

# Time Series Forecasting With ARIMA Model On


**AMAZON STOCK  
PRICE**

# About Amazon

---

Amazon, founded by Jeff Bezos in 1994, began as an online bookstore and evolved into the world's largest e-commerce platform. It offers diverse services, including Amazon Web Services (AWS), Prime membership, and entertainment. Renowned for its customer-centric approach and innovation, Amazon significantly impacts global retail and technology. The company is headquartered in Seattle, Washington.

# AGENDA

- **Business Problem**
  - **Objectives**
  - **Data Overview**
  - **Analysis and Findings**
  - **Recommendations**
- 

# **Business Problem**

---

**As an individual investor interested in optimizing your personal investment decisions, you recognize the importance of leveraging data-driven approaches to achieve financial goals. Amazon, as a leading global technology and e-commerce company, presents lucrative opportunities for investment. However, navigating the volatility of stock markets requires careful analysis and strategic decision-making.**

**Your objective is to develop a time series forecasting model using ARIMA (AutoRegressive Integrated Moving Average) to predict future stock prices of Amazon. By accurately forecasting Amazon stock prices, you can make informed investment decisions, such as buying, selling, or holding Amazon stocks, at opportune moments to maximize returns and minimize risks.**

**Your model should utilize historical stock price data, along with any relevant external factors that may impact Amazon's stock performance, such as market trends, economic indicators, and company news. Additionally, you should evaluate the performance of your forecasting model using appropriate metrics, such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE), to assess its accuracy and reliability.**

---

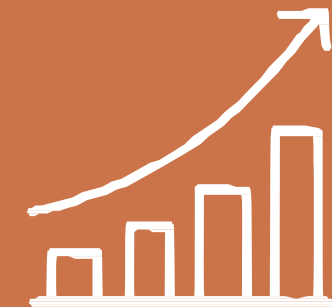
# Objectives



This project aims to develop an ARIMA-based model to predict Amazon's future stock prices, leveraging historical data and external factors for informed investment decisions.



The project aims to enhance investment strategies using an ARIMA model trained on historical Amazon stock data



The model's performance will be evaluated using MAE and RMSE to ensure prediction accuracy and reliability for informed investment decisions.

---

# Data Overview: Amazon Stock Data

## **Dataset Description:**

Source: Historical stock data for Amazon (AMZN).

Format: CSV file.

Period: Contains daily stock prices over a specified period.

## **Key Columns:**

Date: The date of the stock price observation.

Open: The opening price of Amazon stock on each day.

## **Processing Steps:**

Data Extraction:

Extracted the "Date" and "Open" columns into a new data frame.

Datetime Conversion:

Converted the "Date" column to datetime format to ensure accurate time-based operations.

Index Setting:

Set the "Date" column as the index of the Data Frame for easy time series manipulation.

Frequency Adjustment:

Adjusted the data frequency to business days using `freq("b")`, accounting for weekends and holidays.

Handling Missing Values:

Filled any missing values in the "Open" column using backward fill (`bfill`) to maintain data continuity.

