# Stock Price Forecasting Using Time Series Analysis and Machine Learning

Abhi Thaker
Ayushi Arora
Bishwajit Dutta
Mohit Rathod
Ruchi Tiwari

***This study outlines a comprehensive, model-driven framework for time series forecasting using stock price data from Toronto Dominion (TD) Bank. The methodology spans exploratory data analysis (EDA), time series decomposition, and a comparative evaluation of forecasting models: ARIMA, Exponential Smoothing (ETS), Prophet, and Long Short-Term Memory (LSTM). An interactive Streamlit web application is developed to enable user-friendly forecasting and visualization. The project emphasizes rigorous statistical evaluation and business insights applicable to finance and related domains.***

## I. Introduction

Forecasting time series data plays a vital role in financial decision-making, inventory control, healthcare planning, and more. This project uses historical stock closing prices of TD Bank, sourced from Yahoo Finance[1], to analyze trends, seasonality, and future value prediction. Our goals are threefold: (1) conduct comprehensive EDA, (2) decompose the time series using both additive and multiplicative models, and (3) compare multiple forecasting models via a web app built using Streamlit. These steps not only help enhance forecasting accuracy but also create a modular system adaptable for other domains.

## II. Dataset and Exploratory Data Analysis

### A. Dataset Description

The dataset comprises daily opening-closing prices from 2020 to 2025, which we aggregate into monthly averages to stabilize variance and reduce short-term volatility. This transformation improves interpretability for long-term trend analysis.

- Source: Yahoo Finance via CSV download
- Attributes: Date (timestamp), Close (closing stock price), Open (opening stock price), High, Low.
- Frequency: Daily, aggregated to monthly for modeling stability

### B. Visualizations

The first step in the exploratory data analysis (EDA) involved visualizing the monthly closing prices of the TD

Bank stock. A time series plot was generated using the matplotlib library. The graph revealed a clear upward trend over the five-year period, with occasional drops corresponding to major economic disruptions, such as the COVID-19 pandemic. These visual insights helped establish the presence of long-term growth and possible cyclic behavior in stock prices.


Fig. 1. Time Series Plot

### C. Missing Values and Outlier Detection

Upon loading the dataset, we examined it for missing entries. Fortunately, the data was largely complete, with very few null values, which we handled using forward-filling techniques. We then checked for outliers using the Interquartile Range (IQR) method, which flagged extreme values that could represent anomalies such as market shocks or data collection errors. These outliers, visible as sudden spikes or drops in the graph, typically occurred during earnings seasons or global market corrections. Rather than eliminating these points, we retained them to maintain the integrity of the market's real-world behavior.

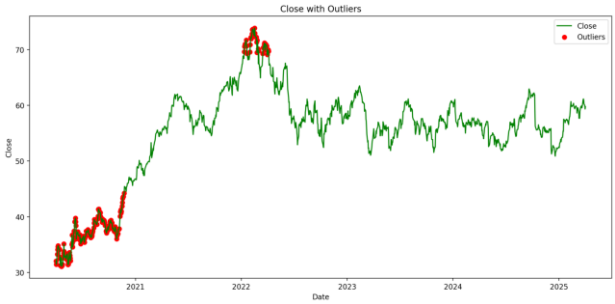


Fig. 2. Outlier Detection

### D. *Stationarity and Differencing*

To assess the stationarity of the series, we applied for the Augmented Dickey-Fuller (ADF) test. The test on the original series yielded an ADF statistic of -2.79 with a p-value of 0.0597, indicating non-stationarity. After applying first-order differencing, the ADF statistic dropped to -33.85 with a p-value of 0.0000, confirming that the transformed series was stationary and suitable for ARIMA modeling.

Before:

```
ADF Test for 'Close Price'
ADF Statistic: -2.7900583288277767
p-value: 0.0597079577182304
# Lags Used: 1
# Observations Used: 1255
❌ Result: The series is non-stationary (fail to reject null hypothesis)
```

Fig. 3. ADF before smoothing

After:

```
ADF Test for 'Close Price (1st Difference)'
ADF Statistic: -33.855965892939736
p-value: 0.0
# Lags Used: 0
# Observations Used: 1255
✅ Result: The series is stationary (reject null hypothesis)
```

Fig. 4. ADF after smoothing

### E. *Need for Smoothing*

Although smoothing was not central to this project, rolling averages were considered during exploratory analysis. However, due to the preference for decomposition and differencing in the modeling phase, smoothing was not applied further. Nonetheless, it remains a valuable tool for visual trend assessment.

