DATA ANALYTICS INTERNSHIP PROJECT REPORT

TIME SERIES ANALYSIS AND FORECASTING FOR STOCK MARKET

SUBMITTED BY
G.S. Harshitha
Batch : 14.1
Internship Duration: 1 month
Email: harshithagiltus@gmail.com

APRIL 17, 2025 - MAY 17, 2025

Abstract

Predicting stock prices is a difficult but important undertaking in financial markets, with big prospects for firms and individuals. Four well-known time series forecasting models—Autoregressive Integrated Moving Average (ARIMA), Facebook Prophet and Long Short-Term Memory (LSTM) networks are thoroughly compared in this report. The theoretical foundations, advantages, and disadvantages of each stock prediction model are examined. The approach is described in full, including data preparation, model implementation, and assessment measures. The findings illustrate the models' applicability for various aspects of financial time series and show how differently they perform on historical stock data. This study offers insightful information for choosing a suitable forecasting strategy depending on the characteristics of the data and the goals of the prediction.

1. Introduction:

Economic data, corporate performance, geopolitical events, and market mood are just a few of the many variables that affect the stock market, which is a dynamic and intricate system. A competitive advantage in risk management, algorithmic trading, and investment decisions can be gained by accurately forecasting stock prices. However, this is a difficult operation due to the inherent volatility, non-stationarity, and non-linearity of stock data. For a long time, time series forecasting has been done using conventional statistical models like ARIMA.

1.1 Problem Definition:

The stock market is a highly dynamic and unpredictable system shaped by a variety of factors, including economic trends, company performance, political developments, and investor psychology. This complexity makes forecasting stock prices a challenging task, especially given the non-linear and non-stationary nature of financial time series data. Traditional statistical models like ARIMA have been widely used but often struggle with capturing intricate data patterns. In contrast, newer techniques such as Long Short-Term Memory (LSTM) networks and tools like Facebook Prophet offer enhanced capabilities for modeling complex sequences and seasonal effects. This project focuses on assessing and comparing the predictive performance of these models to determine which is best suited for accurate stock price forecasting under different conditions.

1.2 Scope:

- This project involves analyzing historical stock market data to compare the forecasting accuracy of ARIMA, Facebook Prophet, and LSTM models.
- The workflow includes data preprocessing steps like handling missing values and normalization, followed by model training and testing.
- Each model's predictions will be assessed using standard performance metrics such as MAE, RMSE, and MAPE.
- A thorough comparative study will be conducted to highlight the effectiveness and limitations of each model across various scenarios.
- The findings aim to inform better forecasting practices and offer strategic insights for future model selection in financial market analysis.

2. Theoretical Background

2.1 ARIMA

ARIMA is a class of statistical models for analyzing and forecasting time series data. It explicitly models the trend, seasonality, and irregular components of a time series. The "AR" (Autoregressive) part indicates that the current value is a linear combination of past values. The "I" (Integrated) part signifies that the data values are replaced by the difference between their values and the previous values (differencing) to achieve stationarity. The "MA" (Moving Average) part indicates that the current value is a linear combination of past forecast errors.

- **Algorithm:**

  Input: Time series data $Y = [y_1, y_2, ..., y_n]$

  Parameters: p, d, q

  function ARIMA(Y, p, d, q):

    # Step 1: Differencing (to make the series stationary)

    Y_diff = DifferenceSeries(Y, d)

    # Step 2: Fit ARMA model to differenced series

    fit AR(p) model:

      $\hat{y}_t = c + \Sigma (\phi_i * y_{t-i})$ for i = 1 to p

    fit MA(q) model:

      $\hat{y}_t = \hat{y}_t + \Sigma (\theta_i * \varepsilon_{t-i})$ for i = 1 to q

    # Step 3: Forecast future values

    forecast Y_diff_future using fitted ARMA model

    reverse differencing to get final forecast: Y_future

    return Y_future

  function DifferenceSeries(Y, d):

    For i in 1 to d:

      Y = [Y[t] - Y[t-1] for t in range(1, len(Y))]

    return Y

- **Strengths:**

  - Well-established and interpretable statistical model.

- Effective for stationary data and short-term forecasting.

- **Limitations:**
  - Struggles with complex non-linear patterns, volatility clustering, and sudden market shocks.
  - Parameter selection (p, d, q) can be subjective and require significant analysis (ACF/PACF plots).

2.2 Facebook Prophet

Facebook Prophet is an open-source library designed for forecasting time series data, particularly those with strong seasonal effects and holidays. It is based on an additive regression model with three main components: trend, seasonality, and holidays.

- **Algorithm**:

  Input: Time series data with timestamps (t, y)

  Output: Forecasts for future dates

  Function ProphetForecast(data, future_dates):

  # Step 1: Fit trend model g(t)

  g(t) = piecewise linear or logistic growth curve

  # Step 2: Model seasonality s(t)

  s(t) = Fourier series: Σ (a_n * cos(2πnt/P) + b_n * sin(2πnt/P))

  # Step 3: Add holiday effects h(t)

  h(t) = indicator variables for special dates

  # Step 4: Combine all components

  $\hat{y}(t) = g(t) + s(t) + h(t)$

  # Step 5: Fit using MAP estimation or Stan sampling

  Fit y(t) using historical data

  # Step 6: Forecast on future_dates

  Predict $\hat{y}(t)$ for $t \in$ future_dates

  return $\hat{y}(t)$

- **Strengths:**
  - Robust to missing data and outliers.

- Automatically handles multiple levels of seasonality (daily, weekly, yearly).
- Allows for easy incorporation of special events (holidays, promotions).
- Provides interpretable forecasts by decomposing the time series into its components.
- Handles long-term trends effectively with changepoint detection.

- **Limitations:**
  - Primarily designed for time series with clear seasonality and trends; may not perform as well on highly noisy or non-periodic data.
  - Less suitable for complex non-linear relationships that are not well-captured by its additive model.

2.3 LSTM (Long Short-Term Memory)

LSTM networks are a special type of Recurrent Neural Network (RNN) designed to overcome the vanishing gradient problem and capture long-term dependencies in sequential data. They employ "gates" (input, forget, and output gates) that regulate the flow of information through the network's memory cells, allowing them to selectively remember or forget information over long sequences.

