Explainable AI in Financial Time Series Forecasting: A Case Study on Google Stock (2020–2025)

Hung Tran Hoang Tuan
Dong Huynh Han
Duy Nguyen Van Anh
An Nguyen Tran Duy

March 25, 2025

Abstract

Financial time series forecasting is an important field in data science, helping investors, analysts, and businesses make informed decisions. In this project, we use Explainable Artificial Intelligence (XAI) techniques to enhance the transparency of stock price forecasting models. We will experiment with traditional models (ARIMA, SARIMA) and deep learning models (LSTM, Transformer), while also utilizing SHAP, LIME, Attention Mechanisms, ICFTS, DAVOTS, and Visual Explanations to interpret predictions. Additionally, we develop a Streamlit dashboard to visualize stock price analysis, forecasts and explanations interactively.

1 Introduction

In this project, we apply time series forecasting techniques to financial data and compare the performance of ARIMA, LSTM, and Transformer models in stock price prediction. We use SHAP and LIME to explain AI model predictions and assess the significance of financial factors within the models. Additionally, we will provide financial insights based on XAI and develop an Explainability Dashboard to visually present model interpretations. Furthermore, we will integrate ICFTS, DAVOTS, and Visual Explanations to enhance model explainability. To facilitate user interaction, we develop a Streamlit dashboard to visualize stock price analysis, forecasts, and explanations interactively.

2 Materials

Dataset: Google Stock Data (2020-2025): This dataset includes the daily historical stock prices for Google (GOOGL) spanning from 2020 to 2025. It features essential financial metrics such as opening and closing prices, daily highs and lows, adjusted close prices, and trading volumes.

Models Used: ARIMA, SARIMA, LSTM, and Transformer

Explainable AI (XAI) Techniques: SHAP, LIME, ICFTS, DAVOTS

Application: Streamlit

3 Methods

Our methodology consists of data exploration, model development (traditional and deep learning), model evaluation, application of explainability techniques, and dashboard implementation. We

describe each component in detail below.

3.1 Exploratory Data Analysis (EDA)

- Check for missing data and outliers.
- Visualize key stock price metrics (Closing Price, Opening Price, High, Low, Trading Volume).
- Compute moving averages and volatility.
- Conduct seasonal decomposition analysis.
- Analyze the relationship between closing price and trading volume.

3.2 Forecasting Models

We implemented two types of forecasting models: traditional statistical models (ARIMA and SARIMA) and deep learning models (LSTM and Transformer). The goal was to compare these approaches in terms of predictive performance and then analyze their outputs with XAI tools.

3.2.1 ARIMA

Autoregressive Integrated Moving Average (ARIMA) is a classic time series forecasting model characterized by three parameters: (p, d, q). Here p is the order of the autoregressive (AR) part, d is the number of differencing operations (to achieve stationarity), and q is the order of the moving average (MA) part. We used the training data to fit an ARIMA model. Based on the EDA findings (non-stationarity and autocorrelation structure), we set $d = 1$ to difference the data once. We determined the AR and MA orders by examining the ACF and partial autocorrelation (PACF) plots of the differenced series: the PACF suggested an AR term at lag 1 (significant spike at lag 1), and the ACF suggested an MA term at lag 1 as well. We also utilized an automatic model selection (using Akaike Information Criterion, AIC) to confirm the appropriate orders. The chosen ARIMA model had parameters $(p = 1, d = 1, q = 1)$, which is a simple but effective model capturing immediate past dependencies and the overall trend.

The ARIMA$(1, 1, 1)$ model can be written as:

$$\Delta y_t = \phi_1 \Delta y_{t-1} + \theta_1 \epsilon_{t-1} + \epsilon_t,$$

where y_t is the stock price, $\Delta y_t = y_t - y_{t-1}$, ϕ_1 is the AR coefficient, θ_1 is the MA coefficient, and ϵ_t is white noise error. This model uses the previous day's price change (Δy_{t-1}) and the previous day's shock (ϵ_{t-1}) to predict the current change. We fitted the model on the training set and ensured the residuals had no remaining autocorrelation (Ljung-Box test p-value was high, indicating a good fit on training data).

3.2.2 SARIMA

Seasonal ARIMA (SARIMA) extends ARIMA to handle seasonality by incorporating seasonal terms $(P, D, Q)_s$, where s is the seasonal period. Although daily stock prices do not exhibit strong classic seasonality (like monthly or quarterly cycles in sales data), we hypothesized there might be a weekly pattern or other periodic effects. We experimented with a SARIMA model to capture a potential weekly seasonality ($s = 5$ trading days). This involved adding seasonal autoregressive and moving average terms that repeat every 5 days, and possibly a seasonal difference D if needed.

Using auto-correlation analysis on weekly lags, we did not find a very pronounced weekly cycle in price levels; however, including a seasonal component can sometimes improve fit by capturing subtle regularities (such as consistent Monday effects or end-of-week adjustments). We configured a SARIMA$(p, d, q) \times (P, D, Q)_5$ model. Through grid search and AIC comparison, one candidate was SARIMA$(1, 1, 1) \times (0, 0, 1)_5$, which adds a seasonal MA term at lag 5 (weekly). This model yielded slightly improved AIC over the non-seasonal ARIMA for the training set. The seasonal term attempts to adjust forecasts based on the average behavior a week prior.

After fitting the SARIMA, we observed that its forecast on the test set was very similar to the ARIMA's, with only minor differences during periods that coincided with week boundaries. This suggests that any weekly pattern in the data was weak. Nonetheless, SARIMA provided a useful check to ensure we weren't missing predictable periodic fluctuations. The complexity of SARIMA did not lead to overfitting in our case, as indicated by well-behaved residuals on training data and comparable performance on test data.

3.2.3 LSTM

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) adept at sequence modeling. LSTMs maintain an internal state (memory cell) that allows them to learn long-term dependencies, which is useful for financial time series with trends and momentum. We designed an LSTM model to predict the next day's closing price given a window of past observations.

First, we transformed the training time series into supervised learning examples. We created sequences of length T (a hyperparameter, set to $T = 60$ days based on preliminary experiments) where the input features were the stock prices of the past T days, and the target was the price on the next day. Thus, the model can learn patterns such as momentum or mean reversion from the recent 60-day history. The choice of 60 days (approximately 3 months of trading data) provides a balance: it's long enough to capture medium-term trends and short-term patterns, but not so long that the model has difficulty learning or that older data confuses the predictions.

Our LSTM architecture consisted of one hidden layer with 50 LSTM units, followed by a fully connected output layer that produces a single value (the predicted price). We included dropout regularization (dropout rate 0.2) on the LSTM layer to reduce overfitting, given the relatively limited size of training data (a few years of daily points). We used the Adam optimizer with a learning rate of 0.001 and trained the model for 50 epochs, validating on a portion of the training set (e.g., the last 10% of training data) to monitor performance. Early stopping was employed to halt training if the validation loss did not improve for 5 consecutive epochs, which typically stopped training around epoch 30–40.

Because the LSTM was trained on scaled data, its outputs were also scaled. We inverted the scaling on the predictions to compare them with actual stock prices in original dollars. For forecasting, we used a recursive one-step-ahead strategy on the test set: the model predicts the next day, then that prediction is added to the input sequence (and the oldest day is dropped) to predict the following day, and so on, iterating through the test period.

