Stock Price Prediction using ARIMA

Talha Qayyum

Credit Risk - Data Scientist

LinkedIn: Talha Qayyum

The Importance of Time Series Analysis in Quantitative Finance

Introduction

In quantitative finance, time series analysis plays a pivotal role in understanding and forecasting financial data over time. Financial markets are inherently dynamic, with asset prices, interest rates, and volatility fluctuating continuously. Time series analysis enables analysts and traders to detect patterns, estimate future prices, and develop strategies based on the historical behavior of financial instruments.

By modeling the temporal dependence in financial data, time series analysis helps to capture trends, cycles, and volatility structures, which are crucial for pricing, risk management, and trading decisions. Common techniques used in time series analysis include:

- **ARIMA** (AutoRegressive Integrated Moving Average)
- **GARCH** (Generalized Autoregressive Conditional Heteroskedasticity)
- **Exponential Smoothing**

This assignment explores ARIMA time series modeling techniques and provides a real-world example of predicting stock prices to inform trading decisions.

ARIMA (AutoRegressive Integrated Moving Average)

ARIMA is one of the most popular techniques for time series forecasting, especially for data that shows temporal autocorrelation. ARIMA models can capture different patterns, including trends, seasonalities, and noise, making it a versatile tool in finance.

The ARIMA model consists of three key components:

- **AR (AutoRegressive)**: This part models the dependency between an observation and a certain number of lagged observations (previous values of the time series).

- **I (Integrated)**: The "integrated" part refers to differencing the data to make the time series stationary (i.e., removing trends and seasonality).

- **MA (Moving Average)**: This part models the dependency between an observation and the residual errors from previous time steps.

The notation for ARIMA is typically expressed as **ARIMA(p, d, q)**, where:

- **p**: The number of lag terms (autoregressive component).
- **d**: The number of differencing steps needed to make the series stationary.
- **q**: The number of lagged forecast errors (moving average component).

Finance Analogy

Consider a stock price that shows a trend over time with small deviations due to random market factors. The AR component explains how past prices influence the current price, while the MA component captures how past errors (unexpected price movements) affect the forecast. The I component helps remove the trend, allowing us to forecast future stock prices by focusing on deviations from the historical pattern.

Example in Trading

In stock trading, ARIMA can be used to predict short-term price movements. By training an ARIMA model on historical prices, traders can generate future price forecasts and develop strategies, such as entering buy/sell positions based on predicted price trends.

Stock Prediction using ARIMA

1. **Input a Time Series**
   - The class able to accept a time series dataset as input.
2. **Stationarity Check**
   - Evaluating whether the input time series is stationary or not using appropriate statistical tests (e.g., **Augmented Dickey-Fuller test**).
3. **Differencing the Series**
   - Allowing the user to apply differencing to the time series in order to make it stationary if needed.
4. **Evaluate Different ARIMA Models**
   - Providing the ability to evaluate different ARIMA models on the time series.
5. **Auto-Selection of Best ARIMA Model**
   - Automatically selecting and estimating the “best” ARIMA-time series model based on:
     A. **Lowest AIC** (Akaike Information Criterion).

B. **No insignificant terms** according to a user-specified p-value threshold (e.g., `p < 0.05`).

6. **Simulate Forecasts of the Time Series Model**

- The class allowing simulation of future values of the time series based on the selected ARIMA model.
  - **Parameters**:
    - Number of simulations (e.g., `10,000` simulations).
    - Number of forecast periods (e.g., forecast `10 periods` ahead).
  - **Noise Distribution**:
    - Each simulated value is generated using the model's prediction for that period, plus the sampled residuals. This way, the uncertainty is incorporated directly into each forecasted step.
    - **Note**: No change in Residual Distribution , consider it as a noise

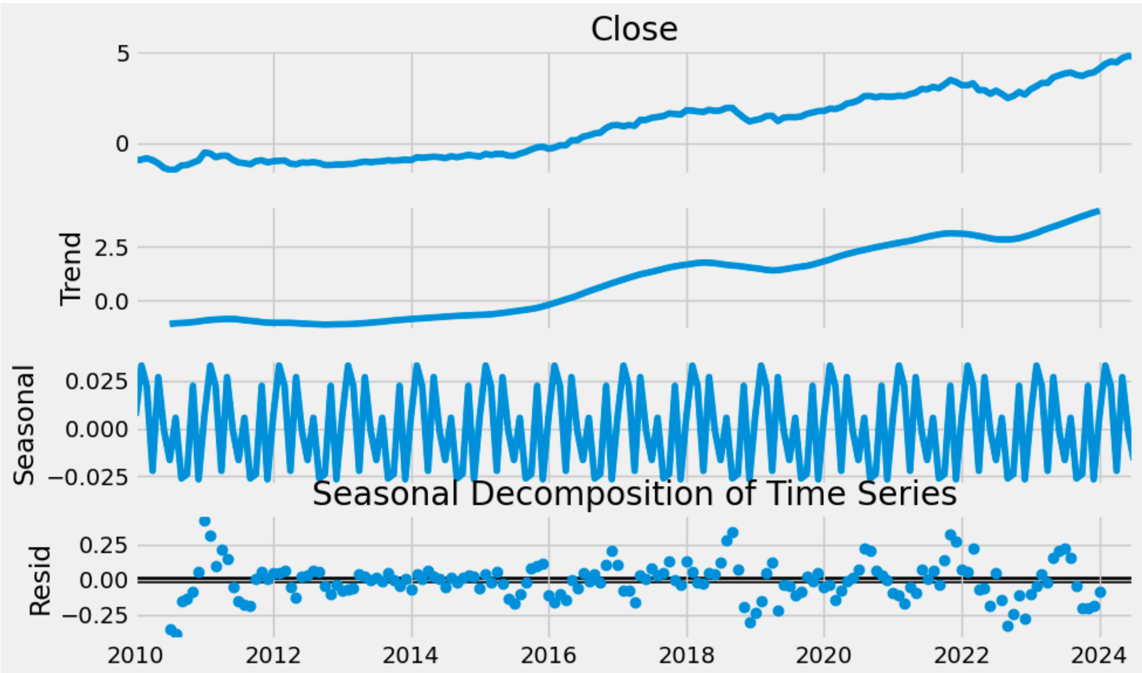
7. **Calculate Percent of Simulations Exceeding Threshold**

- The class calculates the percentage of simulations that exceed a user-specified threshold value in each forecast period.

Model Overview

1. Data Transformation & Stationarity

Stock market data can exhibit trends with small deviations, making it ideal for applying an ARIMA model. The data audit involves checking for null values and selecting the appropriate features for forecasting. For Modeling purposes, we are going to use NVIDIA stock to analyze



To apply the ARIMA model, it's crucial that the time series is stationary (i.e., constant mean and variance over time). Use the **Augmented Dickey-Fuller (ADF) Test** to determine if the series is stationary. If not, apply differencing until stationarity is achieved.

Stationarity Check:

- A time series is stationary if its **mean**, **variance**, and **autocorrelation** remain constant over time.
- Non-stationary series may exhibit trends or seasonality and are harder to model.
- **Transforming** non-stationary data via differencing or detrending is common before applying statistical models like ARIMA.