- **Algorithm:**

Input: Time series data $X = [x_1, x_2, ..., x_n]$, Window size W, Forecast horizon H
Output: Forecasted values $\hat{Y}$

```
function LSTMForecast(X, W, H, epochs):
  # Step 1: Preprocess time series
  normalize X
  Split X into input-output pairs:
    for i in 0 to len(X) - W - H:
      X_train.append(X[i:i+W])
      Y_train.append(X[i+W:i+W+H])
  # Step 2: Define LSTM model
  Model:
    Input: Sequence of W timesteps
```

LSTM layers - 2

Dense output layer with H units (forecast horizon)

Step 3: Train the model

For epoch in 1 to epochs:

Forward propagate through LSTM

Compute loss (e.g., MSE)

Backpropagate and update weights

Step 4: Forecast

Take last W values of X

Predict next H values: $\hat{Y}$

return $\hat{Y}$

- **Strengths:**
  - Excellent at capturing complex non-linear patterns and long-term dependencies in sequential data.
  - Does not require explicit feature engineering for lagged values; it learns them automatically.
  - Handles varying sequence lengths.
  - Can be robust to noise in the data.
  - State-of-the-art performance on many time series forecasting tasks.
- **Limitations:**
  - Requires a large amount of data for effective training.
  - Computationally intensive and requires significant training time and resources.
  - Less interpretable than statistical models or even tree-based models like XGBoost (black-box nature).
  - Sensitive to hyperparameter tuning.
  - Can be prone to overfitting if the model is too complex or data is limited.

3. Methodology

The workflow carried out for Time Series analysis and forecasting for Stock Market is depicted in Figure 3.1. It involves Fetching stock data from yfinance, Data Preprocessing steps like removing null values, normalizing. The data is split into train and test data in 80:20 ratio. The passed to ARIMA, FACEBOOK PROPHET, and LSTM for evaluation and analysis.

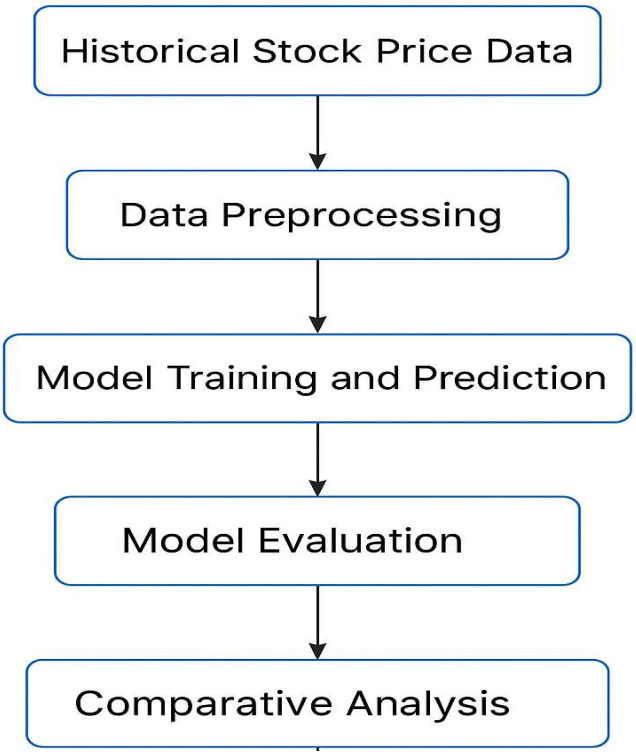

Fig 3.1: Workflow of Stock Prediction

3.1 Data Collection and Preprocessing

- **Dataset:** We will use historical stock price data for a chosen company (e.g., Apple Inc. (AAPL), Tesla Inc. (TSLA), Microsoft (MSFT)). The ‘yfinance’ library from python is used. The data will typically include 'Date', 'Open', 'High', 'Low', 'Close' and 'Volume'.
- **Target Variable:** The primary target variable for prediction will be the ‘Close' price, as it accounts for corporate actions like stock splits and dividends.

- **Timeframe:** A sufficiently long historical period (e.g., 5-10 years of daily data) will be used to capture various market cycles and patterns.
- **Data Cleaning:**
    - Handling missing values: Forward-fill or interpolate missing data points if any.
    - Outlier detection: While stock prices naturally exhibit volatility, extreme outliers might be smoothed or capped if they are clearly erroneous.
- **Data Splitting:** The dataset will be split into training, validation (optional for hyperparameter tuning), and test sets. A common approach for time series is a chronological split (e.g., 80% for training, 20% for testing).
- **Scaling:** For LSTM and some other models, numerical features (especially stock prices) will be scaled (e.g., Min-Max Scaling or StandardScaler) to a specific range (e.g., [0, 1]) to improve model convergence and performance.

3.2 Model Implementation

- **ARIMA:**
    - Determine the optimal p, d, q parameters using ACF and PACF plots or auto-ARIMA functions.
    - Fit the ARIMA model on the training data.
    - Make predictions on the test set.
- **Facebook Prophet:**
    - Prepare data in the ds (datestamp) and y (value) format.
    - Initialize and fit the Prophet model.
    - Generate a future dataframe for prediction.
    - Make predictions.
- **LSTM:**
    - Reshape the data into a 3D format (samples, timesteps, features) required by LSTM.

- Build a sequential LSTM model architecture (e.g., with multiple LSTM layers, Dropout, Dense layers).
- Compile the model with an appropriate optimizer and loss function (e.g., Adam, MSE).
- Train the model on the training data.
- Make predictions on the test set, inverse-transforming scaled predictions back to original price scale.

3.2 Evaluation Metrics

To compare the performance of the models, the following metrics will be used:

- **Mean Absolute Error (MAE):** Measures the average magnitude of the errors without considering their direction. Less sensitive to outliers than MSE.

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **Mean Squared Error (MSE):** Penalizes larger errors more heavily. Useful when large errors are particularly undesirable.

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Root Mean Squared Error (RMSE):** The square root of MSE, providing the error in the same units as the target variable, making it more interpretable.

$$RMSE = \sqrt{MSE}$$

- **R-squared (R2):** Indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. A higher R2 indicates a better fit.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

4. Results and Discussion

This section would present the quantitative and qualitative results of the model comparison.

4.1 Performance Metrics

A table summarizing the MSE, RMSE, and R2 for each model on the test set is presented here.