# Analysis and Findings

- Data Frame named Amazon. To verify the data has been loaded correctly, the code then uses the head(10) method to display the first 10 rows of the Data Frame. This preview typically includes columns such as Date, Open, High, Low, Close, Adjusted Close, and Volume, offering an initial look at Amazon's stock prices and trading volumes for those dates.
- The "Date" is then set as the Data Frame's index. The frequency of the data is adjusted to business days using the as freq ("b") method, which accounts for weekends and holidays. Any missing values in the "Open" column are filled using backward fill to maintain data continuity. Finally, the first 12 rows of the processed Data Frame are displayed to verify that the transformations have been applied correctly. This process ensures that the data is properly formatted, consistent, and ready for further analysis.

```
[15]:
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1997-05-16	1.968750	1.979167	1.708333	1.729167	1.729167	14700000
1	1997-05-19	1.760417	1.770833	1.625000	1.708333	1.708333	6106800
2	1997-05-20	1.729167	1.750000	1.635417	1.635417	1.635417	5467200
3	1997-05-21	1.635417	1.645833	1.375000	1.427083	1.427083	18853200
4	1997-05-22	1.437500	1.447917	1.312500	1.395833	1.395833	11776800
5	1997-05-23	1.406250	1.520833	1.333333	1.500000	1.500000	15937200
6	1997-05-27	1.510417	1.645833	1.458333	1.583333	1.583333	8697600
7	1997-05-28	1.625000	1.635417	1.531250	1.531250	1.531250	4574400
8	1997-05-29	1.541667	1.541667	1.479167	1.505208	1.505208	3472800
9	1997-05-30	1.500000	1.510417	1.479167	1.500000	1.500000	2594400

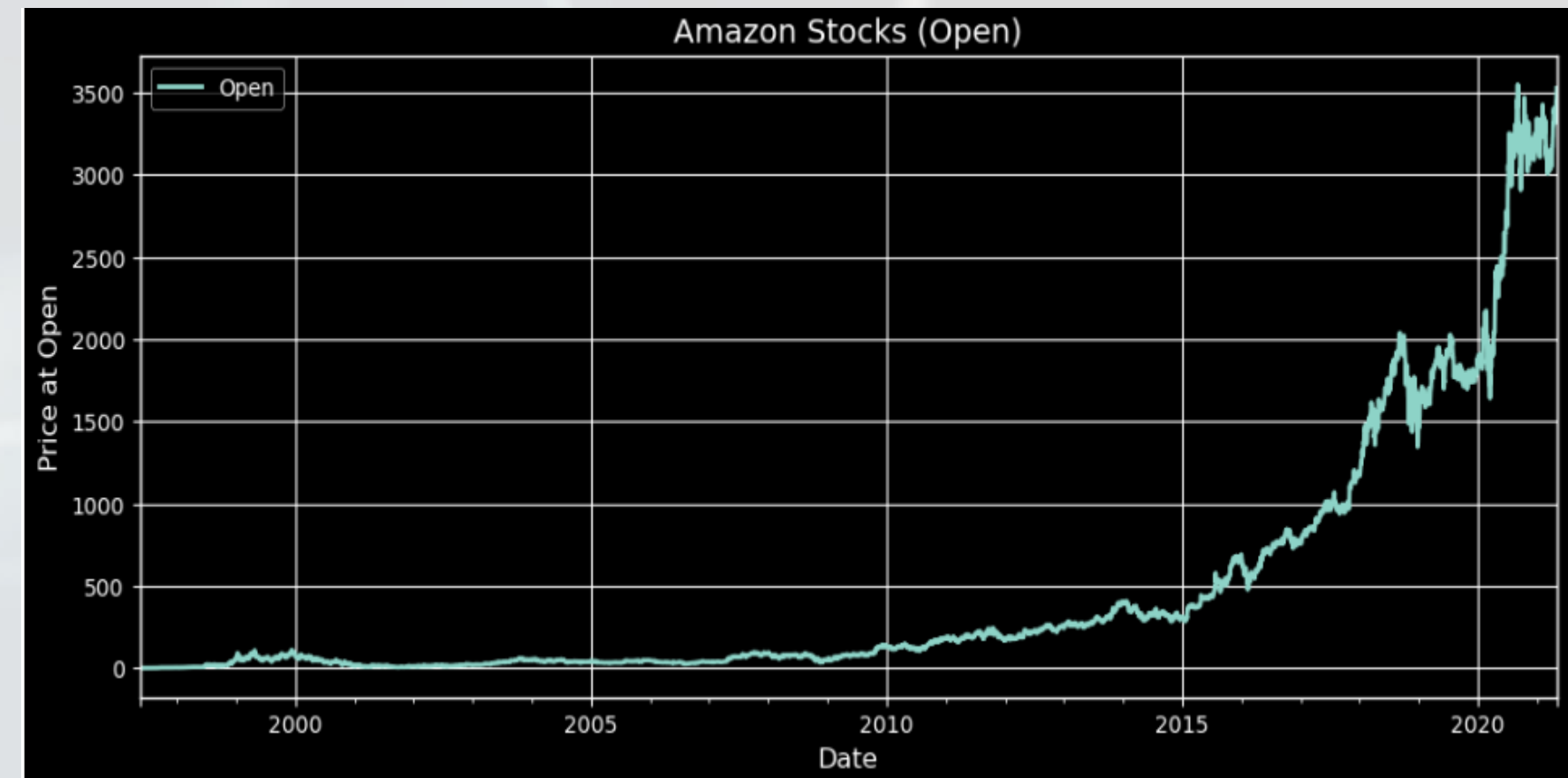
```
:
```

	Open
	1997-05-16
	1997-05-19
	1997-05-20
	1997-05-21
	1997-05-22
	1997-05-23
	1997-05-26
	1997-05-27
	1997-05-28
	1997-05-29
	1997-05-30
	1997-06-02



# "Visualizing Amazon's Market Journey: Open Stock Prices Unveiled"

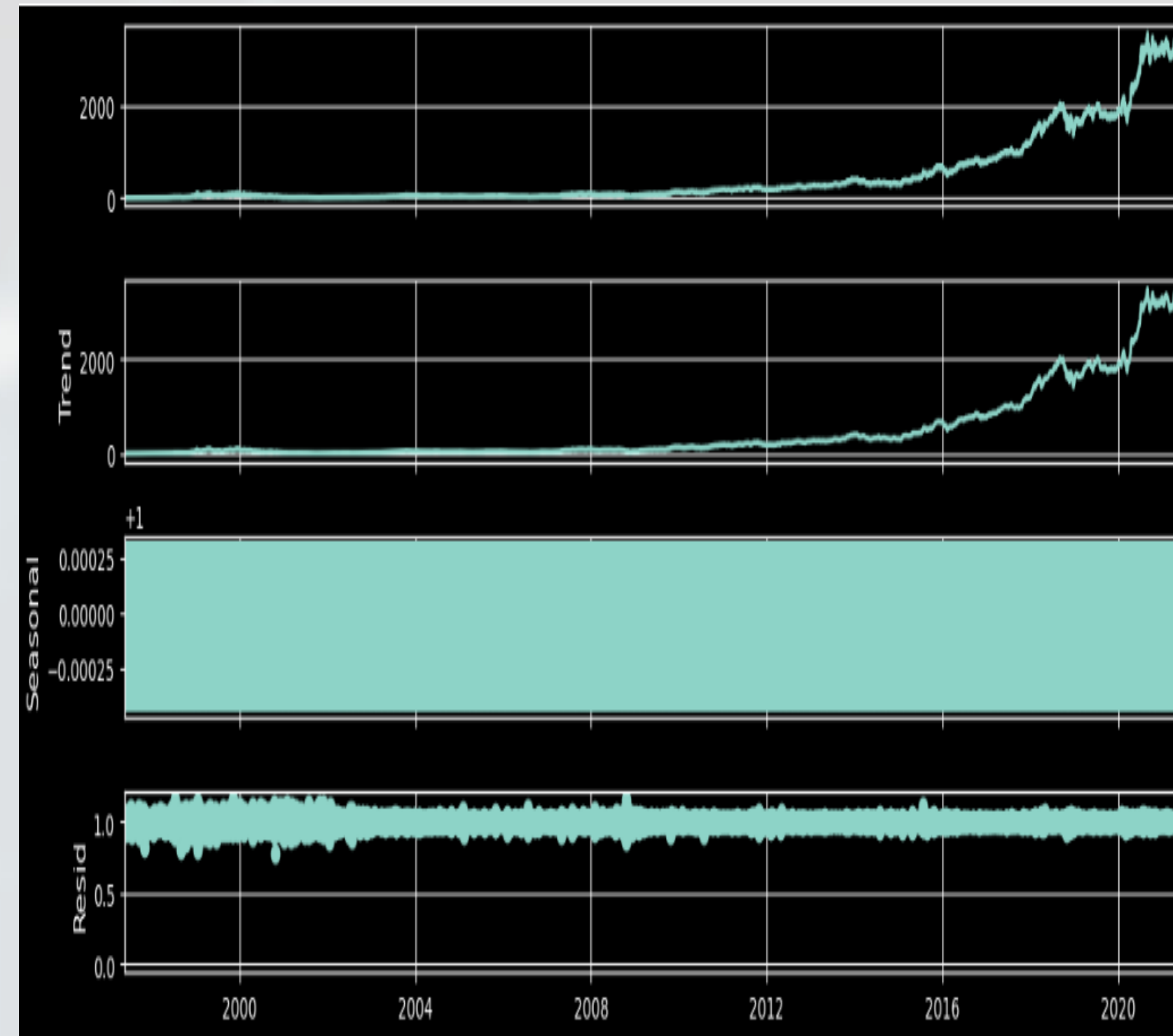