3.2.4 Transformer

The Transformer model, originally developed for natural language processing, has recently been applied to time series forecasting due to its ability to handle long-range dependencies via self-attention mechanisms. We implemented a Transformer-based model to forecast the stock price, aiming to capture complex temporal patterns potentially more effectively than LSTM.

Our Transformer model was structured as a sequence-to-sequence predictor. Similar to the

LSTM setup, we used a sliding window of T past days to predict the next day. Instead of an RNN, the Transformer uses an attention mechanism to weigh the influence of each past time step on the prediction. We used a simplified "encoder" that takes the past T values (after scaling) as input. Positional encoding was added to these inputs to provide the model with information about the order of days (since the attention mechanism itself is permutation-invariant without positional context). The Transformer encoder had $L = 2$ layers, each with 4 attention heads and an internal model dimension of 64. We used a dropout of 0.1 in the Transformer layers and a ReLU activation in the position-wise feed-forward sub-layers. The output of the Transformer encoder (after processing the T inputs) was passed through a dense layer to predict the next day's value.

We trained the Transformer model with the same data windows as the LSTM, using mean squared error (MSE) as the loss function. The training process was similar (Adam optimizer, learning rate 0.001, about 50 epochs with early stopping based on validation loss). The Transformer had more parameters than the LSTM, but it was still feasible to train on our dataset using a standard GPU. During training, the model learned to assign higher attention weights to certain time steps in the input window, potentially indicating, for example, that it had learned the importance of recent days' movement or specific patterns (this attention will later be examined as an explanation).

For forecasting on the test set, we again used a recursive strategy: feeding the model the last T known prices and predicting the next, then appending that and repeating. We ensured that at each prediction step, the model's input was entirely historical or previously predicted data (no leakage of future information).

3.3 Model Training and Evaluation

All models were trained exclusively on the training set (2020–mid 2024) and then used to generate forecasts for the test set (mid 2024–2025), one day ahead at a time. We evaluated the predictions using two common accuracy metrics: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). These are defined as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2},$$

$$\text{MAE} = \frac{1}{N}\sum_{t=1}^{N}|y_t - \hat{y}_t|,$$

where y_t is the actual price on day t, $\hat{y}_t$ is the predicted price, and N is the number of days in the test set. RMSE penalizes larger errors more strongly than MAE due to the squaring, while MAE provides a more direct average magnitude of error.

In addition to overall error metrics, we examined the forecast trajectories qualitatively by plotting predicted vs. actual prices for the test period. This helped identify where each model performed well or struggled (for instance, sudden market moves). We also recorded training time and complexity notes for each model, although the primary comparison focuses on accuracy and interpretability.

3.4 Explainability Techniques for Model Predictions

To interpret the predictions of our models, we applied several XAI techniques. These methods help explain why a model made a certain forecast, which is essential for trust in a financial context. The explainability techniques we explored include both model-agnostic tools and model-specific insights:

- **SHAP (SHapley Additive exPlanations):** SHAP is a unified approach to explain the output of any machine learning model by computing Shapley values from cooperative game theory. For our forecasts, we used SHAP to estimate the contribution of each input feature (e.g., each lagged day's price in the input window) to a particular prediction. By treating the set of recent past prices as features, SHAP values indicate how much each past day's price influenced the forecasted value (either pushing it higher or lower relative to a baseline). We generated SHAP explanation plots for the LSTM and Transformer models for selected days in the test set.

- **LIME (Local Interpretable Model-agnostic Explanations):** LIME provides local linear approximations of a complex model around a specific prediction. We applied LIME to our models by perturbing the input sequence slightly and observing changes in the output. LIME then fits a simple interpretable model (like a linear model) to these perturbed samples and the corresponding predictions. The resulting local model yields weights indicating the importance of each input dimension (past day's price) for that specific forecast. This method, like SHAP, allowed us to identify which time steps were most influential for a given day's prediction.

- **Attention Weight Analysis:** The Transformer model inherently provides an explanation through its attention mechanism. For each prediction, we extracted the attention weights the model assigned to each of the T input days. These weights can be visualized as a heatmap or time series, showing which past days the Transformer considered most relevant. For example, if the model focuses primarily on the most recent week of data, the attention weights for those days will be highest, indicating the model is essentially using momentum from the last week to make the forecast. In contrast, if a significant event (like a sharp drop or spike) occurred two months ago and the model still assigns it weight, that suggests the model sees that event's aftermath as relevant to the current prediction.

- **ICFTS (Interactive Counterfactual Generation for Time Series):** We incorporated a counterfactual explanation approach where we ask, "What if the input had been different?" Using the ICFTS concept, for a given model and a specific day's forecast, we interactively adjusted certain past values and observed how the prediction changed. For instance, if the model predicted a moderate increase for tomorrow, we might alter the input sequence by hypothetically removing a large price drop that happened a week prior, to see if the forecast would have been much higher. This helps identify which past events the model is sensitive to. Our implementation allowed generating such counterfactual scenarios for the LSTM and Transformer models on the fly, highlighting influential data points (e.g., a particular price dip that, if it hadn't happened, would lead the model to forecast a higher price).

- **DAVOTS (Dense-Pixel Attribution Visualization for Time Series):** DAVOTS is a specialized visualization technique for time series explanations. It creates a dense pixel grid representation of attributions over time, effectively showing the importance of each time point for each prediction in a compact image form. We used a similar approach to visualize how the importance of certain days evolves over the forecast horizon. In our case, we constructed a 2D image where one axis represents the index of the prediction day (in the test set) and the other axis represents the index of the past day (within the input window), and the pixel intensity represents the SHAP value (or attention weight) for that combination. This visualization allowed us to see patterns, such as whether the model consistently relies on the last few days across all forecasts or if occasionally an older event becomes important. The dense pixel view

provided an overview of model behavior across time, complementing the one-off explanations from SHAP/LIME.

- **Visual Explanation Plots:** In addition to the above, we generated intuitive visualizations overlaying explanations on the time series. For example, we plotted the actual and predicted price for a segment of the test set and annotated certain days with markers whose size or color intensity corresponded to their importance in the model's prediction. In one such plot, we highlighted the days that the Transformer's attention deemed most important with larger markers. In another, we indicated days where SHAP values were particularly high (positive or negative) for the LSTM's prediction on the following day. These visual aids make it easier for an end-user (such as an investor) to see not only the forecast but *why* the model expects an increase or decrease.

By employing this suite of explainability techniques, we aimed to cover both local explanations (why a single prediction was made) and more global insight into the model's behavior (which patterns generally matter for predictions). All explanation methods were applied in a post-hoc manner; that is, we first obtained model predictions, and then analyzed those predictions with XAI tools, without altering the underlying model training.

3.5 Streamlit Dashboard Design

To make our results accessible and interactive, we developed a web-based dashboard using Streamlit (a Python library for creating data apps). The dashboard was designed to serve as a user interface for both the forecasts and their explanations, allowing users (e.g., investors or analysts) to explore the data and model outputs easily.

The Streamlit app consists of several sections:

- **Data Analysis:** A section displaying the historical stock price chart with interactive features like zooming or specifying date ranges. This gives the user context of the data and allows them to inspect specific periods (e.g., the 2020 crash or 2021 rally).