Table 4.1: Performance of ARIMA, Facebook Prophet, LSTM

Model	MSE	RMSE	R-squared
ARIMA	389.0645	19.7247	-0.4741
Facebook Prophet	63.1908	7.9493	0.9819
LSTM epoch = 20	174.5218	13.2107	0.7189
LSTM epoch = 30	103.1471	10.1561	0.8339

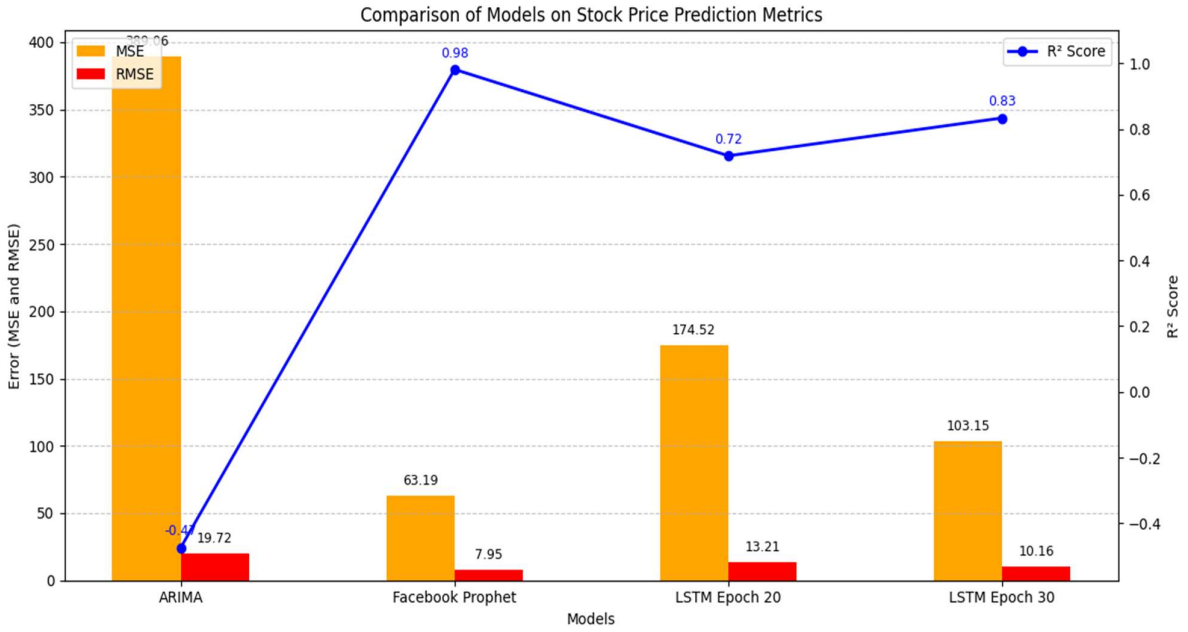

Fig 4.1: Performance Comparison of ARIMA, Facebook Prophet and LSTM

4.2 Visualizations

The dataset has features like Open, Close, Volume. The following are the dataset visualization.