time series data of Amazon's opening stock prices. The `plot()` function from `matplotlib` is used to create the plot, and the title "Amazon Stocks (Open)" is assigned to provide context about the data being visualized. Additionally, the y-label is set to "Price at Open" using the `set()` method, which makes it clear what the y-axis represents. Finally, `plt. show()` is called to display the plot to the user. This visualization allows users to get an initial understanding of the patterns and fluctuations in Amazon's stock prices over time.





# "Unlocking Amazon's Time Series Dynamics: Multiplicative Seasonal Decomposition Illuminates Patterns and Trends"

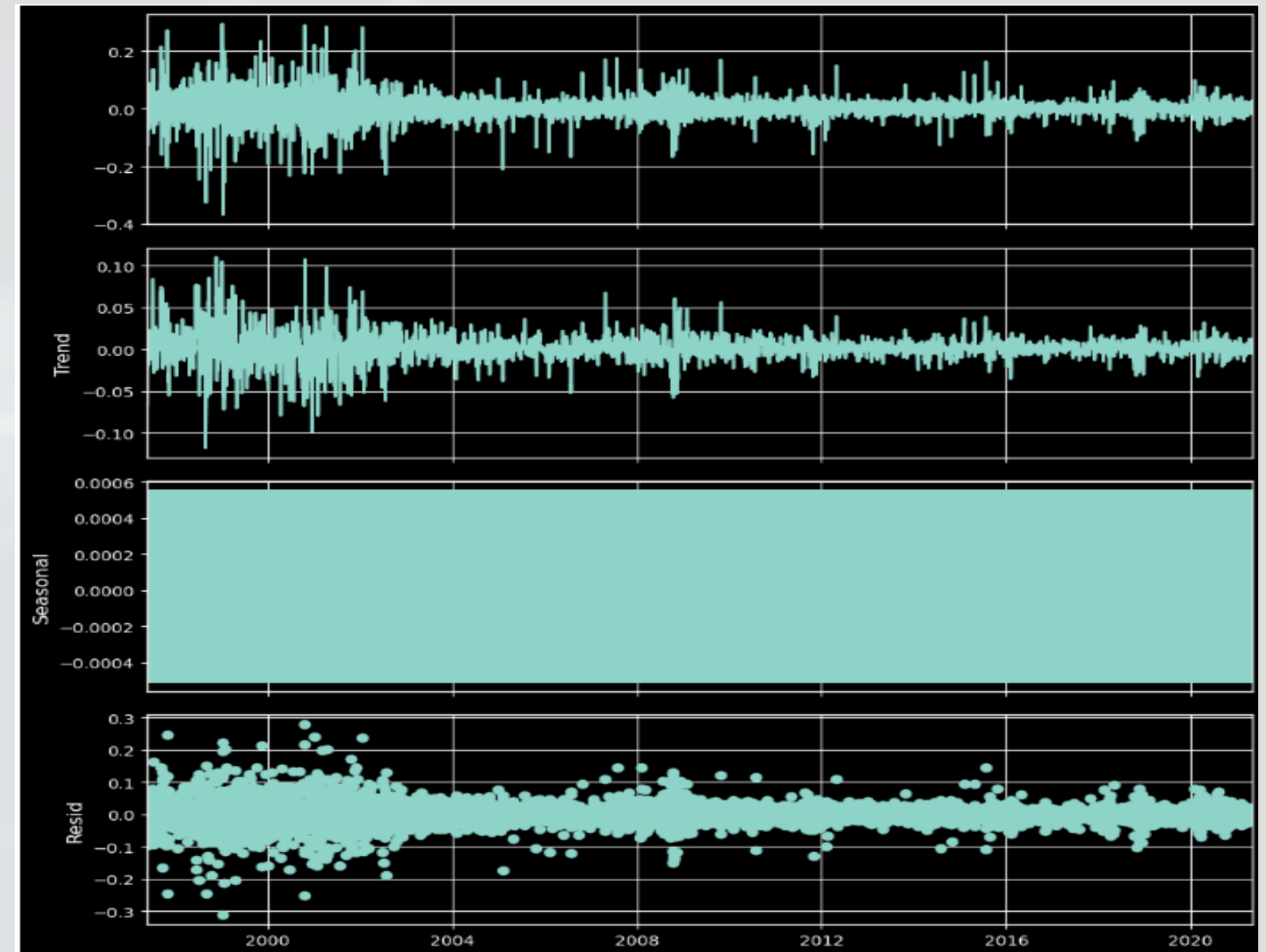
Indicates that the decomposition should be done using a multiplicative model, which assumes that the seasonal component varies proportionally with the level of the time series. The resulting decomposition is then visualized using the `plot()` method, which typically displays four subplots: the original time series, the trend component, the seasonal component, and the residual component. This visualization helps in understanding the underlying patterns and variations in Amazon stock prices, including any seasonality and trend present in the data.