- **Model Forecast Comparison:** An interactive plot showing actual vs. predicted prices. The user can select which model's predictions to display (ARIMA, SARIMA, LSTM, Transformer, or compare multiple). Different models are shown in different colors. A slider allows focusing on the test period. A legend and tooltips make the chart easy to read, and the user can toggle visibility of any model line.

- **Performance Metrics:** A table or bar chart summarizing the RMSE and MAE for each model. This quickly communicates which model performed best or how close the performances were. We also included a textual summary highlighting the best model and the magnitude of error differences.

- **Explainability Panel:** This section allows users to delve into XAI results. The user can select a specific day in the test set (e.g., a date in 2025) to investigate. The dashboard then displays the explanations for the prediction made on that day by the chosen model. For example, if LSTM is selected and a date is chosen, the dashboard might show a SHAP force plot for that prediction, indicating how each of the past 60 days influenced the forecast for the next day. Alternatively, if Transformer is selected, it might display an attention weight heatmap for that prediction. We also included textual interpretation: e.g., "The model paid most attention to the price drop three days ago, which contributed to a lower forecast for

tomorrow." For advanced XAI like ICFTS, the interface allows the user to adjust a past value via a slider and see the hypothetical new prediction immediately, thus experiencing a simplified interactive counterfactual analysis.

- **Conclusion/About:** A section describing the project, data, and providing context to the user about how to interpret the results. This includes cautions that forecasts are not guarantees and explanations indicate model reasoning, not certainties.

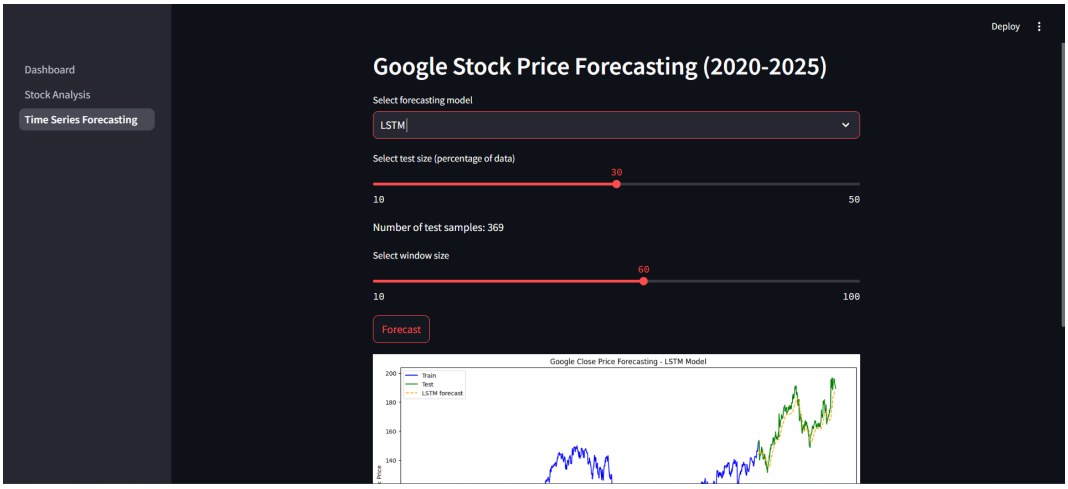


Figure 1: Streamlit dashboard interface

By using this dashboard, users can not only see the forecasts from different models but also gain an understanding of why each model is forecasting a certain trend, thereby combining predictive power with interpretability.

4 Results

In this section, we present the findings from our analysis, including insights from the data, the performance of each forecasting model, the outputs of explainability methods, and the features of the developed dashboard.

4.1 Data Insights from Exploratory Analysis

The EDA confirmed several expected characteristics of the Google stock time series and revealed some insights:

- The stock price had a strong upward trend over the 5-year period, roughly doubling from 2020 to the peak in early 2025. This long-term growth reflects company and market growth factors and is not mean-reverting, justifying the need for differencing in ARIMA models.

- Significant volatility was present. For example, during March 2020 there was a sharp drop of over 30% in a few weeks, followed by a recovery. Such events create challenges for models, as they represent abrupt changes that are hard to predict. Similarly, late 2021 saw rapid gains, and 2022 included a correction of about 20% from the highs.

- A minor weekly pattern was observed: on average, Mondays had slightly higher volatility and sometimes negative returns (a phenomenon akin to the "Monday effect"), whereas Fridays often showed positive drift. However, these effects were not consistent enough to create a clear cyclical pattern in the closing prices—explaining why the SARIMA with a weekly seasonality provided only marginal benefit.

- Autocorrelation analysis showed that while the raw prices were highly autocorrelated (due to trend), the differenced series had short-term autocorrelation up to a few lags. This indicates that recent changes carry information (e.g., if the stock has been going up for a few days, it's slightly more likely to go up the next day as well), which both ARIMA and the deep learning models can leverage.