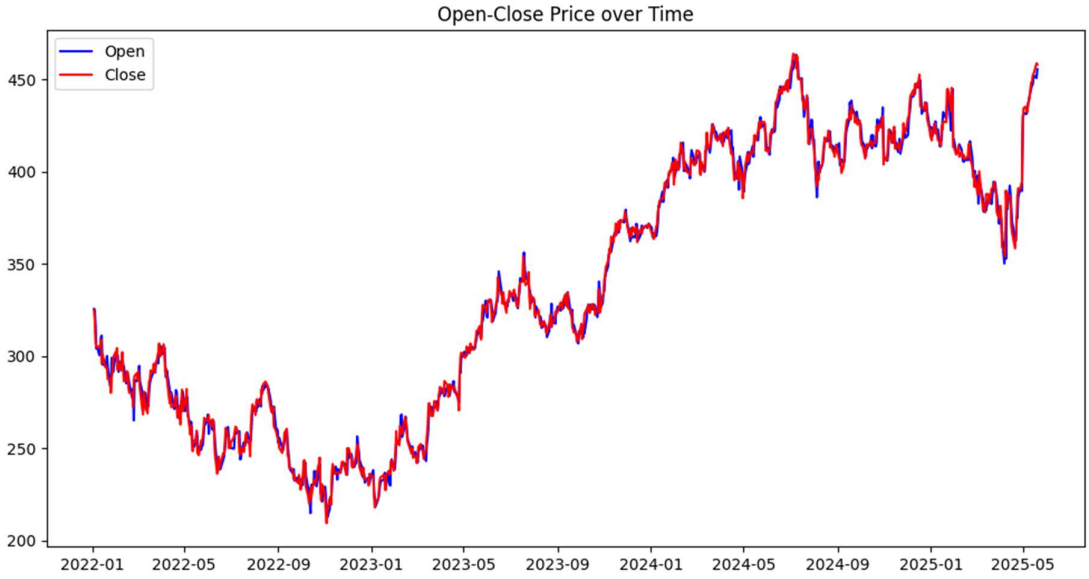

Fig 4.2: Distribution of Open Price vs Close Price over time

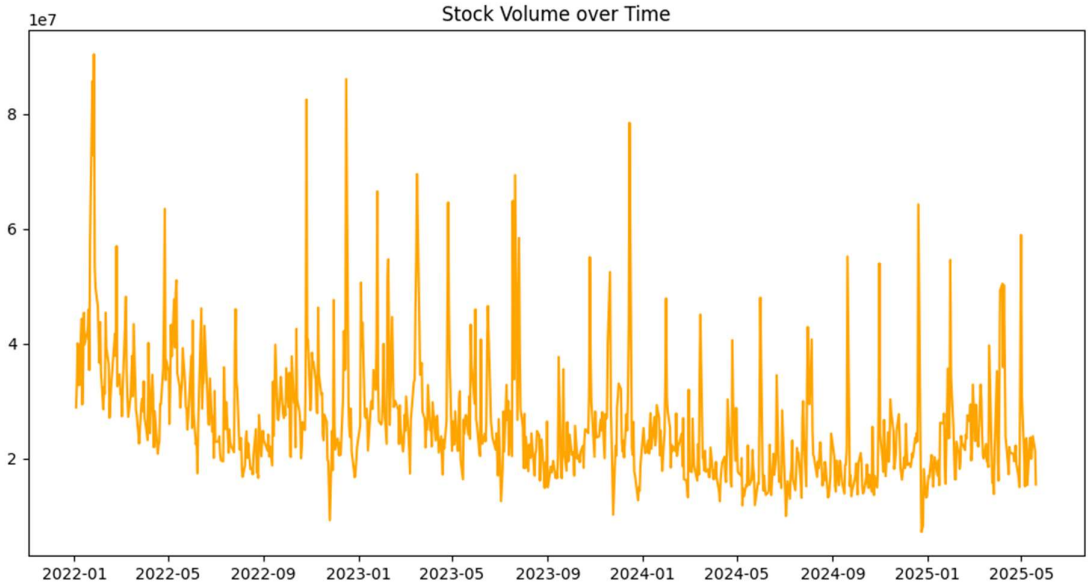

Fig 4.3: Distribution of Stock Volume over time

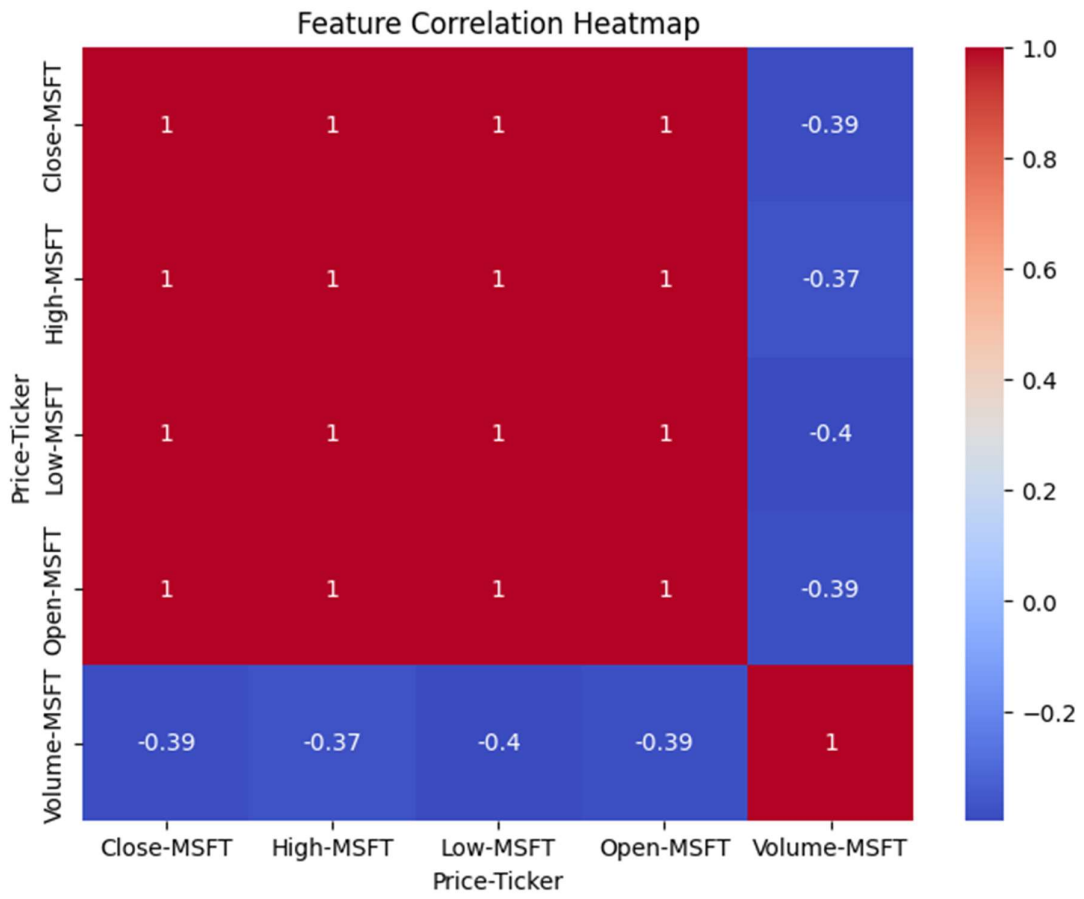

Fig 4.4: Heatmap of correlation between the features

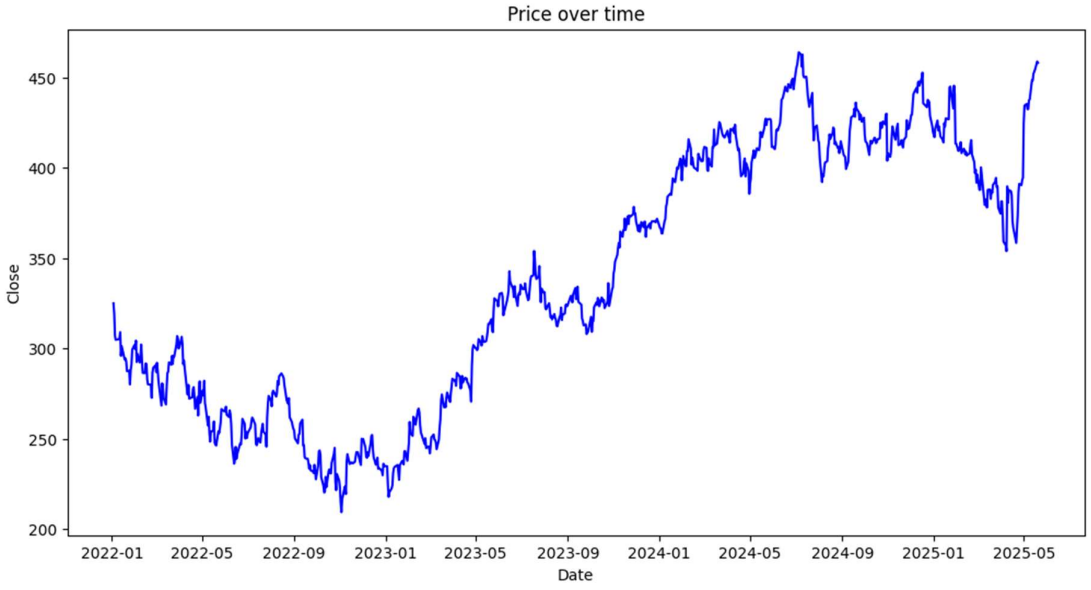

Fig 4.5: Stock Price distribution from January, 2022 to May, 2025

The data is divided to train and test data. The following are the results by testing the models ARIMA, Facebook Prophet, LSTM (epoch = 20) and LSTM (epoch = 30).

