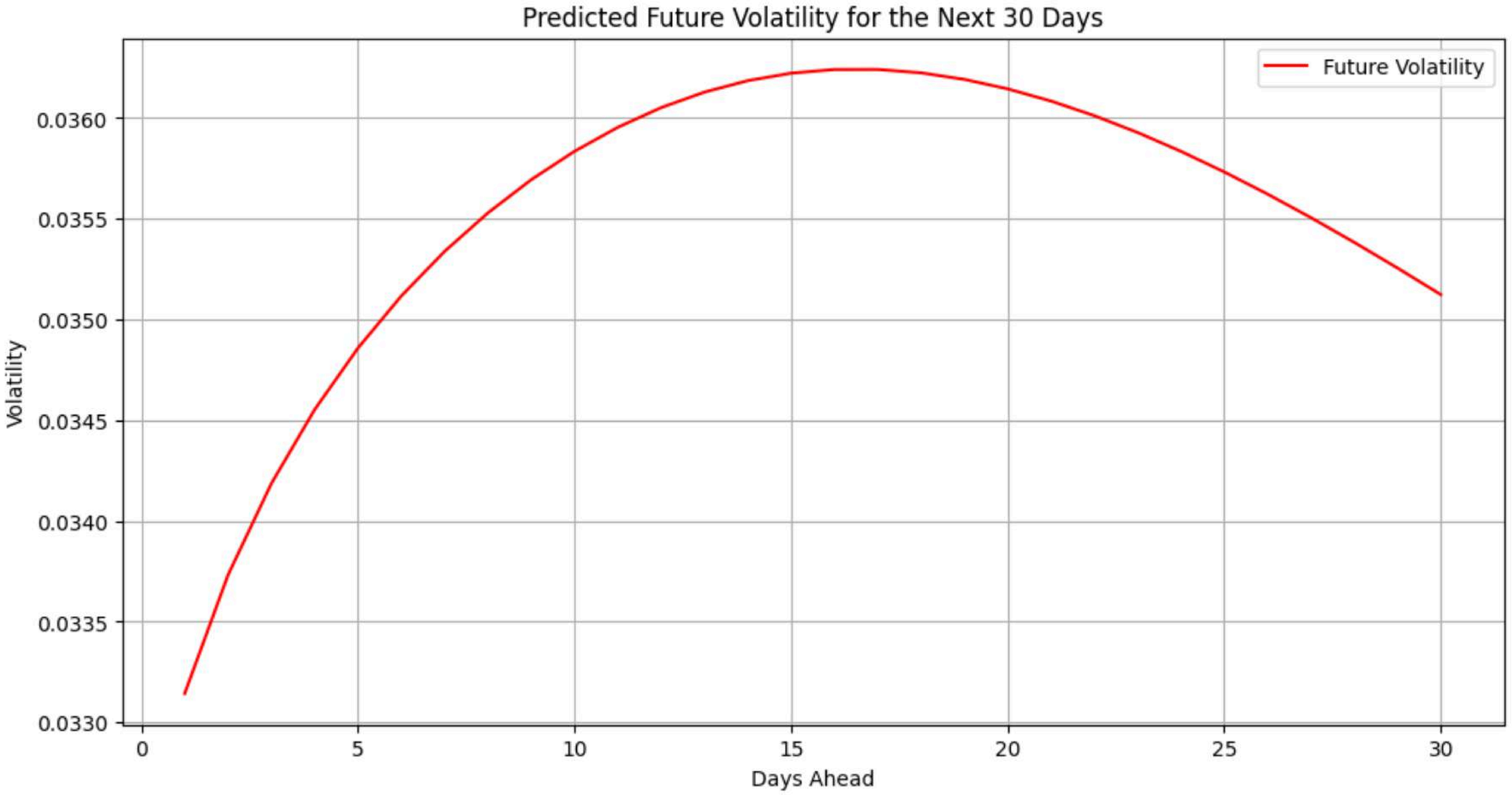
6) Compile the Model:
- Use the Adam optimizer, which dynamically adjusts learning rates for efficient training.
- Choose Mean Squared Error (MSE) as the loss function, suitable for regression tasks.

7) Train the Model:
- Fit the LSTM model to the training data using a batch-wise approach.
- Adjust weights during each epoch to minimize the loss function.

8) Predict Volatility:
- Use the trained model to predict volatility on the testing set.
- Denormalize predictions (if necessary) to compare with actual values.

Mean Absolute Error (MAE): 0.0009436009558425844
Root Mean Squared Error (RMSE): 0.0016275424537182854


Actual vs Predicted Volatility
Actual Volatility
Predicted Volatility
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0
200
400
600
800
Days


Predicted Future Volatility for the Next 30 Days
Future Volatility
Volatility
0.0360
0.0355
0.0350
0.0345
0.0340
0.0335
0.0330
0
5
10
15
20
25
30
Days Ahead