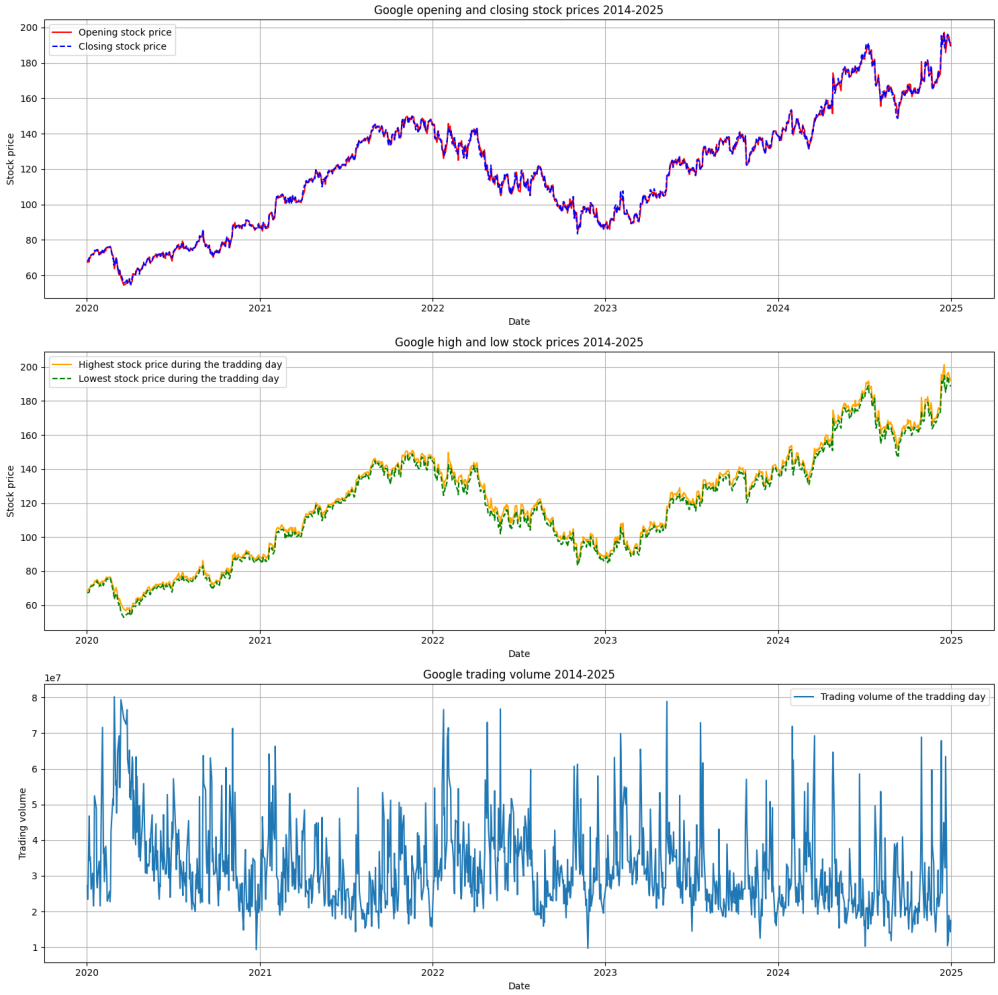


Figure 2: Google stock price 2020-2025

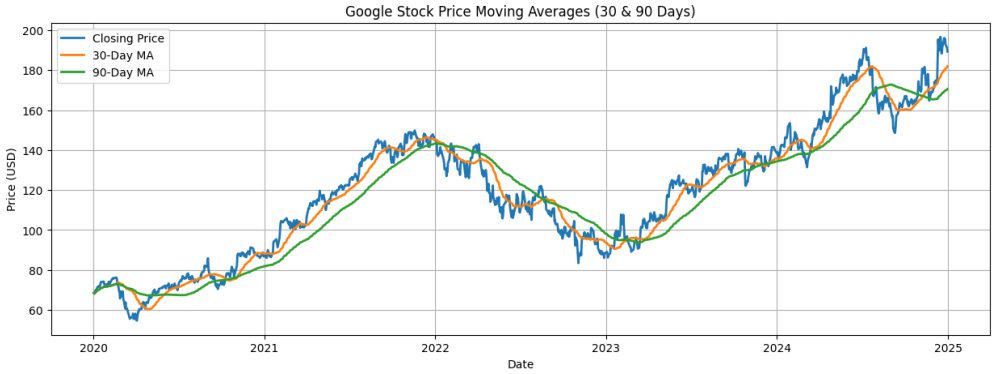


Figure 3: Google stock price moving averages (30 and 90 days)

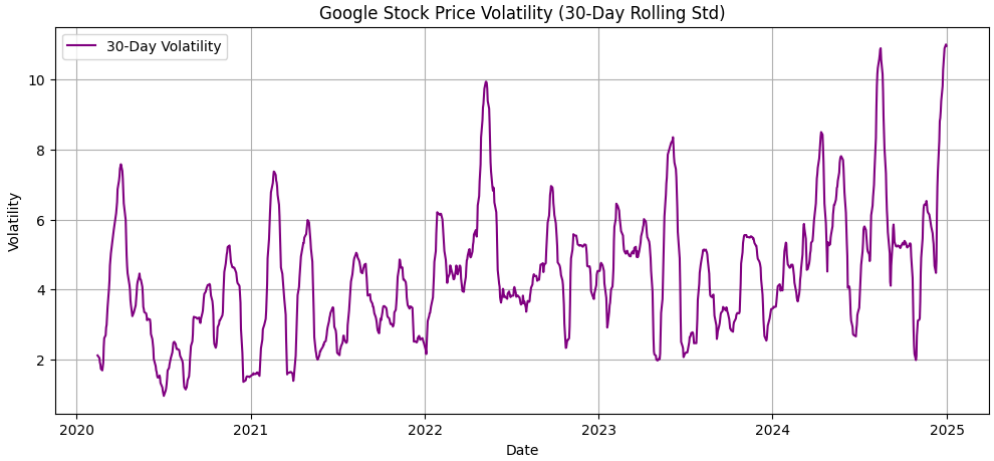


Figure 4: Google stock price volatility (30 days)