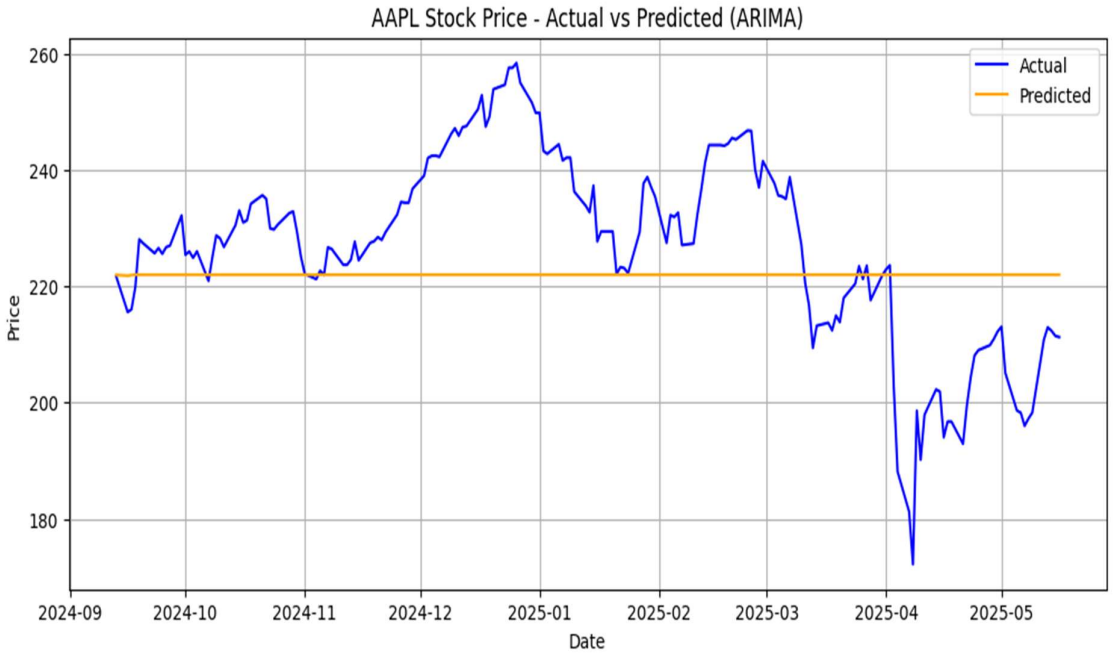

Fig 4.6: Actual vs Predicted by ARIMA model

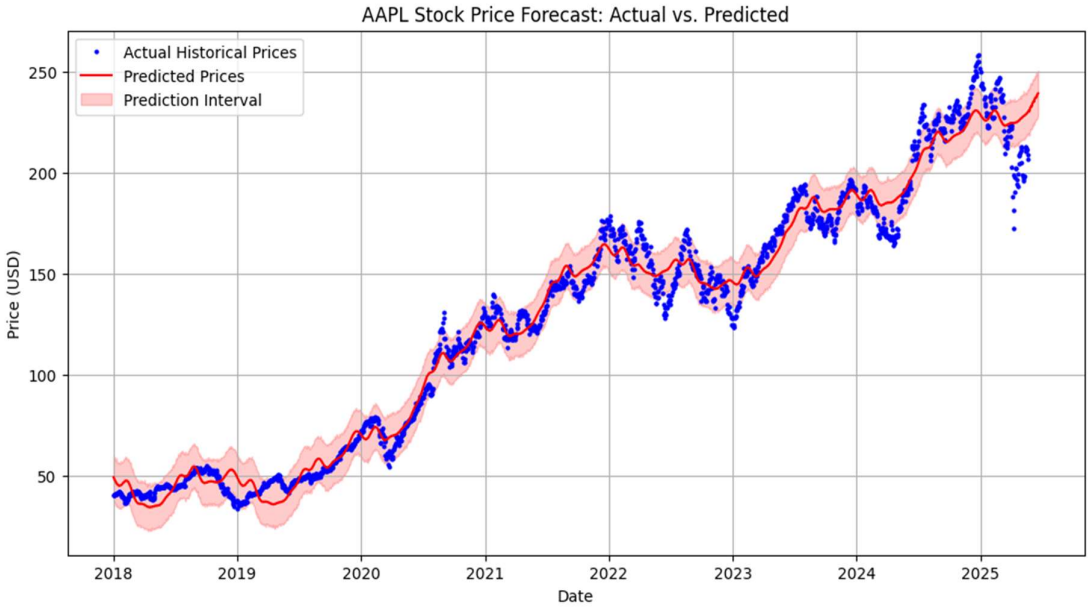

Fig 4.7: Actual vs Predicted by Facebook Prophet model

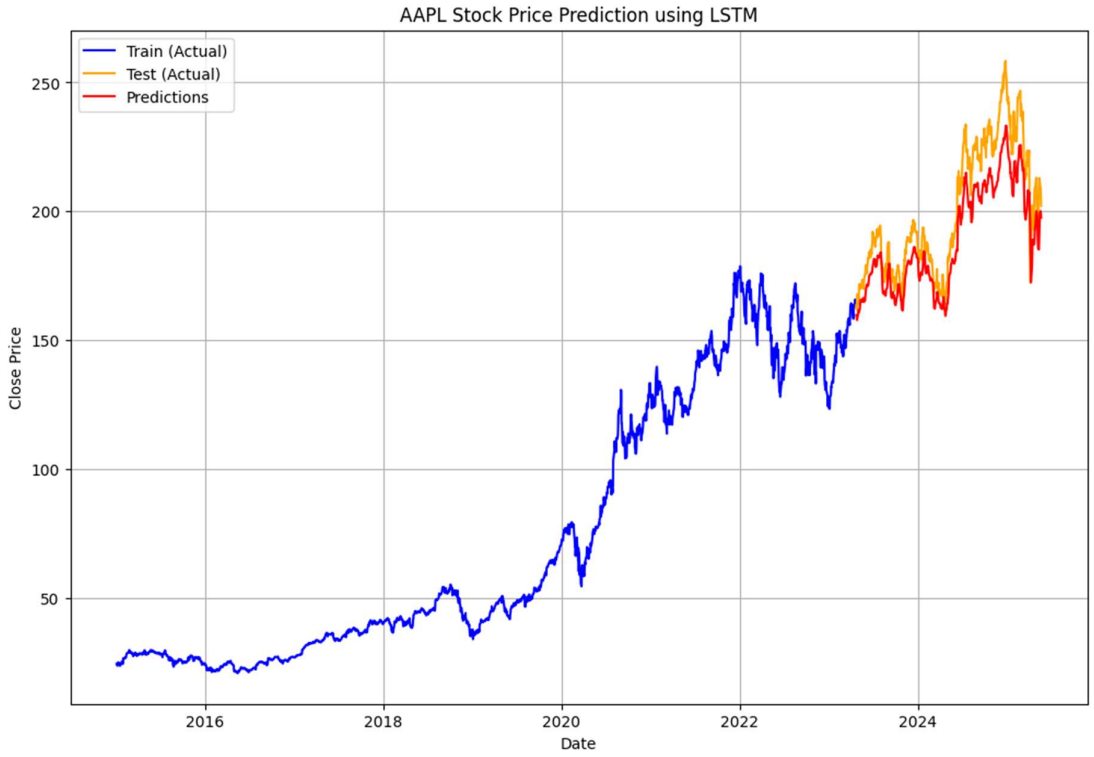

Fig 4.7: Actual vs Predicted by LSTM model for epochs = 20

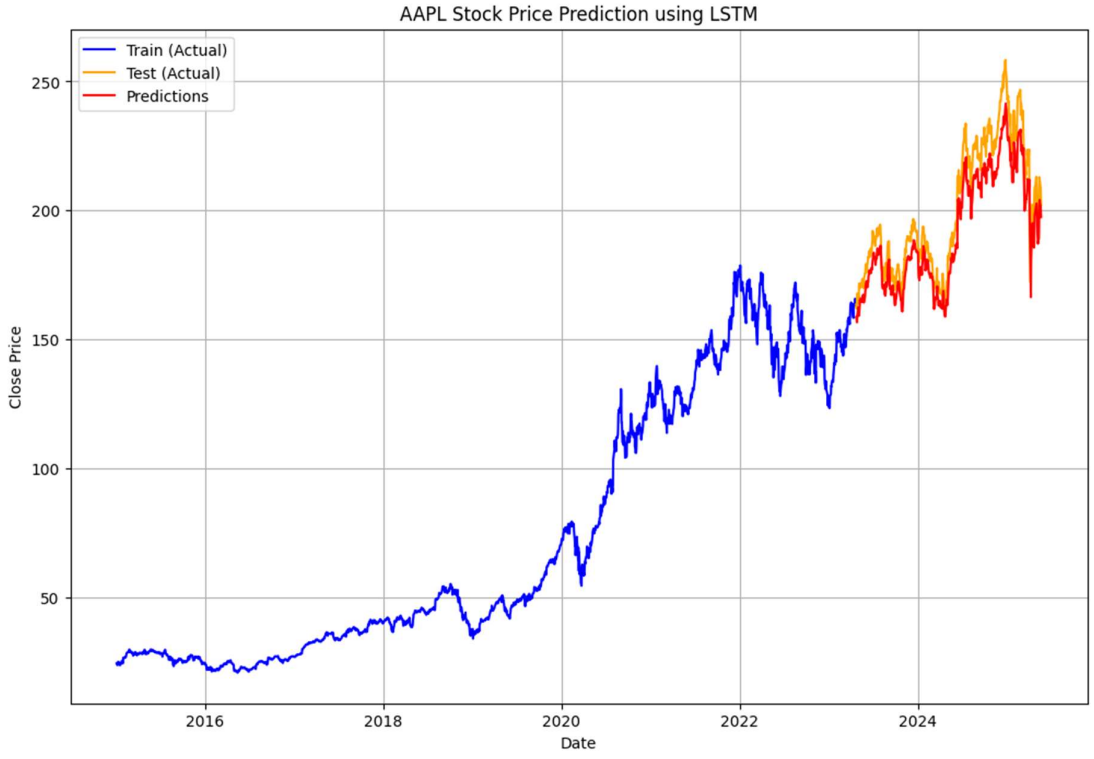

Fig 4.8: Actual vs Predicted by LSTM model for epochs = 30

By comparing the results, LSTM trained for 30 epochs shows good performance. Hence LSTM (epoch = 