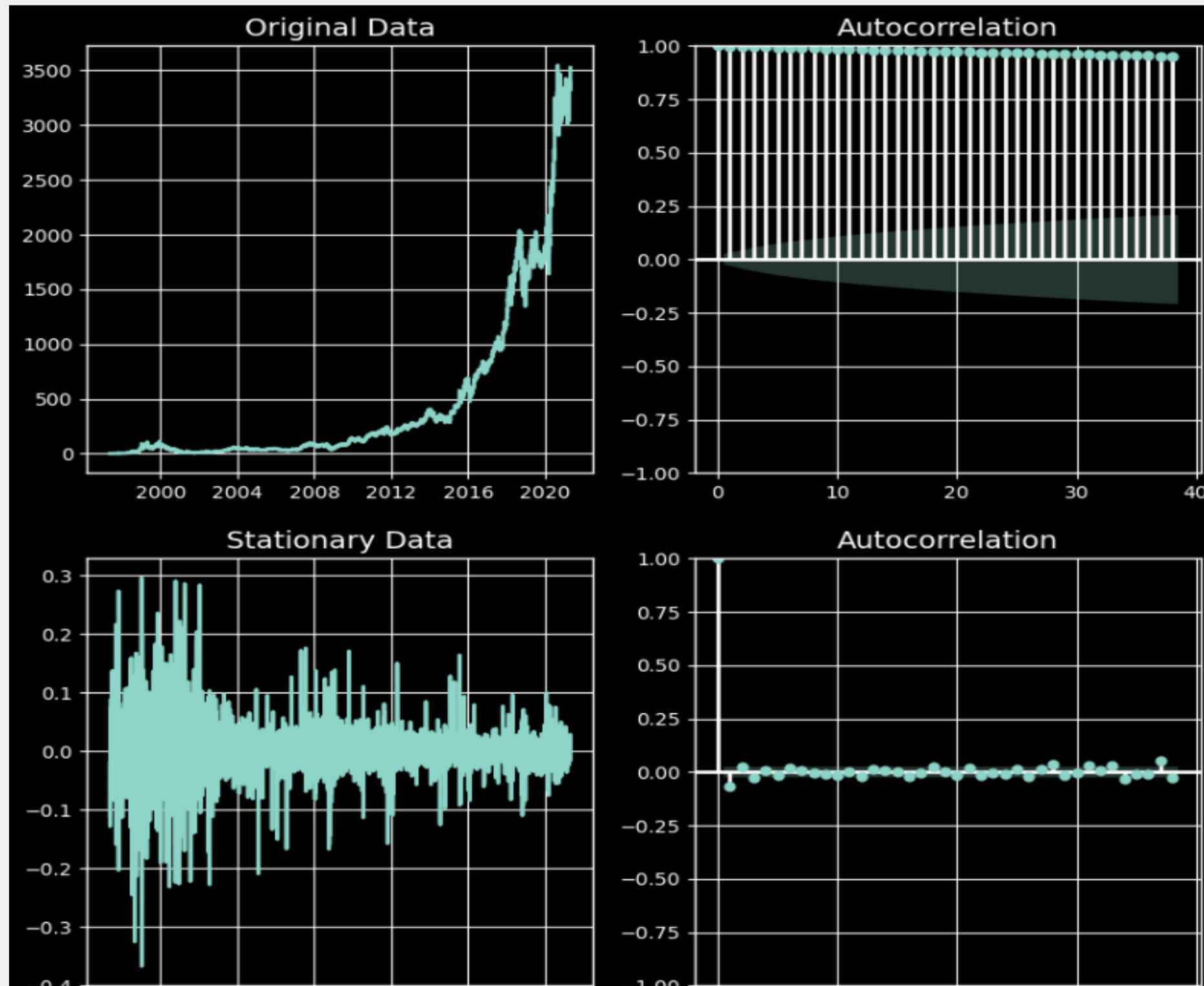
# "Unveiling Amazon's Time Series Secrets: Seasonal Decomposition Offers Insightful Component Analysis"

Seasonal decomposition of the stationary data using the `seasonal_decompose()` function. Seasonal decomposition is a technique used to decompose a time series into its constituent components: trend, seasonality, and residual (or noise).

The `plot()` method is then used to visualize the decomposition, which typically displays four subplots: the original time series, the trend component, the seasonal component, and the residual component. This decomposition helps in understanding the underlying patterns and variations in the time series data, aiding in further analysis and modeling.



# "Decoding Amazon's Stock Trends: Comprehensive Grid Reveals Original Data, Stationarity, and Correlation Insights"

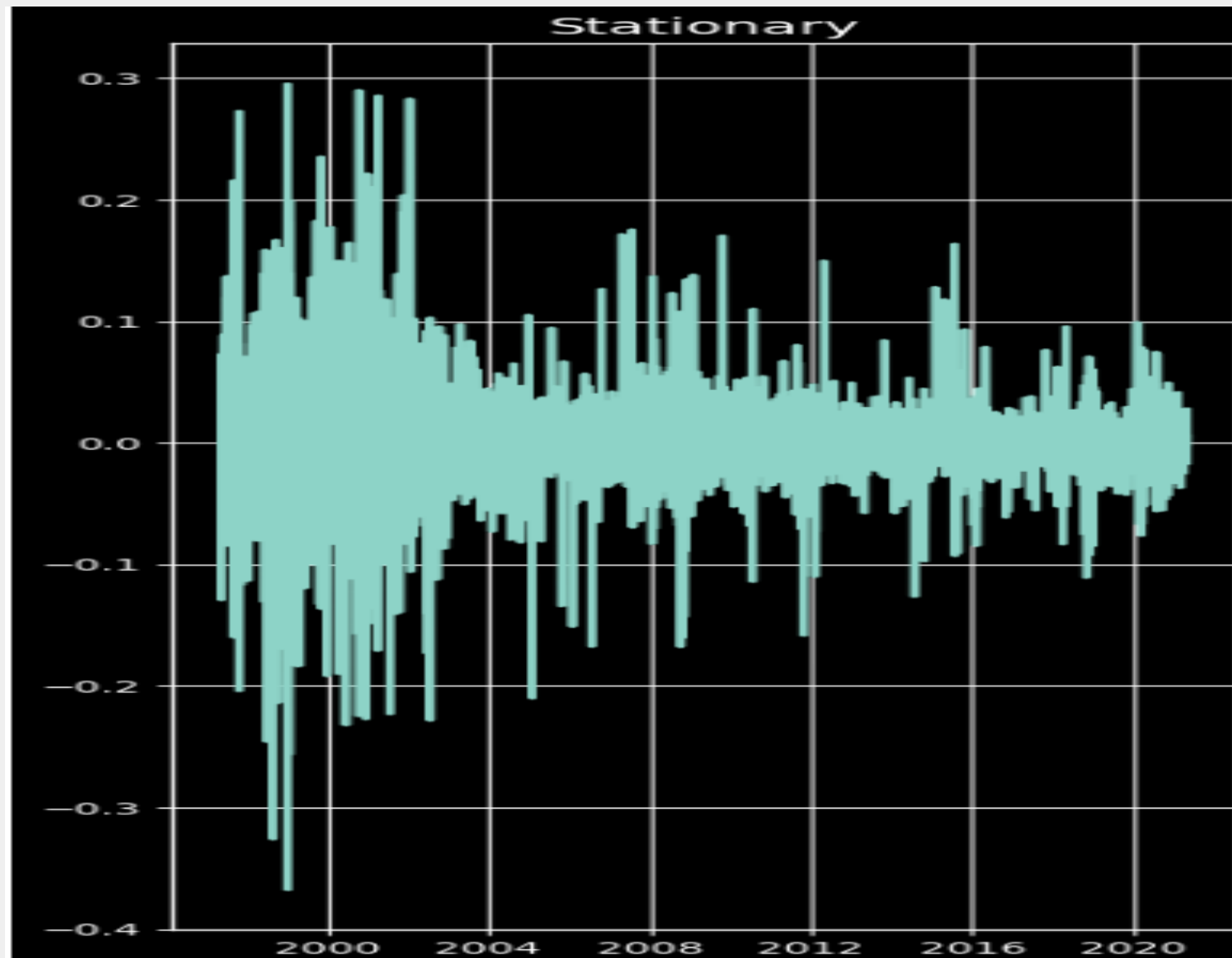