### F. *External Factors*

Stock prices are often influenced by external factors including macroeconomic indicators, earnings announcements, and geopolitical developments. In this project, Canadian public holidays were incorporated into the Prophet model using a built-in holiday calendar. This addition accounted for periods of low market activity or predictable investor behavior, enhancing the accuracy of forecasts during cyclical lulls.

## III. Time Series Decomposition

Time series decomposition was conducted using the seasonal_decompose function from the statsmodels library. The purpose was to separate the series into three components: trend, seasonality, and residual. This allowed us to study the structural behavior of the stock price series independently.

### A. *Additive and Multiplicative Models*

A To better understand the underlying structure of the TD Bank stock price series, we applied both additive and multiplicative decomposition using the seasonal_decompose() function.

In the additive model (see Figure 5), the original time series is expressed as the sum of its three components:

$$Yt = Tt + St + Rt$$

This model assumes that seasonal variations and residuals remain constant in amplitude, regardless of the level of the trend. The trend component in the additive plot reveals a strong upward movement from 2020 through early 2022, followed by stabilization. The seasonal component displays consistent periodic patterns with regular peaks and troughs, suggesting predictable cycles — likely monthly, given the dataset frequency. The residual component appears symmetric and centered around zero, with greater volatility in 2022, coinciding with post-pandemic market fluctuations.

In contrast, the multiplicative decomposition (see Figure 6) uses the formula:

$$Yt = Tt \times St \times Rt$$

This approach assumes that seasonal effects grow or shrink in proportion to the trend. The seasonal component here shows a relatively flat, oscillating range around a baseline of 1.0, with minor fluctuations (~0.995 to 1.007). This supports the conclusion that the stock's seasonality is stable and not strongly proportional to trend changes. The residual component is expressed as a multiplicative deviation from the trend-seasonality product and stays within a tighter band than in the additive case, though it still reflects significant spikes in periods of high market volatility.

Based on the TD stock data, additive decomposition appears more interpretable and effective, as the amplitude of seasonality does not scale dramatically with the trend.

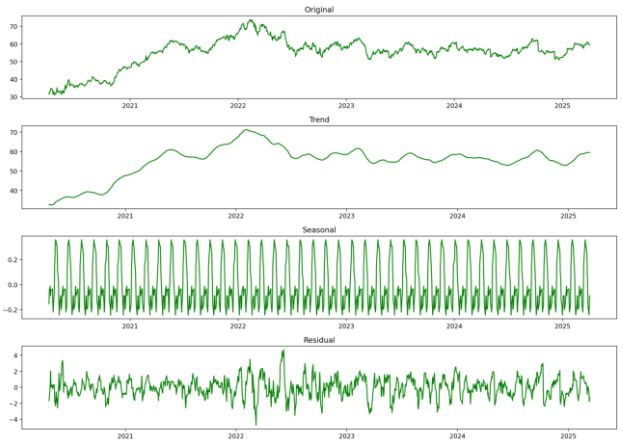


Fig. 5. Additive Model

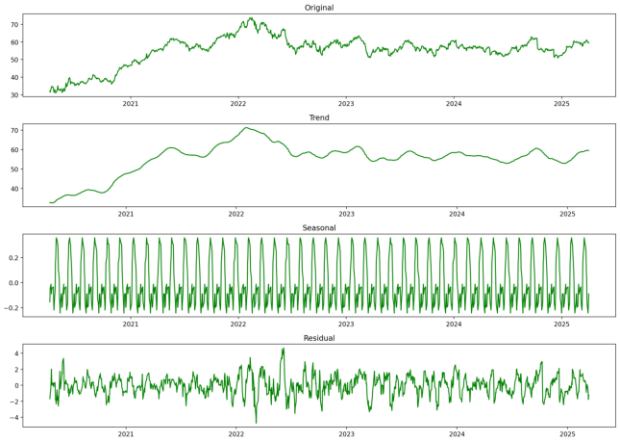

Fig. 6. Multiplicative Model

### B. *Autocorrelation Analysis and Simulated Series Behavior*

The ACF plot of the first-order differenced TD stock series (Figure 7) shows a single significant spike at lag 1, with all subsequent values lying within the confidence band. This confirms that the series has been effectively transformed to a stationary process with minimal autocorrelation beyond immediate lags. The absence of long-range dependence supports the suitability of ARIMA modeling, and guided our use of a (7,1,9) configuration based on AIC optimization.