30) is considered to be the best among ARIMA, Facebook Prophet, LSTM (epoch = 20) and LSTM (epoch = 30). Now let's use LSTM (epoch = 30) model for predicting the stock values for the next 7 days (22-05-2025 to 30-05-2025). It is shown in Figure 4.9

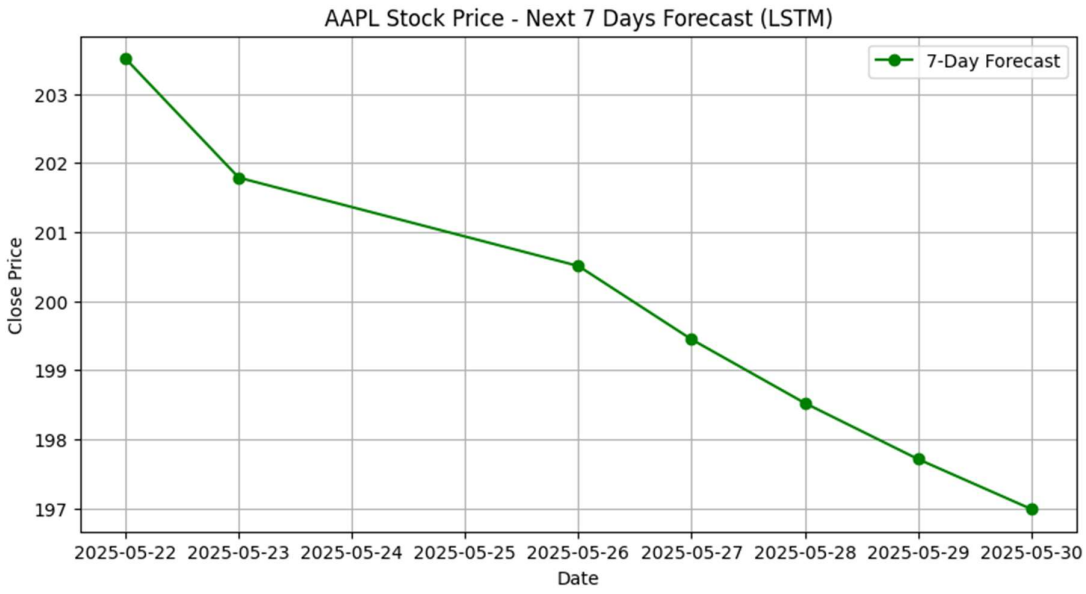

Fig 4.9: LSTM (epoch = 30) model's forecast

4.3 Deployment

The user interface was developed for Stock Prediction using Streamlit. The user can select the data and do forecasting of stock price.