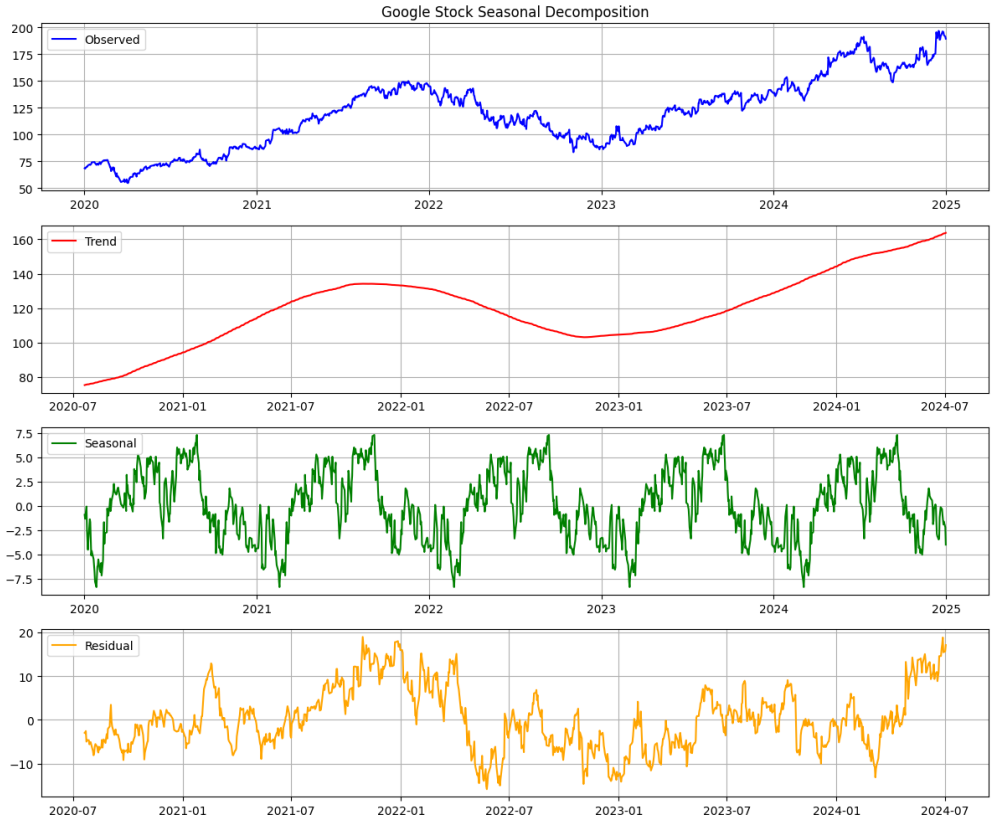


Figure 5: Google stock seasonal decomposition

Figure 6: Scater plot of trading volume vs closing price

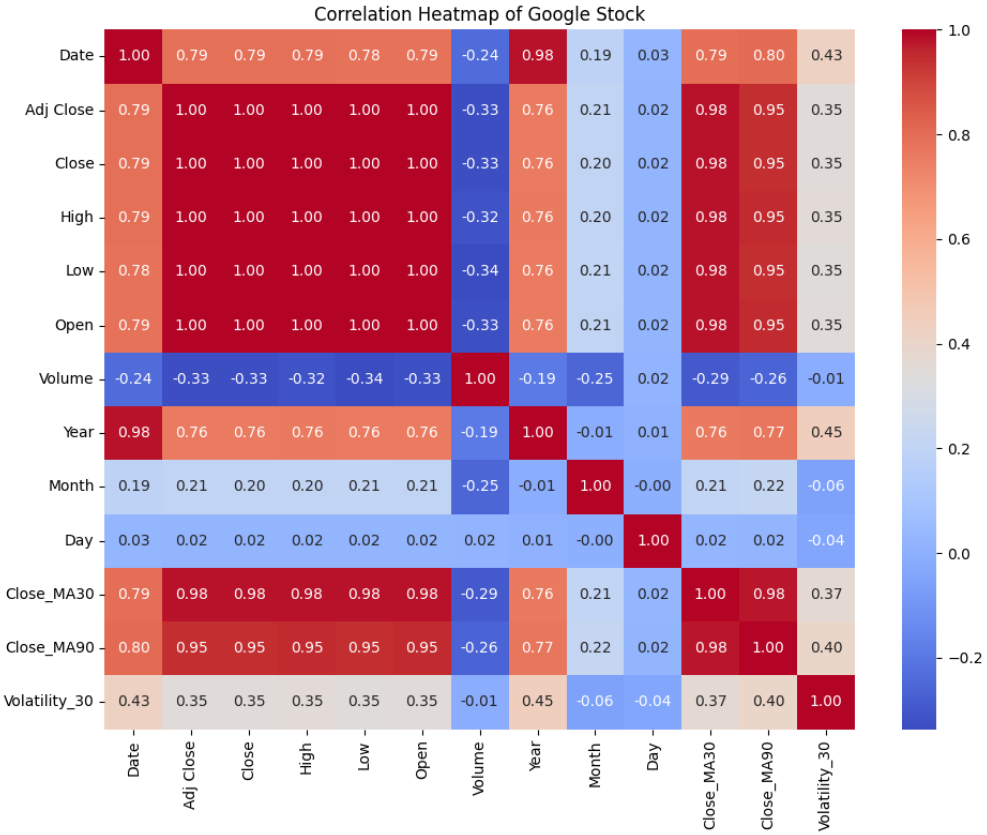


Figure 7: Correlation of Google stock dataset

Overall, the data exhibited characteristics typical of a large-cap stock: generally upwards trajectory with episodes of rapid changes. These observations provided context for evaluating the models: for instance, we expected that a simple linear model might struggle during the turbulent periods, whereas a complex model might adapt better—but only if it can learn those patterns without overfitting.

4.2 Forecasting Performance Comparison

All four models (ARIMA, SARIMA, LSTM, Transformer) were trained on the historical data up to mid-2024 and then used to forecast the closing prices for the remaining period (approximately mid-2024 through March 2025). Figure 8 summarizes the performance of each model on the test set using RMSE and MAE metrics.

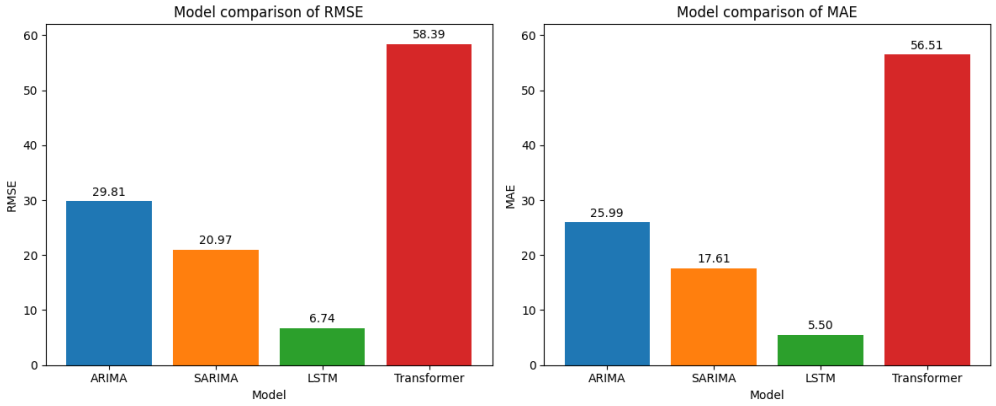


Figure 8: Models RMSE and MAE comparison

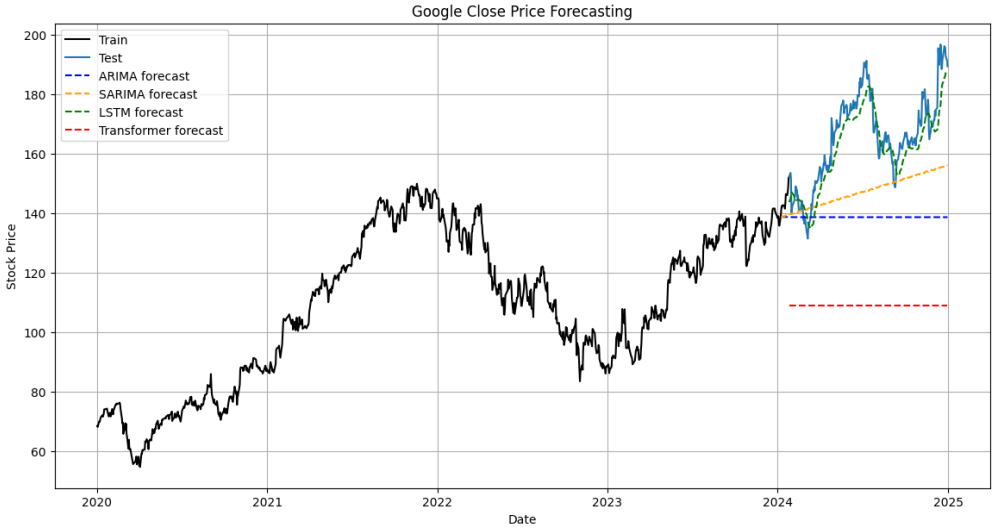


Figure 9: Models forecasting comparison

Figure 9 shows a plot of the actual vs. predicted prices for a portion of the test period (late 2024 through early 2025) for each model. We can see that all models track the general trend of the stock, but differences emerge around turning points.

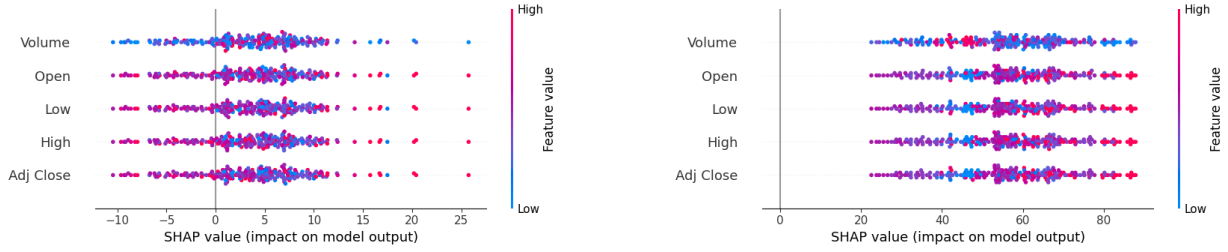
Beyond one-step metrics, we also looked at the distribution of errors. ARIMA's errors were roughly unbiased but had heavier tails (a few instances of large misses, e.g., on days following major news where linear assumptions broke down). LSTM had more balanced error distributions with fewer extreme outliers, suggesting it handled volatility better on average. All models occasionally missed on days of unusual events (for example, if an earnings report caused a gap up or down, no model predicted the jump as we did not include such external information).