In the top-left subplot, the original time series data of Amazon's opening stock prices is plotted, and it's titled "Original Data". In the top-right subplot, the autocorrelation function (ACF) of the original data is visualized using the `plot_acf()` function. The ACF provides insights into the correlation structure of the original data at different lags, which is crucial for understanding the time series properties and selecting appropriate models.

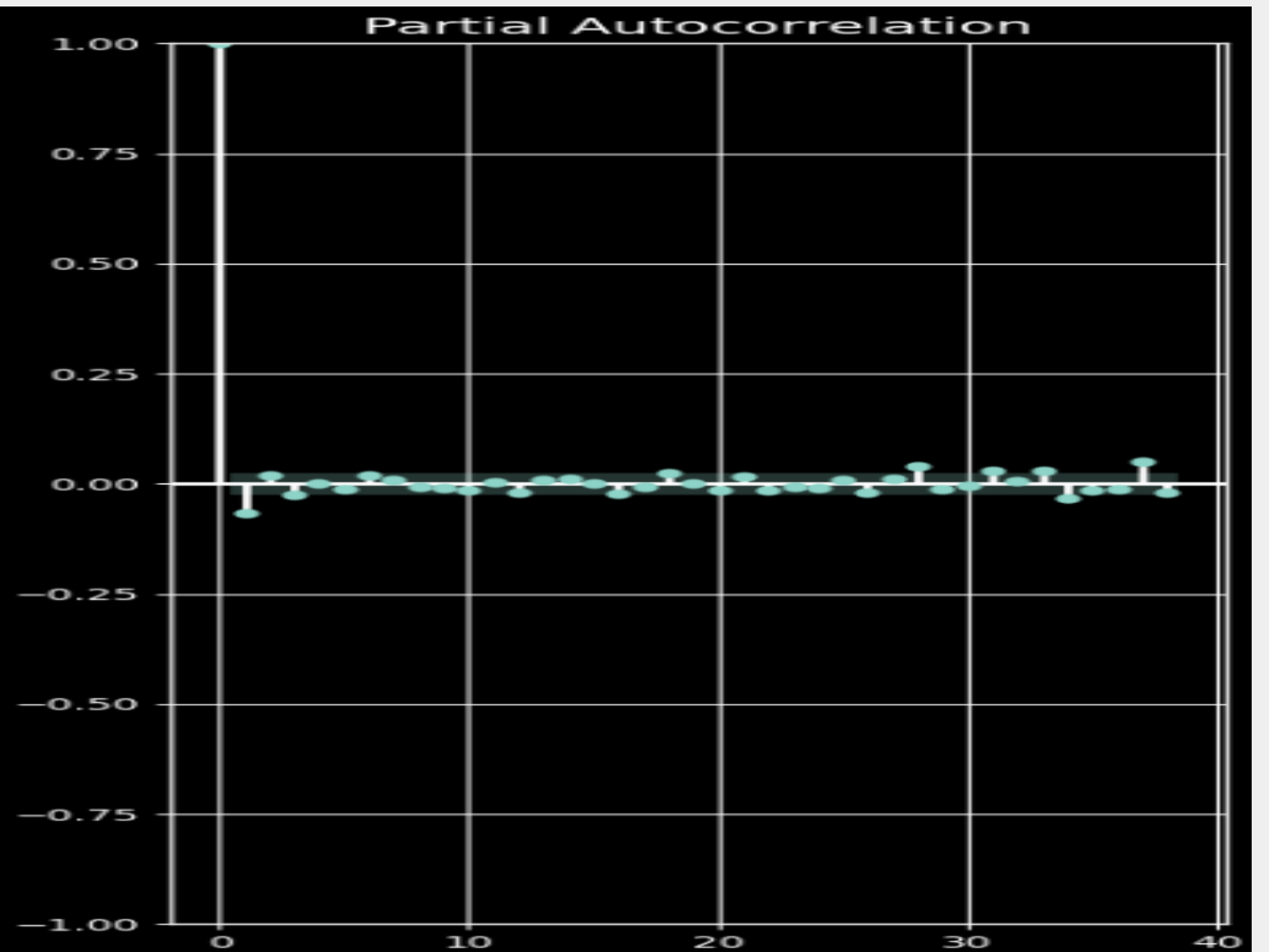
In the bottom-left subplot, the stationary version of the Amazon stock data is plotted, and it's titled "Stationary Data". Stationarity is an important property in time series analysis as it simplifies the modeling process by ensuring that statistical properties such as mean and variance remain

# "Deciphering Amazon's Stock Trends: Dual-panel Insight Unveils Stationarity and Lagged Effects"

Stationary



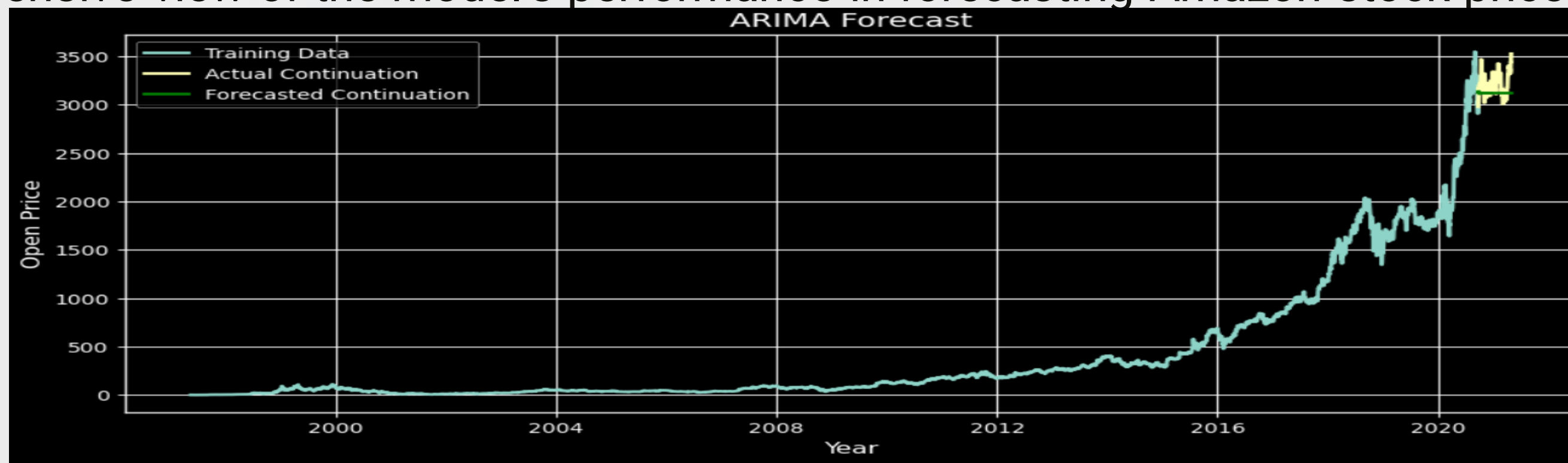
Partial Autocorrelation



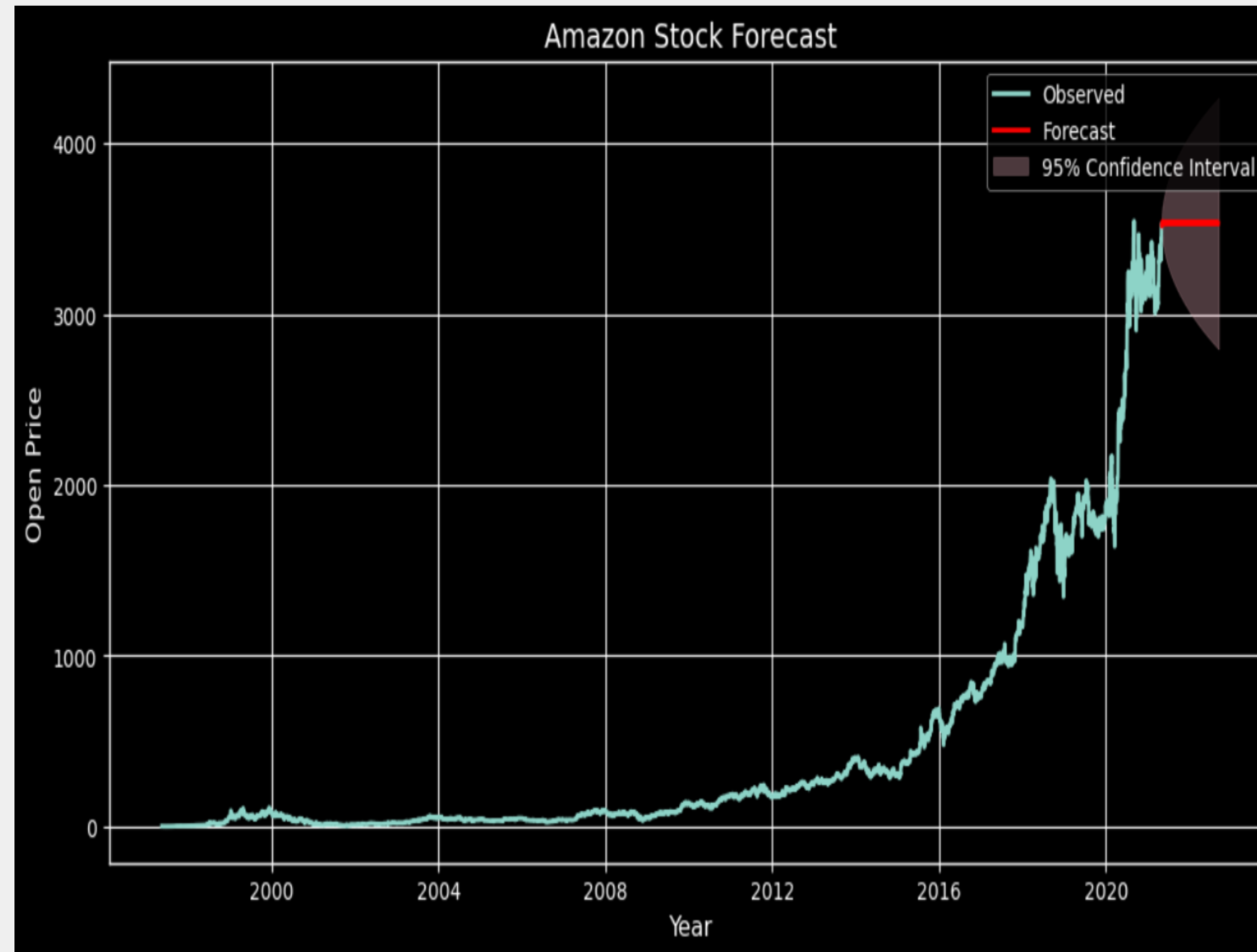