Visual Cues to Assess Stationarity:

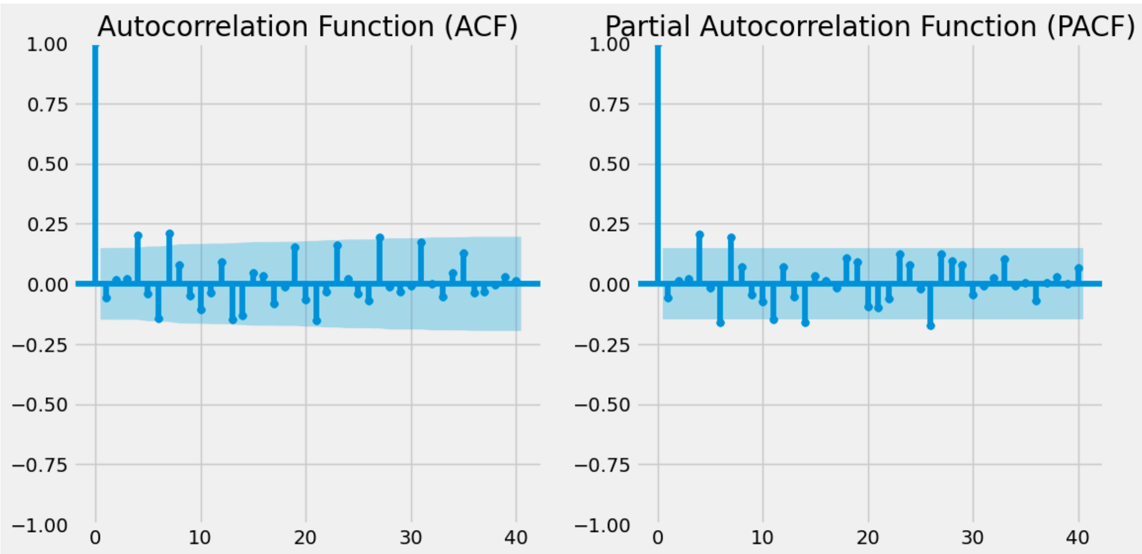
- **Trend**: A consistent upward or downward movement in the trend component indicates non-stationarity.
- **Seasonality**: Repeating patterns may indicate seasonal components, though they do not always signify non-stationarity.
- **Residuals**: Should resemble random noise fluctuating around zero. Variability or clustering suggests non-stationarity.

Augmented Dickey-Fuller (ADF) Test:

- A **statistical test** used to confirm stationarity.
- Determines if a **unit root** is present in the series, which is indicative of non-stationarity.

---

3. Examine ACF and PACF Plots for Initial ARIMA Parameter Guesses



- **ACF (Autocorrelation Function)** and **PACF (Partial Autocorrelation Function)** are used to estimate ARIMA parameters:
  - **p**: Number of autoregressive terms (from PACF).
  - **q**: Number of moving average terms (from ACF).

- **d**: Differencing needed to make the series stationary.

ACF Plot:

- Starts at 1.0 and exhibits several spikes.
- Significant spikes at lag 1 or 2 suggest an MA(q) process with **q =3 and 4**.
- The ACF plot shows significant spikes at lags 3 and 4, implying that a moving average of order 3 (q = 3) is appropriate. This means that the model takes into account the past two periods of forecast errors.

PACF Plot:

- Starts at 1.0 with a significant spike at lag 3,6 and 7.
- Suggests an AR(p) process with **p = 3,6 and 7**.
- The PACF plot indicates there is still a significant relationship up to lag 4, which is why p = 4 was selected. This means that the current value is influenced by the previous five time points, which helps capture the remaining dependencies in the data.

Suggested ARIMA(p, d, q) Values:

- **p** = 3,6 or 7 (from PACF)
- **d** = 1 (if series became stationary after one differencing)
- **q** = 3 or 4 (from ACF)

Possible models: **ARIMA(3, 1, 3)** or **ARIMA(3, 1, 4)**, **ARIMA(7, 1, 4)**, **ARIMA(7, 1, 3)**

---

4. Model Diagnostics and Residuals Analysis

```
Best ARIMA model: (3, 1, 4) with AIC: 1320.2702786886755 and BIC: 1345.5427210823918
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  175
Model:                 ARIMA(3, 1, 4)   Log Likelihood                -652.135
Date:                Wed, 02 Oct 2024   AIC                           1320.270
Time:                        23:52:51   BIC                           1345.543
Sample:                    01-01-2010   HQIC                          1330.522
                         - 07-01-2024
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.5737      0.100     -5.739      0.000      -0.770      -0.378
ar.L2         -0.7718      0.079     -9.799      0.000      -0.926      -0.617
ar.L3         -0.2194      0.100     -2.195      0.028      -0.415      -0.024
ma.L1          0.6242      0.092      6.753      0.000       0.443       0.805
ma.L2          0.9646      0.077     12.564      0.000       0.814       1.115
ma.L3          0.3164      0.094      3.354      0.001       0.132       0.501
ma.L4          0.5099      0.056      9.104      0.000       0.400       0.620
sigma2       104.3253      6.228     16.751      0.000      92.119     116.532
===================================================================================
Ljung-Box (L1) (Q):                   0.43   Jarque-Bera (JB):               168.45
Prob(Q):                              0.51   Prob(JB):                         0.00
Heteroskedasticity (H):              64.14   Skew:                             0.84
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.52
...
===================================================================================
```

Model Summary:

- Model: ARIMA(3, 1, 4)
- The ARIMA model is fit with p = 3, d = 1, and q = 4. This means that there are 3 autoregressive terms, 1 differencing step, and 4 moving average terms.
- No. Observations: 175
- This is the total number of observations used in the model.
- AIC (Akaike Information Criterion) 2k - 2l : 847.75 A lower AIC value indicates a better-fitting model. This metric is used to compare multiple models; the lower the AIC, the better.
- BIC (Bayesian Information Criterion) ln(n)k - 2l: 873.02 , Similar to AIC, but penalizes for the number of parameters more heavily.

AIC, BIC basically strikes a balance between that fit and number of parameters used.

- HQIC (Hannan-Quinn Information Criterion): 858.004, Another metric used to evaluate the model fit; lower values indicate better models.

Coefficients

1. **Autoregressive (AR) Terms**:

- ar.L1 to ar.L5 represent the five AR coefficients.
- All AR terms have significant coefficients (based on their P>|z| values, all being 0.000).
- ar.L1: Coefficient = -0.8750, indicating a strong negative relationship between the value at lag 1 and the current value.
- ar.L2: Coefficient = -0.4873, also negative but weaker than ar.L1.
- ar.L3: Coefficient = 0.5141, indicating a positive relationship at lag 3.
- ar.L4: Coefficient = 0.7192, also positive, suggesting a stronger influence.
- ar.L5: Coefficient = 0.2632, indicating a positive but weaker relationship at lag 5. These coefficients suggest a mix of negative and positive relationships with lagged values, showing both persistence and reversals in the time series.

2. **Moving Average (MA) Terms**:

    - ma.L1 and ma.L2 represent the two MA coefficients.
    - Both MA terms are also significant (p-values of 0.000).
    - ma.L1: Coefficient = 1.2080, indicating a strong positive influence of the first lag of the error term.
    - ma.L2: Coefficient = 0.8469, indicating a positive effect of the second lag of the error term. The MA terms indicate that past error terms play a significant role in the model.