To validate the statistical nature of the original series, we compared it with simulated white noise and random walk data (Figure 8). The TD stock series resembled a random walk—non-stationary with persistent trends. After differencing, the behavior aligned closely with white noise, suggesting the removal of trend and serial correlation. This further confirmed the stationarity of the transformed series and the appropriateness of statistical forecasting models such as ARIMA and ETS.

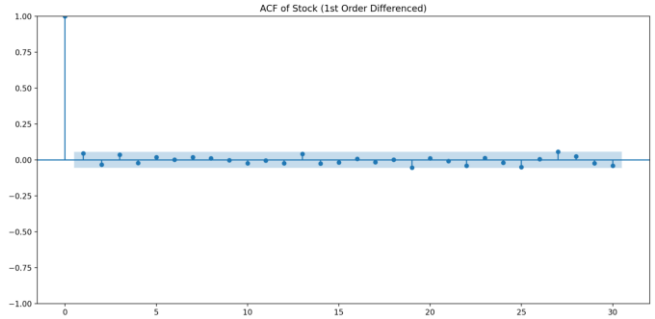

Fig. 7. ACF analysis

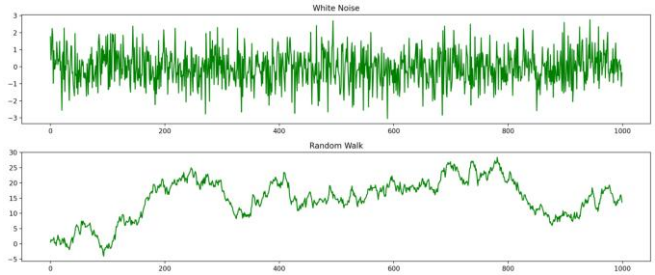

Fig. 8. White Noise vs. Random Walk

### C. *Interpretation of Components*

The trend component revealed sustained growth in TD Bank's stock prices, especially after 2020. The seasonality component indicated recurring patterns every 12 months, suggesting investor optimism during the early part of the year. The residual component, which reflects random noise or unexplained variance, was higher during periods of economic uncertainty, providing insight into periods of elevated risk.

### D. *Business Insights*

Understanding these components helps businesses and investors make informed decisions. Trend analysis supports strategic investment and expansion planning. Seasonal patterns help financial analysts time market entries or exits. Residuals assist in risk assessment by identifying periods of abnormal volatility.

## IV. Forecasting Model Comparison

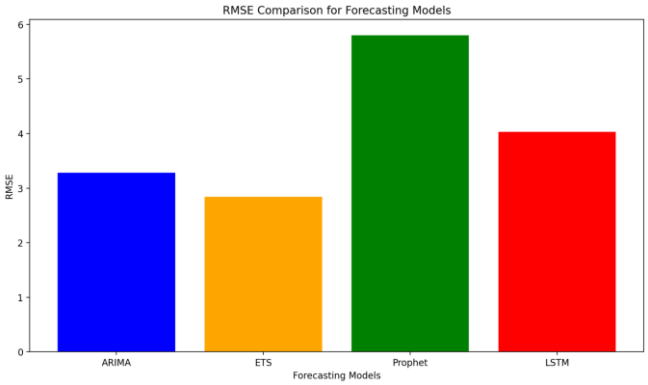

Fig. 9. RMSE comparison

### A. *Models Applied*

Four forecasting models were implemented:

- ARIMA (AutoRegressive Integrated Moving Average) – A linear model suited for stationary data.
- ETS (Exponential Smoothing) – Assigns greater weight to recent observations.
- Prophet – A robust model developed by Facebook that handles holidays and missing data.
- LSTM (Long Short-Term Memory) – A deep learning model capable of capturing nonlinear temporal patterns.