# ARIMA Forecast Unveils Amazon Stock Price Trends with Precision

- visualizes the training data, actual continuation data (test set), and forecasted continuation data using matplotlib. It plots these datasets with distinct labels, including "Training Data" for the training set, "Actual Continuation" for the test set, and "Forecasted Continuation" for the forecasted data.
- Additionally, it adds the title "ARIMA Forecast", places the legend in the upper-left corner, and labels the x-axis as "Year" and the y-axis as "Open Price". This visualization offers a comprehensive view of the model's performance in forecasting Amazon stock prices over time.



# Forecasting Amazon Stock Prices for the Next 500 Days



- forecast for the next 500 days of Amazon stock prices using an ARIMA model, along with 95% confidence intervals.
- It computes the forecast values and confidence intervals, creates a date index for the forecast, and then plots the observed data, forecast, and confidence intervals using Matplotlib.

- The observed data is represented in blue, the forecast in red, and the confidence intervals as a shaded region. The plot provides insights into potential future trends in Amazon stock prices while accounting for uncertainty.

# Conclusion:

Through the development of a time series forecasting model using ARIMA, this project successfully aimed to predict the future stock prices of Amazon. By leveraging historical stock price data and incorporating relevant external factors, the model provided valuable insights into potential future price movements. The analysis enabled informed investment decisions, optimizing personal investment strategies for buying, selling, or holding Amazon stocks to maximize returns and minimize risks.

The project demonstrated the effectiveness of ARIMA in capturing patterns and trends within the stock price data, resulting in accurate and reliable predictions. Evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) confirmed the model's performance and highlighted areas for potential improvement.



# Thank You

---