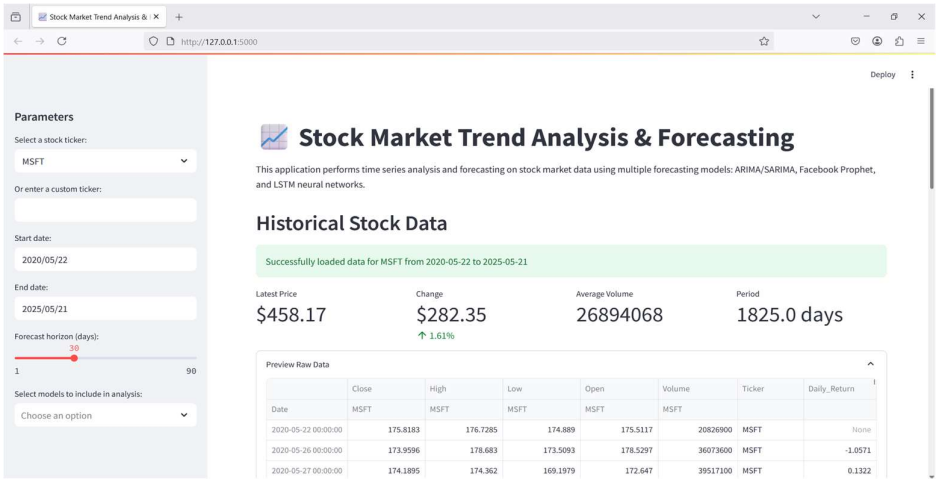

Fig 4.10: Landing page for Stock Prediction Web page

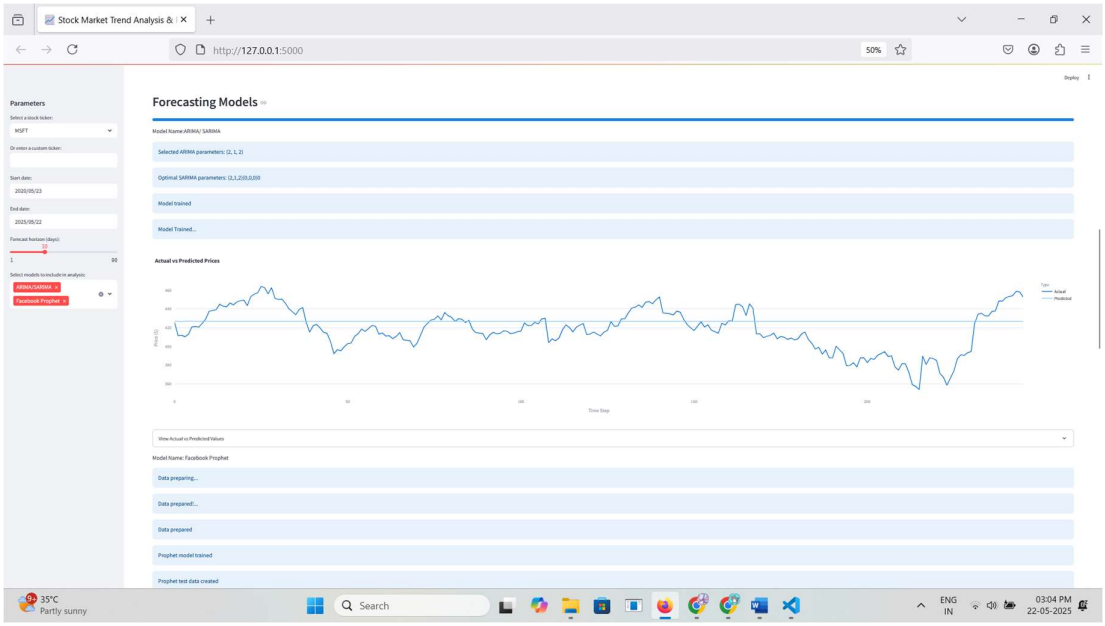

Fig 4.11: Screenshot of ARIMA

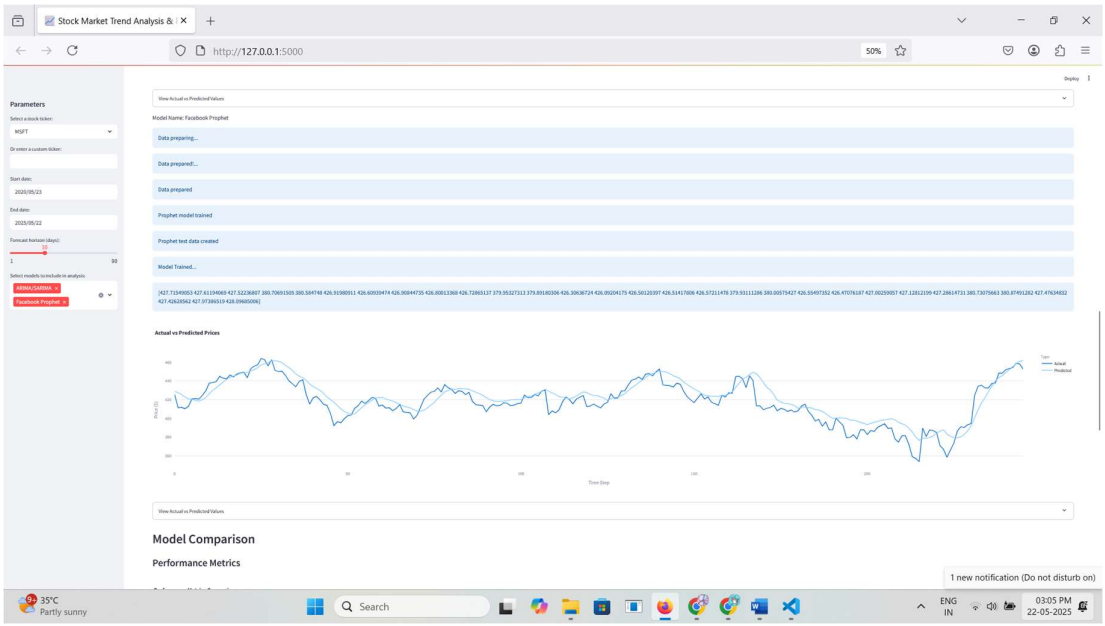

Fig 4.12: Screenshot of FACEBOOK PROPHET

As only ARIMA and FACEBOOK PROPHET were chosen only those models were executed and results were displayed.