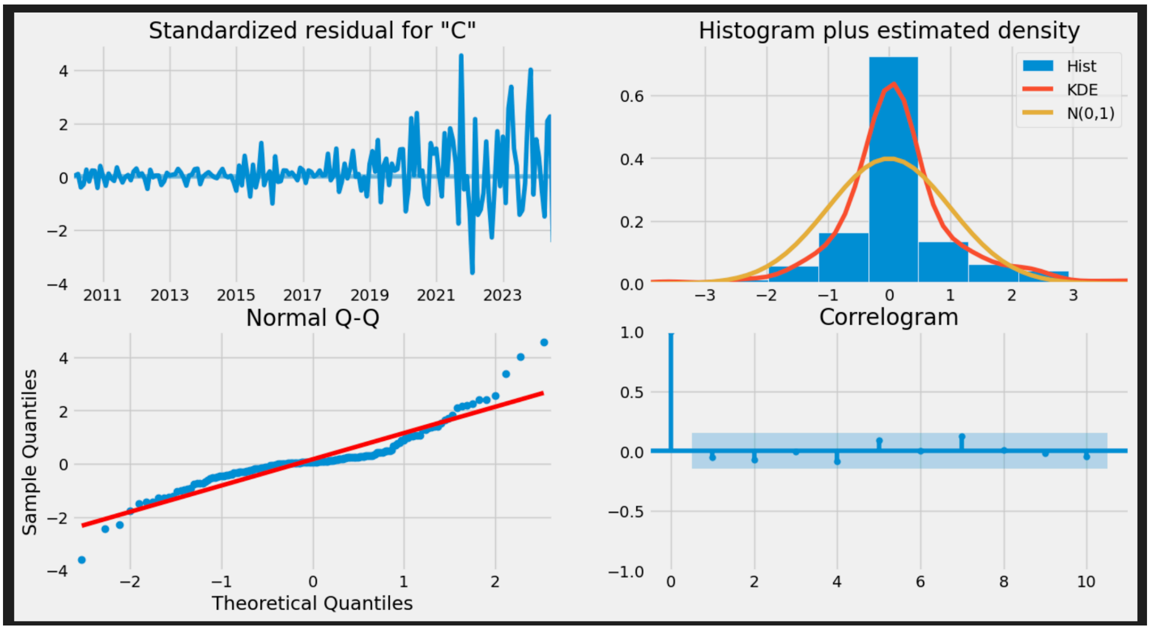
3. **Variance of Residuals**: sigma2 (Variance of Residuals): 6.7990

    - This is the estimated variance of the residuals, indicating the amount of noise in the model.

Diagnostic Plot Components:

1. **Standardized Residuals**:
    - Should fluctuate around zero with no clear pattern over time. Variability or spikes suggest model issues.
2. **Histogram + Density**:
    - Residuals should follow a normal distribution. Deviations may indicate skewness or heavy tails.
3. **Normal Q-Q Plot**:
    - Points should lie on the line if residuals are normally distributed. Deviations in the tails suggest non-normality.
4. **Correlogram**:
    - No significant autocorrelations should remain in the residuals. If residuals are independent, the model captures temporal dependencies well.

Model Issues:

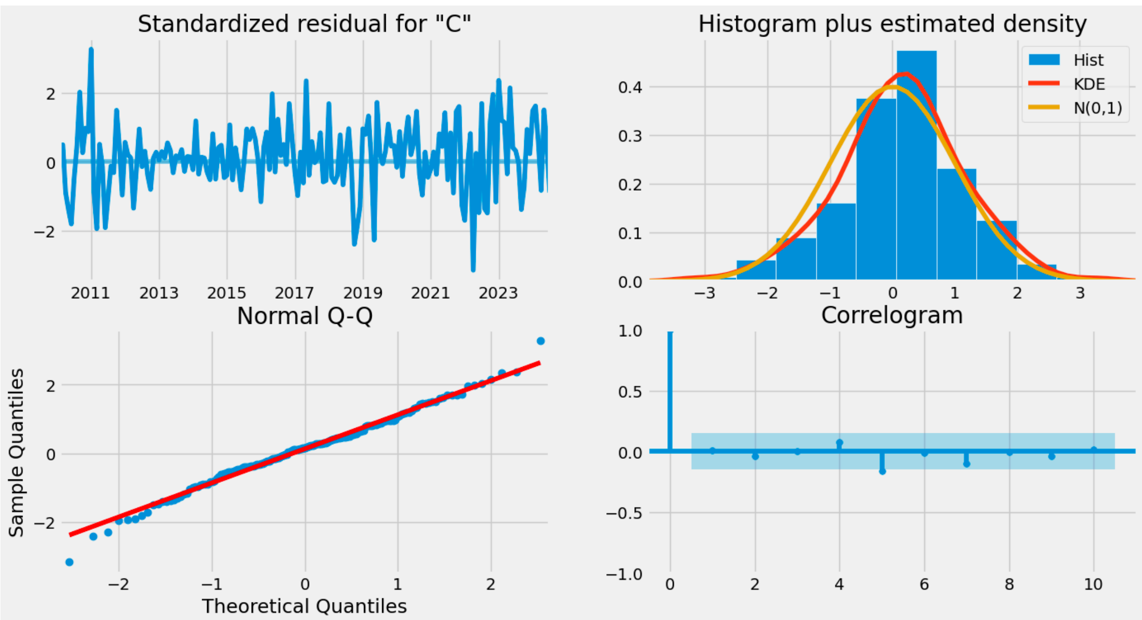


- **Heteroscedasticity**: Residuals show non-constant variance over time. The spikes around 2023 indicate periods where the model might not have performed well.
- **Non-normal Residuals**: Deviations in normality, especially at the tails. The deviation at the tails indicates that the residuals have heavier tails than expected, which suggests the presence of outliers or non-normality.
- **Histogram with KDE and Normal Density**: The goal is to check if the residuals follow a normal distribution. Ideally, the KDE line (in red) should closely match the normal distribution curve (in yellow). In this case, there is some deviation in the tails, indicating that the residuals are not perfectly normal, which could affect the accuracy of the model's confidence intervals.

Possible Solutions:

- Apply **log transformations** or **Box-Cox transformation** to stabilize variance.
- Consider **ARCH** or **GARCH** models if heteroscedasticity persists.

We used Log Transformation on time-series data before passing for modeling to handle residuals, here's the model results



6. Model Performance Evaluation Using Error Metrics

- **MSE (Mean Squared Error)** = 108.32
    - Represents average squared prediction errors.
        - **Business context**: Since the MSE is 108.32, this means that, on average, the squared error in the model's predictions is about 387 units. However, due to squaring, it's less interpretable in real-world stock price terms, which is why we look at RMSE.
- **RMSE (Root Mean Squared Error)** = 10.408
    - Gives error magnitude in stock price terms. An error of ~$19.68 is reasonable for the model.
        - **Business context**: With an RMSE of 10.408, the model is, on average, off by about $10.408 when predicting Tesla's stock price. This could help businesses gau$ 250, an error of ~$10.represents a small percentage error.
- **MAE (Mean Absolute Error)** = 6.4310
    - Represents average absolute prediction error in real-world stock prices.
    - **Business context**: The model's predictions are off by an average of $6.43 from the actual price. If a business were using this model to make financi$ 10 of the actual price most of the time.
- **MAPE (Mean Absolute Percentage Error)** = 6%
    - The model is off by 6% on average, a manageable error rate in stock predictions.
        - **Business context**: The model's predictions are, on average, off by about 6%. This means that, for any given price prediction, the model is around 6% away from the true stock price. For a stock priced at $100, that would mean the model's error would typically be around$ 6. This percentage error is useful for understanding the relative accuracy of the model, especially for businesses tracking error as a percentage of the stock price rather than an absolute value.