### B. *Model Configuration*

Each model was fine-tuned for accuracy. ARIMA used an order of (7,1,9) based on AIC optimization. ETS was implemented with a smoothing level of 0.2. Prophet incorporated Canadian holidays and was set to yearly seasonality. The LSTM model used a 12-month look-back window, one hidden layer with 50 units, and was trained for 100 epochs using a batch size of 1.

### C. *Training and Testing*

The dataset was split into training and testing sets using an 80:20 ratio. All models were trained on the earlier portion of the data and tested on the most recent months to evaluate their forecasting accuracy.

### D. Result Analysis

Among the four models, ETS outperformed others across all evaluation metrics, making it the most effective forecasting tool in this study. ARIMA followed closely and was especially robust after differencing. Prophet and LSTM delivered decent performance, particularly for long-term seasonal patterns, but slightly lagged in precision.

| | RMSE | MAE | MAPE | MSE |
|---|---|---|---|---|
| ARIMA | 3.28 | 2.84 | 4.98 | 10.78 |
| ETS | 2.84 | 2.38 | 4.22 | 8.08 |
| Prophet | 5.8 | 4.77 | 8.15 | 33.65 |
| LSTM | 4.03 | 3.17 | 5.78 | 16.22 |

Fig. 10. Forecasting Metrics Comparison

### E. Forecasting Horizon and Industry Relevance

The suitability of each forecasting model depends on both the forecast horizon and the nature of the industry. ETS and ARIMA are more appropriate for short-term forecasting, particularly in domains like finance, where capturing recent fluctuations, volatility, and rapid market shifts is critical. On the other hand, Prophet and LSTM are better suited for long-term forecasting in industries characterized by strong seasonal behavior. In healthcare, these models can be used to anticipate periodic trends such as flu outbreaks or hospital admissions. Similarly, in retail, Prophet excels in forecasting sales patterns driven by recurring events like holidays and promotional campaigns. Selecting the appropriate model based on domain and forecast length enables more accurate and actionable insights for decision-making.

## V. Streamlit Web Application

### A. Overview of Functionality

The Streamlit web application, hosted at https://my-app-app-c4btndxjogwlseblgfzu9d.streamlit.app, offers an interactive platform that brings our forecasting pipeline to life. It integrates each stage of the time series workflow into a seamless user experience, enabling both technical and non-technical users to explore, analyze, and forecast financial data with minimal effort.

### B. Feature Highlights

1. CSV Upload Interface
   Upon launching the app, users are greeted with a clean interface that includes a drag-and-drop area and file browser to upload a time series CSV file. The system reads the file and instantly visualizes the raw time series data for preliminary inspection.
2. Decomposition Panel
   Users can select either Additive or Multiplicative decomposition using a dropdown selector. Once selected, the app decomposes the uploaded time series into trend, seasonal, and residual components and displays each one in a separate subplot.
3. Stationarity Testing
   The app automatically performs an ADF (Augmented Dickey-Fuller) test on both the original and differenced series. The results are displayed clearly, showing the test statistic, p-value, and a summary of whether the series is stationary.
4. Forecasting Model Selection
   The app includes four selectable models:
   - ARIMA
   - Exponential Smoothing (ETS)
   - Prophet
   - LSTM

   Upon model selection, the app trains the chosen model on the uploaded data, forecasts future values, and displays the output on an interactive line plot.
5. Forecast Accuracy Evaluation
   The final feature includes a metric comparison panel where users can select RMSE, MAE, MAPE, or MSE from a dropdown. The app then displays a bar chart comparing all models side-by-side based on the selected metric.