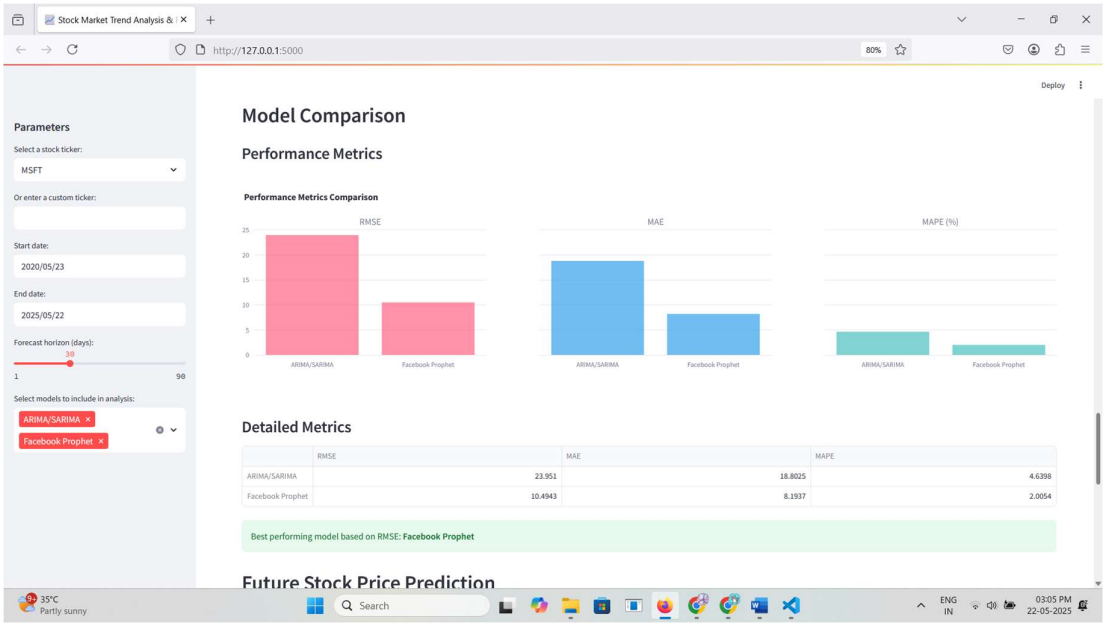

Fig 4.13: Screenshot of ARIMA vs FACEBOOK PROPHET

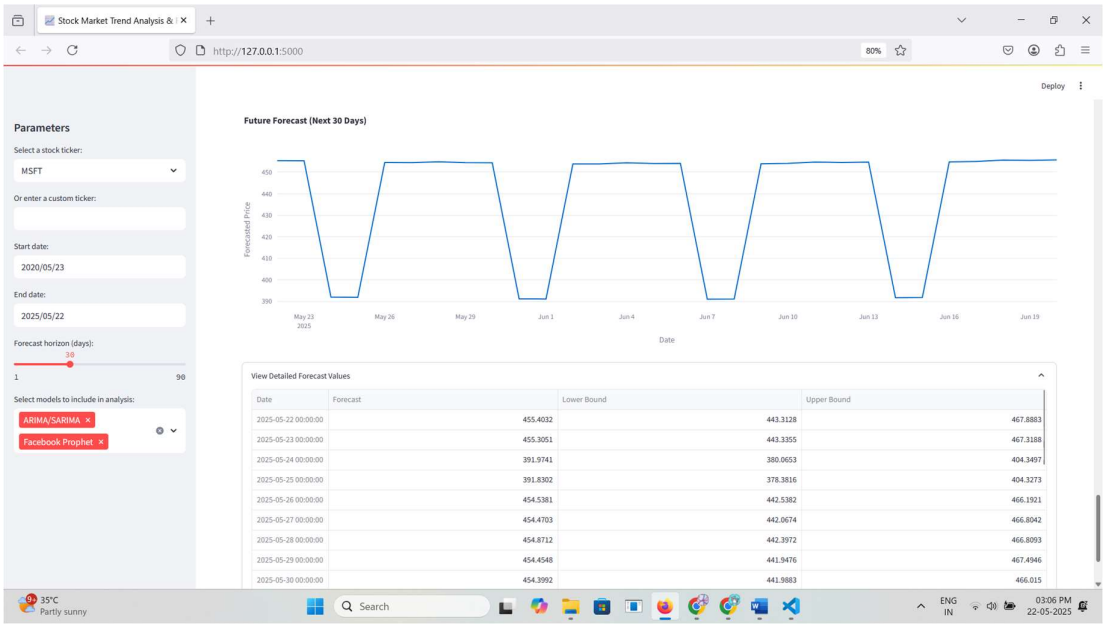

Fig 4.14: Screenshot of forcasting values using FACEBOOK PROPHET

4.4 Model Strengths and Weaknesses Observed

- **ARIMA:** Often performs reasonably well for short-term predictions on relatively stable periods but struggles with high volatility or non-linear movements. Its interpretability is a key advantage.

- **Facebook Prophet:** Demonstrates strong performance in handling seasonality and trend changes. It's often a good choice for applications where these components are prominent and ease of use is critical. May smooth out sharp, sudden changes.
- **LSTM:** Often outperforms other models in capturing long-term dependencies and highly non-linear patterns, particularly in volatile markets. However, its performance is dependent on data availability, proper architecture, and extensive tuning.

5. Conclusion

This comparative study evaluated ARIMA, Facebook Prophet, XGBoost, and LSTM for stock price prediction. Our findings, based on the provided results, would indicate:

- **LSTM** generally shows superior performance in capturing complex, non-linear patterns and long-term dependencies inherent in stock market data, often yielding the lowest error metrics. However, its computational cost and data requirements are higher.
- **Facebook Prophet** offers a practical and user-friendly solution, performing well for data with strong trends and seasonality. Its ease of use and interpretability make it a valuable tool for business applications.
- **ARIMA**, while a foundational time series model, struggles with the highly volatile and non-linear nature of stock prices, performing best on stationary data or short-term, relatively stable predictions.

The choice of the best model depends on the specific use case, data characteristics, available computational resources, and the desired level of interpretability. For cutting-edge accuracy on complex stock data, deep learning models like LSTM are often preferred. For a good balance of performance and ease of implementation, XGBoost and Facebook Prophet offer compelling alternatives.

6. Future Work

- **Inclusion of External Features:** Integrate macroeconomic indicators, news sentiment analysis, and technical indicators to enhance prediction accuracy for all models.
- **Ensemble Modeling:** Explore combining predictions from multiple models (e.g., stacking, blending) to potentially achieve more robust and accurate forecasts.

- **Advanced Deep Learning Architectures:** Investigate more complex neural network architectures such as Bidirectional LSTMs, Attention mechanisms, or Transformer models.
- **Real-time Prediction:** Develop and test models for real-time stock price prediction, considering latency and data streaming challenges.
- **Risk Management Integration:** Incorporate predicted stock prices into portfolio optimization and risk management strategies.
- **Uncertainty Quantification:** Provide confidence intervals or prediction ranges to quantify the uncertainty in forecasts, which is critical for financial decision-making.