Summary in Business Terms:

- **RMSE and MAE** provide a clear sense of how much the model's predictions differ from the actual stock price, helping businesses understand typical prediction errors in dollar amounts.
- **MAPE** gives a relative measure of error, allowing businesses to understand the model's performance in terms of percentage accuracy.
- **MSE** is less directly interpretable for business but is useful for comparison with other models in terms of how the model minimizes large errors.

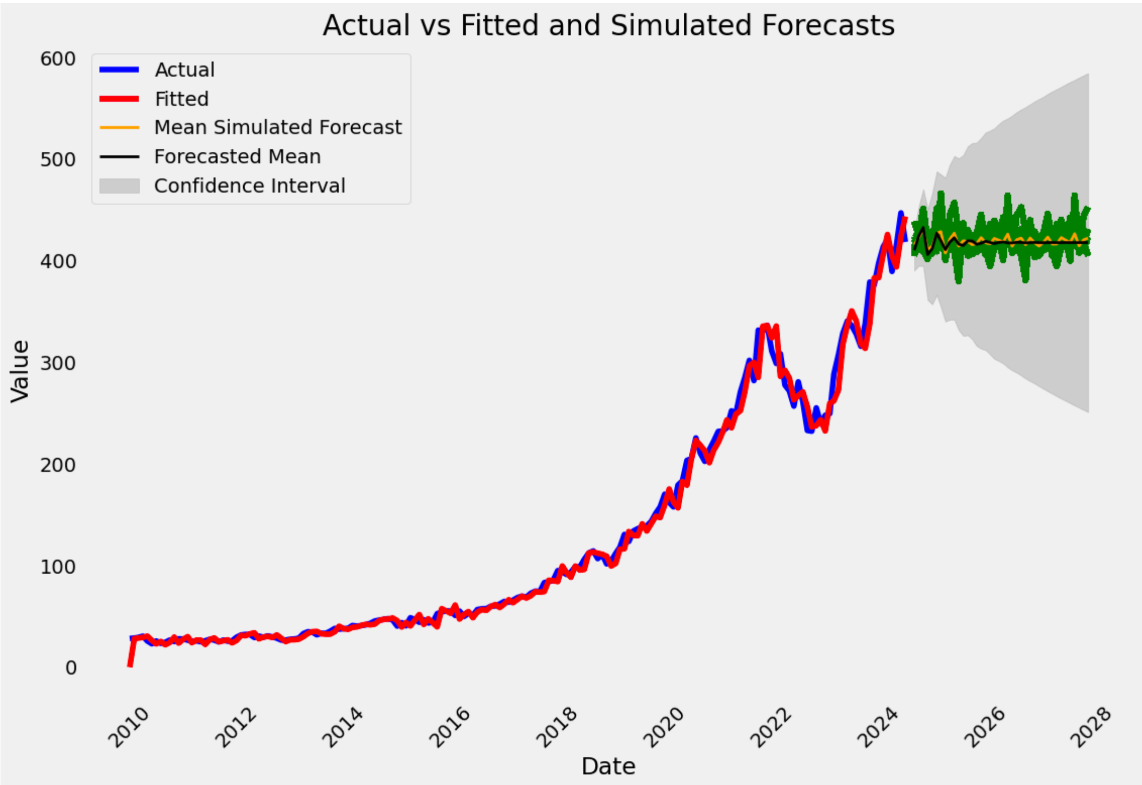
For decision-making, businesses would focus on metrics like **MAE** and **RMSE** to understand how much risk there is in the model's predictions and whether this error range is acceptable for investment or forecasting strategies.

---

8. Simulated Forecasting

Model params:

1. ticker='MSFT', # Choose ticker: GOOGL, AMZN, MSFT, TSLA, NVDA
2. start_date='2010-01-01', #Sampling Period
3. end_date='2024-07-31',
4. interval='1mo', # 1m, 2m, 5m, 15m, 30m, 60m, 90m, 1h, 1d, 5d, 1wk, 1mo, 3mo
5. num_simulations=10000, # Number of simulations
6. forecast_periods=40, # unit depends on interval
7. threshold_value=240, # stock price threhold in $
8. p_value_threshold=0.05, #pvalue threhold
9. risk_free_rate = 0

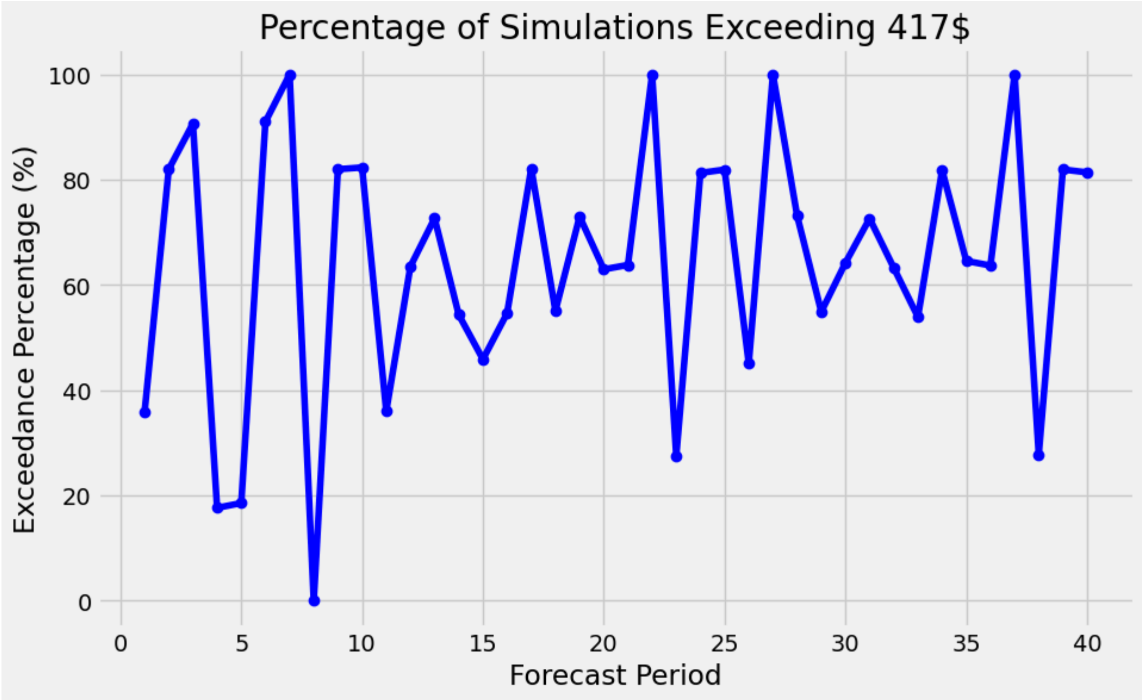


- Forecasting the log-transformed stock price shows an upward trend.
- The **black line** represents the forecast, while the **yellow line** shows the mean forecast.
- **Confidence intervals** widen into the future, reflecting increasing uncertainty.