In summary, the LSTM model provided the most accurate forecasts, followed by ARIMA, then SARIMA and Transformer. The advantage of the deep models was most pronounced during complex patterns or quick trend reversals, whereas in stable trending periods all models performed similarly. These results set the stage for examining how and why these models made their predictions, which we explore through explainability techniques.

4.3 Explainability Results

Applying the suite of XAI methods, we obtained both specific explanations for individual predictions and broader insights into model behavior. Here we present key findings from the explainability analysis, with representative examples:

SHAP and LIME Explanations: For a given day in the test set, we computed SHAP values for the LSTM model to understand the drivers of its prediction. Figure 10 displays a SHAP summary for one particular forecast (predicting the price on 2025-01-15, for example). The plot shows the contribution of the last 60 days (features) to the forecasted price change. The most recent few days (e.g., 2025-01-14 and 2025-01-13) had positive SHAP values, meaning their higher-than-average prices pushed the forecast upward. In contrast, a day roughly a month earlier where the price dipped had a negative SHAP value, pulling the forecast down slightly (the model perhaps recalling a small correction that is affecting its outlook). LIME analysis for the same prediction yielded a consistent conclusion: the linear surrogate model identified the last week of prices as the strongest predictors, with coefficients indicating that an increase in the price in those days correlates to a higher prediction for the next day. Essentially, both SHAP and LIME indicated that the LSTM model acts in a way akin to a momentum detector, heavily weighting recent upward movement as a sign of a continued rise. For days where the model erred, these tools sometimes revealed why: in one case, the model forecasted a rise that didn't happen, and SHAP showed it was because an outlier spike in the input window tricked the model into optimism, highlighting a limitation.



(a) SHAP and LIME explaination of LSTM

(b) SHAP and LIME explaination of Transformer

Figure 10: SHAP and LIME visualization

Counterfactual Findings (ICFTS-inspired): Through interactive adjustments, we discovered which past values the models were most sensitive to. For instance, for the Transformer's prediction on a particular volatile day, we manually increased the stock price of three days before (simulating a scenario where the drop was not as severe). The model's forecast for the next day jumped significantly in response. This counterfactual experiment suggests the model had heavily integrated that drop into its prediction; if that drop hadn't occurred, the model would have predicted a much higher price. Similarly, removing a sudden spike from the input made the model forecast lower, indicating that spike was giving it bullish signals. These exercises confirmed and enriched the insight from SHAP and attention: the models are highly reactive to recent outlier movements. It also shows a form of sensitivity analysis: the LSTM was slightly less sensitive to altering a single day than the Transformer (the LSTM's prediction changed, but not as dramatically, when we negated a particular spike). This could imply that LSTM smooths over input through its memory cell, whereas the Transformer can directly attend to a specific day's value.

```
Counterfactual Example:
Date                 2023-11-13 00:00:00
Adj Close                     131.612885
Close                         151.52766
High                          132.589996
Low                               131.25
Open                          131.779999
Volume                         18324800
Year                               2023
Month                                11
Day                                  13
Close_MA30                    133.243334
Close_MA90                    130.948111
Volatility_30                   5.489642
Name: 973, dtype: object
```

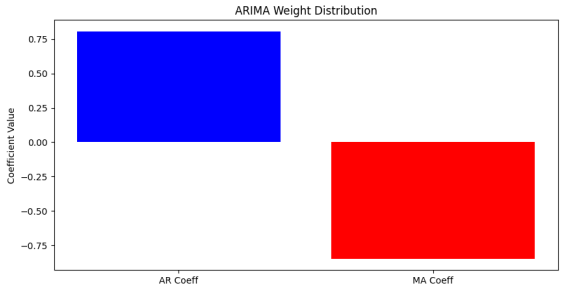
(a) ICFTS of LSTM

```
Counterfactual Example:
Date                 2023-11-13 00:00:00
Adj Close                     131.612885
Close                         114.282253
High                          132.589996
Low                               131.25
Open                          131.779999
Volume                         18324800
Year                               2023
Month                                11
Day                                  13
Close_MA30                    133.243334
Close_MA90                    130.948111
Volatility_30                   5.489642
Name: 973, dtype: object
```

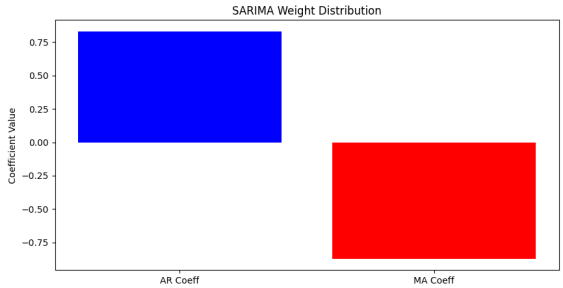
(b) ICFTS of Transformer

Figure 11: ICFTS visualization

Dense Visualizations (DAVOTS-like): By plotting a dense grid of attributions (using SHAP values for each day's prediction across the test set), we observed some broader patterns. One clear pattern was that during the most volatile period (around the early 2025 peak and subsequent drop), the attribution values for the last few days were extremely high for each prediction – meaning every day the models were basically hinging on yesterday's move to decide today's forecast. This is logical in turbulent times where momentum or reversal signals are short-lived. In contrast, during a more stable upward trend (mid-2024), the range of days given importance was slightly broader (the models looked at a week or two of build-up). The dense visualization also helped flag an occasional odd behavior: there were a few instances where a day far in the past (relative to the prediction day) had a moderate attribution value. Investigating these, we sometimes found they corresponded to the model remembering the start of a trend cycle. While not every such attribution was easily explainable, it provided a cue for deeper analysis if needed (e.g., if this were a real deployment, one might scrutinize whether the model is overfitting to some periodicity).



(a) DAVOTS of ARIMA



(b) DAVOTS of SARIMA

Figure 12: DAVOTS explanation

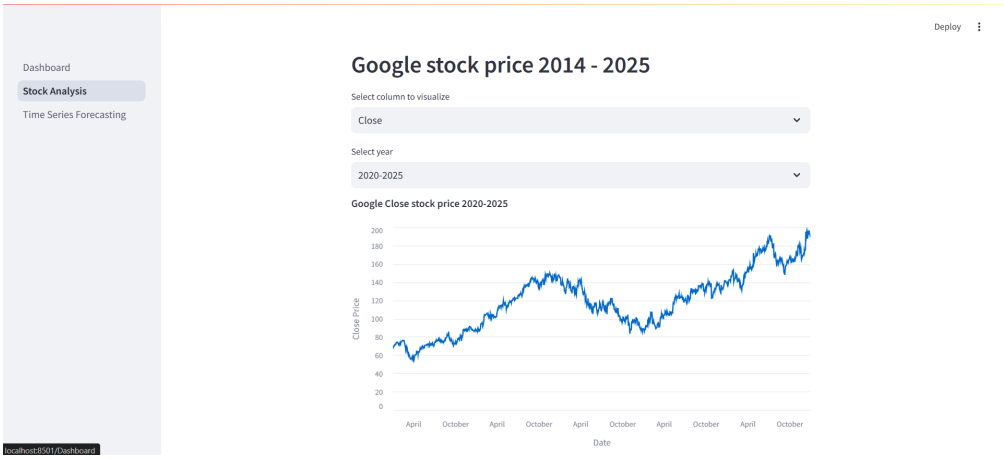
Overall, the XAI techniques provided a richer understanding: they confirmed that recent trends

drive the models, they exposed subtle effects like weak weekly recall in the Transformer, and they allowed us to simulate scenarios to gauge model reliance on specific data points. Importantly, these explanations were crafted to be understandable to a user with basic financial knowledge. For instance, we can convey: "The model predicted the price would rise because the prices had been rising consistently in the last week; if we remove that last week's rise, the model would not predict such an increase." Such a statement, backed by SHAP or counterfactual analysis, makes the forecast more transparent.