Fig. 11. Upload interface

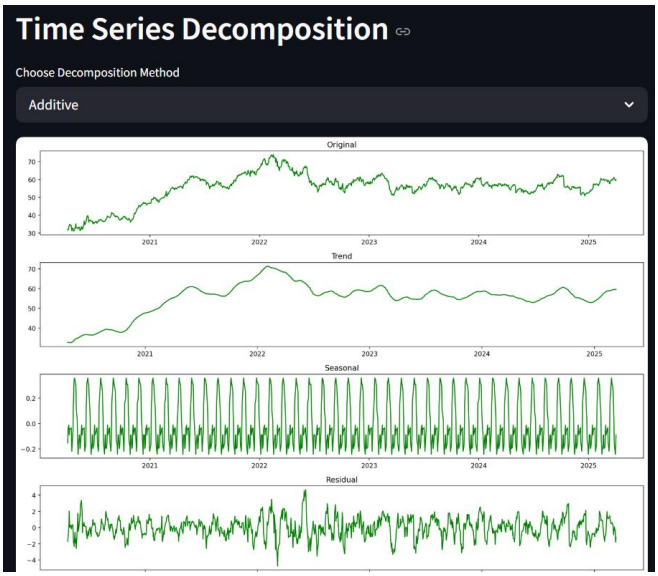


Fig. 12. Decomposition Panel

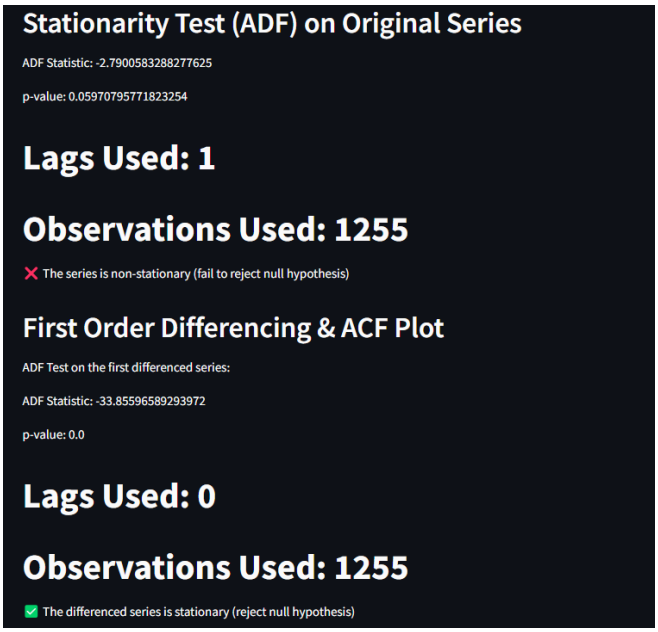


Fig. 13. Stationarity Testing

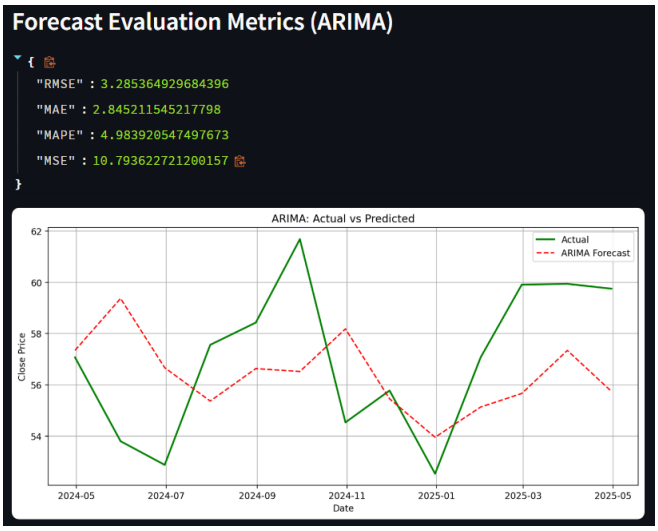


Fig. 14. Forecast Evaluation


Fig. 15. Metrics Evaluation

### C. *User Experience and Value*

The web app was built with ease of use in mind. It requires no programming skills and provides intuitive feedback at every stage of interaction. Users can upload any compatible time series, observe statistical properties, decompose it, and forecast with their chosen model in just a few clicks.

From an academic perspective, this app helps visualize complex time series techniques interactively. From a business perspective, it empowers analysts to make informed decisions based on model performance and trend insights.

### D. *Cloud Deployment and Accessibility*

The deployed version is accessible online and removes the need for local setup or dependencies. This enables professors, classmates, or stakeholders to test and validate the app directly through a web browser.

## REFERENCES

[1] *Yahoo Finance. https://finance.yahoo.com*