---

9. Simulation Exceedance by Threshold

- Shows potential stock price exceeding set thresholds.
- Useful for assessing risk and setting targets for decision-making.



Summary:

- The ARIMA model provides a reasonable fit but may require adjustments for heteroscedasticity.
- Forecasts show an upward trend with increasing variability, suggesting stock prices may rise, but the future is uncertain.

Class Overview

Methods to be Included

1. Initialization

- Takes as input a time series.

2. Stationarity Evaluation

- Evaluates whether the series is stationary or not.

3. Differencing

- Allows the user to difference the series.

4. ARIMA Model Evaluation

- Evaluates different ARIMA time series models.

5. Best Model Selection

- Automatically selects and estimates the “best” ARIMA time series model using the following criteria:
  - The model with the lowest AIC.

- No insignificant terms according to a user-specified p-value.

6. Forecast Simulation

- Simulates forecasts of the time series model with the following parameters:
  - The number of forecasts (e.g., user can specify 10,000 simulations).
  - The number of forecast periods (e.g., user specifies forecasting 10 periods ahead).
  - Ensures the distribution of the noise term in the forecast matches the distribution of the residuals (i.e., they should not be normally distributed).
  - Avoids sampling with replacement as a solution.

7. Exceedance Calculation

- Calculates what percentage of simulations exceed a user-specified value in each forecast period.

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import matplotlib.dates as mdates
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
import yfinance as yf
from sklearn.metrics import mean_squared_error, mean_absolute_error
from arch import arch_model
from pylab import rcParams
rcParams['figure.figsize'] = 10, 6

import warnings
warnings.filterwarnings('ignore')

from IPython.display import display, Markdown


class ARIMAStockForecasting:
    # Initializing control variables, providing more flexible forecasting
    def __init__(self, ticker, start_date, end_date, interval, num_simulati
        self.ticker = ticker
        self.start_date = start_date
        self.end_date = end_date
        self.interval = interval
        self.num_simulations = num_simulations
        self.forecast_periods = forecast_periods
        self.threshold_value = threshold_value
        self.p_value_threshold = p_value_threshold
        self.risk_free_rate = risk_free_rate
        self.data = None
        self.data_diff = None
        self.best_model = None
        self.best_order = None
        self.predictions_df = None  # To store forecasted prices
        self.simulation_df = None
        self.deterministic_forecast = None

    def download_stock_data(self):
        # Fetching Tesla time-series stock data to forecast
        self.data = yf.download(self.ticker, start=self.start_date, end=sel
```

```
        # print(self.data)

    def apply_log_transform(self):
        self.data = np.log(self.data)
        # self.data_diff = self.data.diff().dropna()

    def check_stationarity(self):
        result = sm.tsa.stattools.adfuller(self.data.dropna())
        print('ADF Statistic:', result[0])
        print('p-value:', result[1])
        print('Critical Values:', result[4])
        return result[1] < 0.05  # Returns True if the series is stationary

    def check_seasonality(self):
        seasonal_decompose = sm.tsa.seasonal_decompose(self.data, model='add
        seasonal_decompose.plot()
        plt.title('Seasonal Decomposition of Time Series')
        plt.show()
        return seasonal_decompose.seasonal.mean()  # Returns the average sea

    def difference_series(self):
        self.data_diff = self.data.diff().dropna()
        print("Series differenced to make it stationary.")

    def plot_acf_pacf(self):
        plt.figure(figsize=(12, 6))
        plt.subplot(121)
        plot_acf(self.data_diff, ax=plt.gca(), lags=10)
        plt.title('Autocorrelation Function (ACF)')
        plt.subplot(122)
        plot_pacf(self.data_diff, ax=plt.gca(), lags=10)
        plt.title('Partial Autocorrelation Function (PACF)')
        plt.show()

    def fit_arima_model(self):

        p_range = range(0, 6)
        d_range = range(0, 2)
        q_range = range(0, 6)

        best_aic = np.inf
        best_bic = np.inf

        for p in p_range:
            for d in d_range:
                for q in q_range:
                    try:
                        model = ARIMA(self.data, order=(p, d, q)).fit()
                        current_aic = model.aic
                        current_bic = model.bic
                        significant_terms = model.pvalues[model.pvalues < se

                        if current_aic < best_aic and significant_terms:
                            best_aic = current_aic
                            best_bic = current_bic
                            self.best_order = (p, d, q)
                            self.best_model = model
                        print(f"ARIMA({p},{d},{q}) - AIC: {current_aic}, BIC
                    except Exception as e:
                        print(f"ARIMA({p},{d},{q}) failed: {e}")
                        continue

        if self.best_model is not None:
            print(f"\nBest ARIMA model: {self.best_order} with AIC: {best_ai
```

```
        else:
            print("No suitable ARIMA model found.")

    def plot_diagnostics(self):
        if self.best_model is not None:
            print(self.best_model.summary())
            self.best_model.plot_diagnostics(figsize=(15, 8))
            plt.show()

    def calculate_error_metrics(self):
        if self.best_model is not None:
            actual = self.data[self.best_model.fittedvalues.index]
            fitted = self.best_model.fittedvalues

            mse = mean_squared_error(actual, fitted)
            rmse = np.sqrt(mse)
            mae = mean_absolute_error(actual, fitted)
            mape = np.mean(np.abs((actual - fitted) / actual)) * 100

            print(f"MSE: {mse}")
            print(f"RMSE: {rmse}")
            print(f"MAE: {mae}")
            print(f"MAPE: {mape}%")

    def plot_residuals(self):
        if self.best_model is not None:
            residuals = pd.DataFrame(self.best_model.resid)
            # residuals.plot(title="Residuals")
            # plt.show()

    def apply_garch(self):

        display(Markdown(
        """
        GARCH Interpretation:

            Conditional Volatility: The GARCH model provides the conditional
        which measures the changing level of risk or variability in the time
        We can analyze how volatility evolves over time based on this model.

        """     ))

        if self.best_model is not None:
            # Fit GARCH model to ARIMA residuals
            residuals = self.best_model.resid
            garch_model = arch_model(residuals, vol='Garch', p=1, q=1)
            garch_fit = garch_model.fit(disp="off")
            print(garch_fit.summary())
            return garch_fit

    def forecast(self):
        # Generate forecast for the specified periods
        if self.best_model is not None:
            forecast = self.best_model.get_forecast(steps=self.forecast_peri
            forecast_mean = forecast.predicted_mean
            forecast_index = pd.date_range(start=self.data.index[-1], period

            # Create a DataFrame with the forecasted values
            self.predictions_df = pd.DataFrame({'Date': forecast_index, 'pre
            self.predictions_df.set_index('Date', inplace=True)