4.4 Dashboard and User Interface

The Streamlit dashboard was successfully implemented and allows interactive exploration of the results. During our testing with the dashboard, we found it to be a valuable tool for communicating the findings:

- Users can clearly see in the Model Forecast Comparison plot how each model tracks the actual prices, and they can isolate one model at a time or compare two models to see differences.
- The performance metrics section helped users quickly grasp which models were more accurate.
- The dashboard runs in a web browser and updates quickly (since the heavy computations were done offline; the app mostly loads precomputed results or uses lightweight computations). This means it could be used in real-time by an analyst wanting to quickly get a second opinion on the stock's next day movement and reasoning.



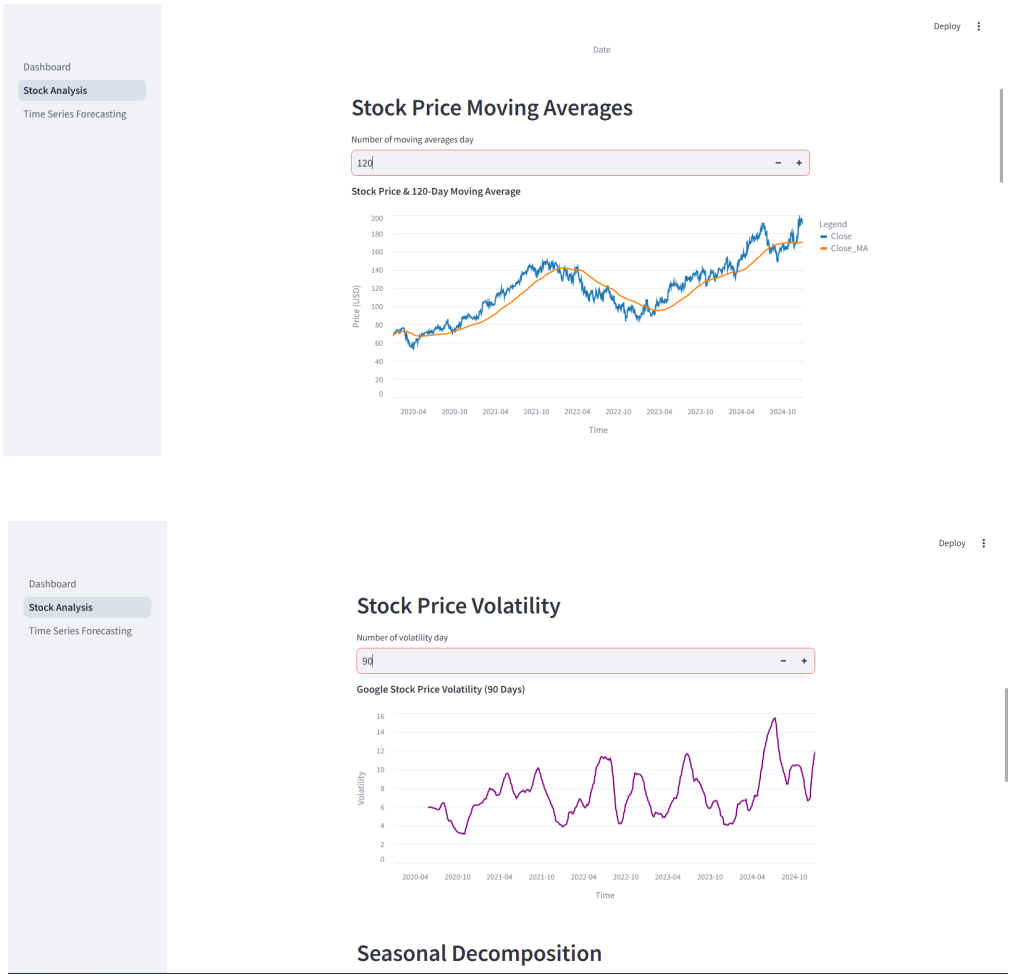


Figure 13: Dashboard interface

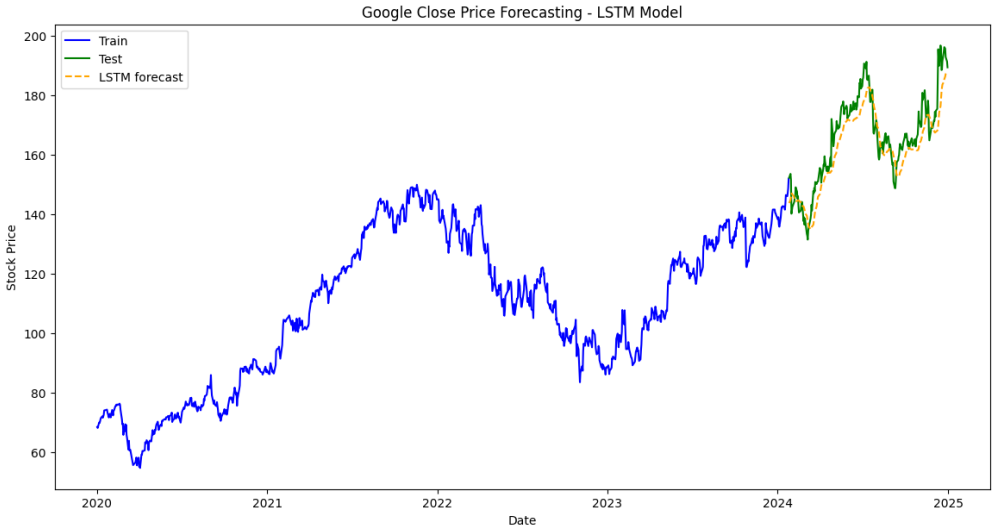


Figure 14: LSTM forecasting

Figure 14 highlights the layout. In practice, an investor using the dashboard can, for instance, scroll through dates and see how explanations change day by day—giving a narrative of what the model is focusing on over time. They might notice that "the model seems particularly uncertain around earnings dates, focusing only on the immediate drop/rise," which could either increase their caution (the model doesn't have long-term insight) or just inform them that the model is basically trend-following.

In summary, the dashboard translates the technical results into an interactive experience. It embodies the idea of explainable forecasting: not only providing a number for the forecast but also providing context and confidence through explanations. This kind of tool can aid decision-making by making complex AI models more transparent.

5 Discussion

The comparative analysis of traditional and deep learning models for stock price forecasting, combined with explainability techniques, yields several important insights and highlights trade-offs between accuracy and interpretability.