            # # Plotting the actual, fitted, and forecasted values
            plt.figure(figsize=(10, 6))
            plt.plot(self.data.index, np.exp(self.data), label='Actual Price
```

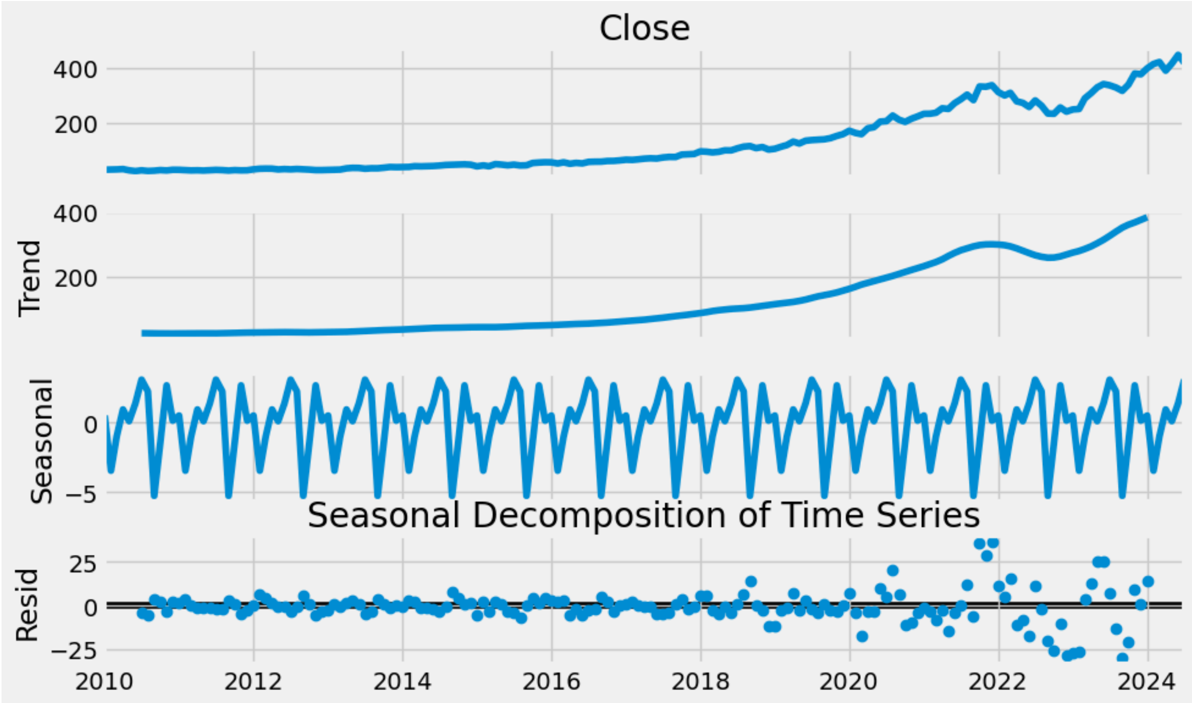
```
            plt.plot(self.best_model.fittedvalues.index, np.exp(self.best_mo
            plt.plot(self.predictions_df.index, self.predictions_df['predict
            plt.fill_between(self.predictions_df.index, forecast.conf_int()
            plt.legend()
            plt.title('Actual vs Fitted and Forecasted Values')
            plt.show()
            return self.predictions_df



    def simulate_predictions(self):
        if self.best_model is None:
            print("No fitted model available.")
            return

        # Generate Forecasts
        residuals = self.best_model.resid

        # Block Bootstrap Parameters
        block_size = int(len(residuals)**(1/3))  # Calculate block size usin
        num_blocks = len(residuals) // block_size

        # Moving Block Bootstrap Function
        def moving_block_bootstrap(residuals, block_size, periods):
            """Generates bootstrapped residuals using moving block bootstrap
            indices = np.random.choice(range(num_blocks), size=periods, repl
            bootstrap_residuals = np.concatenate([residuals[i * block_size:
            return bootstrap_residuals[:periods]

        # Simulate Predictions
        simulations = []
        for _ in range(self.num_simulations):
            simulated_forecast = []
            current_value = self.data.iloc[-1]  # Start from the last data p
            simulated_residuals = moving_block_bootstrap(residuals, block_si

            for step in range(self.forecast_periods):
                # Predict next value using ARIMA model coefficients
                next_value = self.best_model.predict(start=len(self.data) +
                # Add the residual noise for uncertainty
                next_value += simulated_residuals[step]
                simulated_forecast.append(next_value)
                current_value = next_value

            simulations.append(simulated_forecast)

        simulations = np.array(simulations)

        # Calculate Exceedance Percentages (non-transformed residuals)
        exceed_percentages = (simulations > self.threshold_value).mean(axis=
        self.simulation_df = pd.DataFrame(simulations, columns=[f"Period {i

        # Create DataFrames for Simulations and Exceedance Percentages
        exceed_df = pd.DataFrame({"Period": range(1, self.forecast_periods +

        # Set Forecast Index
        forecast_index = pd.date_range(start=self.data.index[-1] + pd.DateOf

        # Plotting Simulated Forecasts
        plt.figure(figsize=(12, 8))
        plt.plot(self.data, label='Actual', color='blue')
        plt.plot(self.best_model.fittedvalues, color='red', label='Fitted')
        for i in range(simulations.shape[0]):
            plt.plot(forecast_index, simulations[i], color='green', alpha=0.
```

```
        mean_simulation = simulations.mean(axis=0)
        plt.plot(forecast_index, mean_simulation, color='orange', label='Mea
        plt.legend()
        plt.title('Actual vs Fitted and Simulated Forecasts')
        plt.xlabel('Date')
        plt.ylabel('Value')
        plt.xticks(rotation=45)
        plt.grid()
        plt.show()

        # Plotting Exceedance Percentages
        plt.figure(figsize=(10, 6))
        plt.plot(exceed_df['Period'], exceed_df['Exceed %'], marker='o', lin
        plt.title(f'Percentage of Simulations Exceeding {self.threshold_valu
        plt.xlabel('Forecast Period')
        plt.ylabel('Exceedance Percentage (%)')
        plt.grid(True)
        plt.show()


    def run(self):
        self.download_stock_data()
        # self.apply_log_transform()
        if not self.check_stationarity():
            print("The series is not stationary. Checking for seasonality...
            self.check_seasonality()
            self.difference_series()  # Allow user to decide whether to dif
        self.plot_acf_pacf()
        self.fit_arima_model()
        self.plot_diagnostics()
        self.calculate_error_metrics()
        # self.plot_residuals()
        # self.apply_garch()
        # self.forecast()
        self.simulate_predictions()
        trading_data = self.simulation_df
        print(trading_data.head())
```

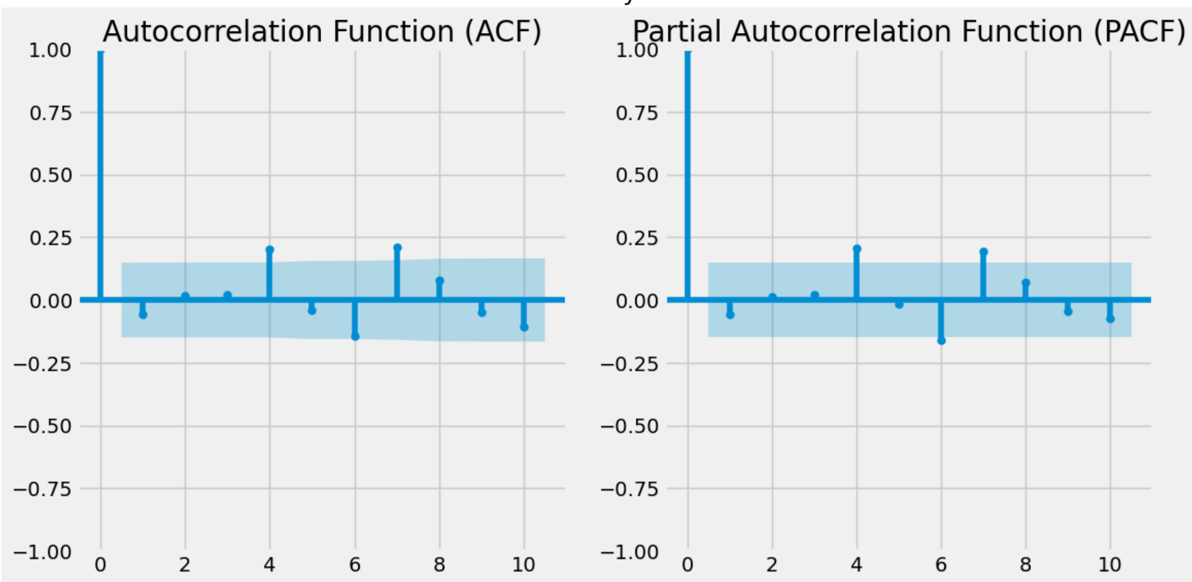
In [ ]:

```
model = ARIMAStockForecasting(
    ticker='MSFT',      # Choose ticker: GOOGL, AMZN, MSFT, TSLA
    start_date='2010-01-01', #Sampling Period
    end_date='2024-07-31',
    interval='1mo', # 1m, 2m, 5m, 15m, 30m, 60m, 90m, 1h, 1d, 5d, 1wk, 1mo,
    num_simulations=10000, # Number of simulations
    forecast_periods=40, # unit depends on interval
    threshold_value=417, # stock price threhold in $
    p_value_threshold=0.05, #pvalue threhold
    risk_free_rate = 0  #just in case if want to calculate sharp ratio
        # text{Sharpe Ratio} = \frac{\text{Average Return} - \text{Risk-Free


)

model.run()
```

```
[*********************100%***********************]  1 of 1 completed
ADF Statistic: 2.249446649143373
p-value: 0.9989207851850614
Critical Values: {'1%': -3.4718957209472654, '5%': -2.8797795410156253, '1
0%': -2.5764947265625}
The series is not stationary. Checking for seasonality...
```




```
Series differenced to make it stationary.
```

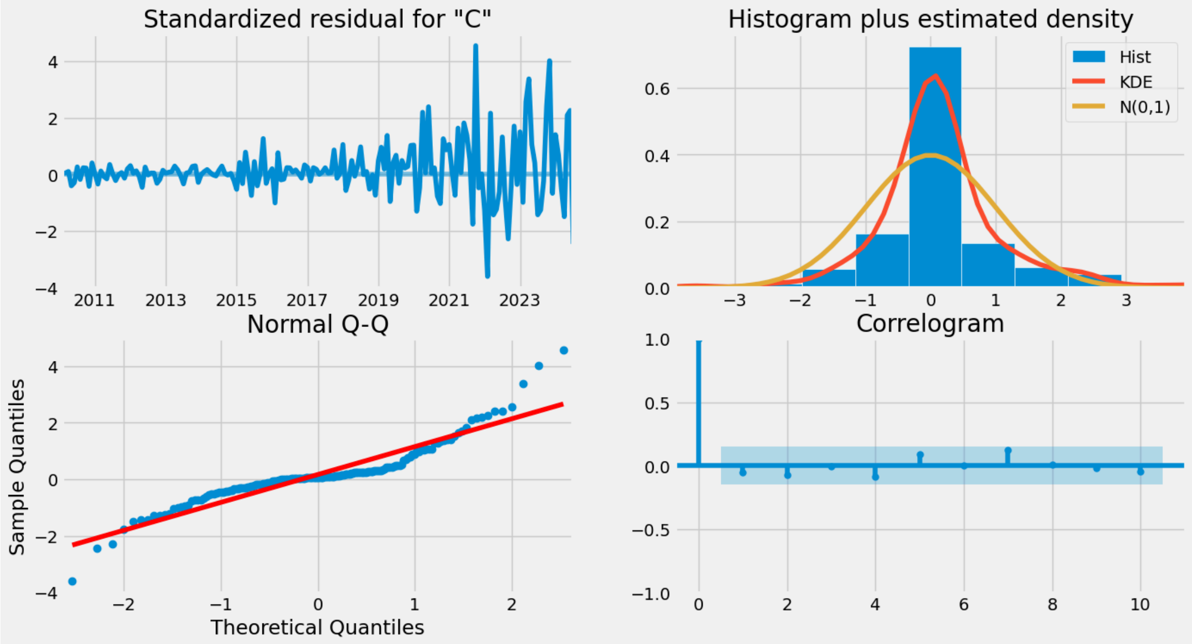