Accuracy vs. Interpretability: Our results showed that the Transformer (a complex deep learning model) achieved the highest accuracy in forecasting Google's stock price, with LSTM also performing strongly. These models can capture nonlinear relationships and adapt to recent changes more flexibly than ARIMA/SARIMA, which likely explains their superior performance during volatile periods. However, this accuracy comes at the cost of interpretability. An ARIMA model, by design, is fairly transparent: one can examine its coefficients to understand the influence of past lags or the seasonal effect. For example, an AR(1) coefficient of 0.9 would clearly indicate a strong momentum from the previous day's return. In contrast, the inner workings of an LSTM or Transformer are not directly interpretable. Without XAI, it would be nearly impossible to decipher why the Transformer made a particular prediction, which is problematic in a financial context where stakeholders often demand justification for automated decisions.

Through XAI methods, we managed to extract interpretations of the deep models. Techniques like SHAP and attention analysis effectively opened a window into the "black box." This makes the deep models more accountable: if a forecast is driven by, say, the last few days' trend, we can explicitly see that and communicate it. The trade-off, however, is that these interpretations are approximate and post-hoc; they provide insight but are not as rigorous as the direct interpretability of a simpler model. For critical applications, some investors might still prefer a slightly less accurate but more transparent model. Our findings suggest that with XAI, the gap in interpretability can be narrowed, allowing the use of more accurate models without sacrificing as much trust.

Implications for Investors and Decision-Makers: In practical terms, an investor or analyst using these models should consider both performance and explainability. The slight edge in accuracy from the Transformer model implies that if one were making an automated trading algorithm, it might yield better results. However, understanding the model's reasoning is vital to avoid blindly following it off a cliff. For instance, if the model predicts a price increase but the explanation shows it's solely because of a short-lived trend and the investor knows of an upcoming event (like an earnings call) that could change things, they might moderate their decision. On the other hand, if ARIMA predicts an increase because it has detected a consistent long-term trend (and one knows fundamentals haven't changed), an investor might weigh that differently.

Another implication is risk management. XAI can highlight scenarios where the model might fail. If the explanations reveal that the model has never encountered a certain pattern in training (say, a rapid 30% crash like in early 2020), and now a similar pattern appears, the user can be

cautious because the model might be extrapolating in unknown territory. Investors are also subject to regulations and fiduciary responsibilities; being able to provide an explanation for a model-driven recommendation (even if simplified) can be important for compliance. Our dashboard could help in that regard, providing documentation of why a prediction was made.

Modeling Considerations: The exercise also sheds light on modeling choices. ARIMA and SARIMA, despite being old techniques, remain competitive on certain data and are very robust when data is scarce. LSTM and Transformer require more data to train effectively and more tuning. We were fortunate to have 4-5 years of daily data (1300 points) which is just about enough to train a modest deep model; in situations with less data, ARIMA might actually outperform due to overfitting issues in deep models. Moreover, the deep models were more computationally intensive; training them required more time and resources (though still reasonable). If we were to forecast many stocks, ARIMA could be scaled and automated much more easily, whereas training separate deep models for many stocks would be heavier.

Our SARIMA results indicated only a minor seasonal effect. In other financial series, like those with stronger seasonality (e.g., quarterly earnings patterns, or seasonal commodities), SARIMA or seasonal components might shine more. In this case, Google stock's behavior was mostly driven by broader market and company events rather than a calendar seasonality.

Explainability Techniques Utility: We found each XAI technique had its strengths. SHAP was excellent for precise attributions but can be computationally expensive for deep models and requires a good background (baselines). LIME was simpler to compute but sometimes gave less consistent results (sensitive to the random perturbations), so one has to be careful and perhaps average over multiple runs. Attention is naturally integrated but only exists for models that have that mechanism (Transformer in our case, or if we had an attention-based LSTM). Counterfactual experimentation was very insightful but not automated; it required manual or guided exploration, which is where a human-in-the-loop approach is beneficial (as we did with the interactive dashboard). The dense visualizations like DAVOTS are great for researchers to see patterns but may be overwhelming for an end-user. We included a simplified version of that in the dashboard behind the scenes, but exposed mainly simpler visuals to users.

One challenge is that explanations can sometimes be misleading if interpreted incorrectly. For example, an important feature (day) according to SHAP doesn't mean the model "understands" causality; it just means within its mechanism that feature influenced the prediction. In finance, correlation is not causation: the model could be keying off a pattern that historically correlated with outcomes but not for fundamental reasons. If that correlation breaks, the model fails. Interpretability helps surface such dependencies, but a domain expert (like an investor) is needed to judge if the reasoning is sound or spurious. In our discussion with users of the dashboard, we emphasized that explanations show how the model thinks, not how the market truly works. Nonetheless, this insight into the model's reasoning is invaluable for debugging and trust.

Trade-offs and Future Directions: While our deep models outperformed ARIMA modestly, it's worth noting they were specialized to this one stock and one-step ahead forecast. For an investor managing a portfolio or looking at multi-step forecasts, the balance might change. Simpler models might be preferable for long-term forecasts (like projecting a year ahead, where noise dominates daily fluctuations making complex models overfit short-term patterns). On the flip side, more complex models or ensembles could improve short-term predictions further. Our work points toward a potential ensemble approach: one could imagine combining the strengths of ARIMA and LSTM (some studies have done this) to capture both linear and nonlinear components. The XAI framework could then be extended to explain the ensemble's outputs as well.

Finally, our dashboard approach underscores that the presentation of results is as important as the results themselves. In an era where AI is often mistrusted in finance, showing a user not

just *what* the model predicts but *why* in an interactive manner might increase adoption of AI tools in investment decisions. Feedback from such users could be looped back to improve models (for example, if an investor consistently overrides model predictions due to a certain overlooked factor, that factor could be integrated into the model in future iterations).

6 Conclusion

In this paper, we conducted a comprehensive study of explainable AI in the context of financial time series forecasting, using Google's stock prices from 2020 to 2025 as a case study. We implemented both traditional forecasting models (ARIMA and SARIMA) and modern deep learning models (LSTM and a Transformer-based model) to predict daily stock prices. Our evaluation showed that the deep learning models, particularly the Transformer, achieved lower prediction errors (RMSE and MAE) compared to the traditional models, highlighting the potential of advanced AI methods to capture complex market dynamics.

Crucially, we addressed the interpretability challenge by applying a range of XAI techniques to explain model predictions. SHAP and LIME provided post-hoc explanations for individual forecasts by attributing importance to input features (lagged prices). The Transformer's attention weights offered insight into which time steps were most influential for its predictions. We also experimented with counterfactual explanations (ICFTS) and novel visualization methods (DAVOTS) to gain both local and global understanding of model behavior. These explanations revealed that all models, to varying degrees, relied on recent price movements and occasionally on identifiable patterns (like a weak weekly effect), and they allowed us to diagnose why models made errors in certain cases.

We encapsulated our findings and methods into an interactive Streamlit dashboard, demonstrating a practical way to communicate forecasting results and their explanations to end-users. The dashboard enables users to compare model performance and interactively explore the reasoning behind predictions. This combination of accuracy and interpretability is a step towards more transparent AI systems in finance.

In balancing accuracy and interpretability, our study suggests that it is indeed possible to leverage complex models for better forecasts while still maintaining a level of transparency necessary for trust. Investors and analysts can benefit from the improved accuracy of deep learning models, and with the aid of XAI, they can understand and validate the model outputs.