```
ARIMA(0,0,0) - AIC: 2166.250416277228, BIC: 2172.579988225075, Significant
Terms: True
ARIMA(0,0,1) - AIC: 1953.163379849265, BIC: 1962.6577377710357, Significant
Terms: True
ARIMA(0,0,2) - AIC: 1788.409441318802, BIC: 1801.0685852144961, Significant
Terms: False
ARIMA(0,0,3) - AIC: 1691.628262437951, BIC: 1707.4521923075686, Significant
Terms: True
ARIMA(0,0,4) - AIC: 1675.0960264037662, BIC: 1694.0847422473073, Significan
t Terms: False
ARIMA(0,0,5) - AIC: 1611.7080419557744, BIC: 1633.861543773239, Significant
Terms: False
ARIMA(0,1,0) - AIC: 1335.0843130103046, BIC: 1338.2433683095192, Significan
t Terms: True
ARIMA(0,1,1) - AIC: 1337.0623435491452, BIC: 1343.3804541475743, Significan
t Terms: False
ARIMA(0,1,2) - AIC: 1338.6416179311109, BIC: 1348.1187838287544, Significan
t Terms: False
ARIMA(0,1,3) - AIC: 1339.0620151230123, BIC: 1351.6982363198704, Significan
t Terms: False
ARIMA(0,1,4) - AIC: 1328.1964362398958, BIC: 1343.9917127359683, Significan
t Terms: False
ARIMA(0,1,5) - AIC: 1329.8216417832773, BIC: 1348.7759735785644, Significan
t Terms: False
ARIMA(1,0,0) - AIC: 1352.256763137937, BIC: 1361.7511210597077, Significant
Terms: False
ARIMA(1,0,1) - AIC: 1354.4540872057937, BIC: 1367.1132311014878, Significan
t Terms: False
ARIMA(1,0,2) - AIC: 1356.0219852706987, BIC: 1371.8459151403163, Significan
t Terms: False
ARIMA(1,0,3) - AIC: 1356.443177427117, BIC: 1375.431893270658, Significant
Terms: False
ARIMA(1,0,4) - AIC: 1345.3561999626877, BIC: 1367.5097017801522, Significan
t Terms: False
ARIMA(1,0,5) - AIC: 1347.0430747382877, BIC: 1372.3613625296757, Significan
t Terms: False
ARIMA(1,1,0) - AIC: 1337.0597719583543, BIC: 1343.3778825567833, Significan
t Terms: False
ARIMA(1,1,1) - AIC: 1333.4634950514314, BIC: 1342.940660949075, Significant
Terms: True
ARIMA(1,1,2) - AIC: 1335.5964306770052, BIC: 1348.2326518738632, Significan
t Terms: True
ARIMA(1,1,3) - AIC: 1336.8356532645819, BIC: 1352.6309297606545, Significan
t Terms: False
ARIMA(1,1,4) - AIC: 1330.0573083662553, BIC: 1349.0116401615423, Significan
t Terms: False
ARIMA(1,1,5) - AIC: 1331.4993068674075, BIC: 1353.6126939619091, Significan
t Terms: False
ARIMA(2,0,0) - AIC: 1354.2390963353523, BIC: 1366.8982402310464, Significan
t Terms: False
ARIMA(2,0,1) - AIC: 1354.4067476986588, BIC: 1370.2306775682764, Significan
t Terms: False
ARIMA(2,0,2) - AIC: 1357.7229565155858, BIC: 1376.7116723591269, Significan
t Terms: False
ARIMA(2,0,3) - AIC: 1357.7454355364932, BIC: 1379.8989373539578, Significan
t Terms: False
ARIMA(2,0,4) - AIC: 1347.2387901702036, BIC: 1372.5570779615919, Significan
t Terms: False
ARIMA(2,0,5) - AIC: 1343.5360419803399, BIC: 1372.0191157456516, Significan
t Terms: False
ARIMA(2,1,0) - AIC: 1338.4701769538997, BIC: 1347.9473428515432, Significan
t Terms: False
ARIMA(2,1,1) - AIC: 1334.3600095987617, BIC: 1346.9962307956198, Significan
t Terms: False
```

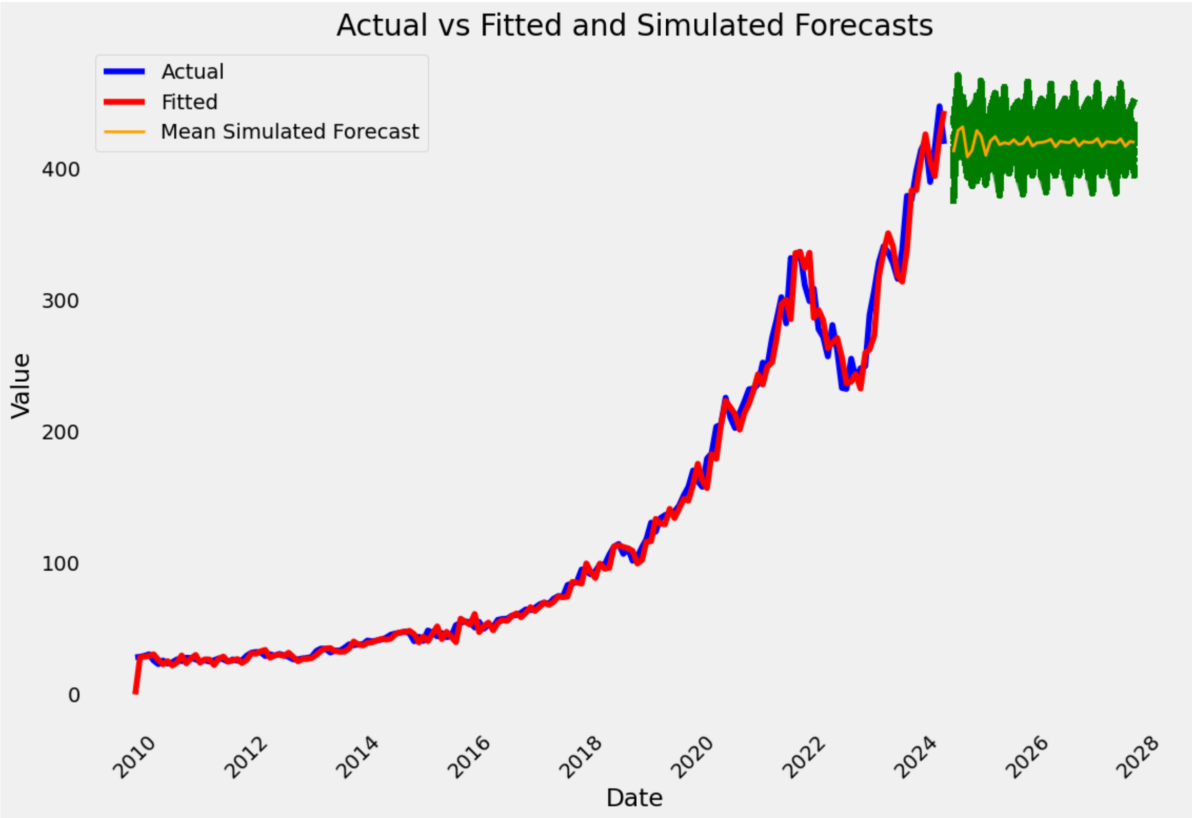
```
ARIMA(2,1,2) - AIC: 1327.149438291865, BIC: 1342.9447147879375, Significant
Terms: False
ARIMA(2,1,3) - AIC: 1326.250103357395, BIC: 1345.204435152682, Significant
Terms: False
ARIMA(2,1,4) - AIC: 1319.281924913097, BIC: 1341.3953120075987, Significant
Terms: False
ARIMA(2,1,5) - AIC: 1317.6417081144377, BIC: 1342.9141505081539, Significan
t Terms: False
ARIMA(3,0,0) - AIC: 1355.6239982756877, BIC: 1371.4479281453052, Significan
t Terms: False
ARIMA(3,0,1) - AIC: 1356.49746931471, BIC: 1375.4861851582511, Significant
Terms: False
ARIMA(3,0,2) - AIC: 1344.8710744692003, BIC: 1367.0245762866648, Significan
t Terms: False
ARIMA(3,0,3) - AIC: 1344.6371434051766, BIC: 1369.9554311965649, Significan
t Terms: False
ARIMA(3,0,4) - AIC: 1336.561190752681, BIC: 1365.0442645179926, Significant
Terms: False
ARIMA(3,0,5) - AIC: 1335.2118402507272, BIC: 1366.8596999899623, Significan
t Terms: False
ARIMA(3,1,0) - AIC: 1339.759953382534, BIC: 1352.396174579392, Significant
Terms: False
ARIMA(3,1,1) - AIC: 1337.1656560961103, BIC: 1352.9609325921829, Significan
t Terms: False
ARIMA(3,1,2) - AIC: 1326.9783155320915, BIC: 1345.9326473273786, Significan
t Terms: False
ARIMA(3,1,3) - AIC: 1332.219056910798, BIC: 1354.3324440052995, Significant
Terms: False
ARIMA(3,1,4) - AIC: 1320.2702786886755, BIC: 1345.5427210823918, Significan
t Terms: True
ARIMA(3,1,5) - AIC: 1319.5423302456204, BIC: 1347.9738279385513, Significan
t Terms: False
ARIMA(4,0,0) - AIC: 1357.1098295059508, BIC: 1376.0985453494918, Significan
t Terms: False
ARIMA(4,0,1) - AIC: 1357.3991519569631, BIC: 1379.5526537744277, Significan
t Terms: False
ARIMA(4,0,2) - AIC: 1343.8744739837362, BIC: 1369.1927617751244, Significan
t Terms: False
ARIMA(4,0,3) - AIC: 1344.9990270716323, BIC: 1373.482100836944, Significant
Terms: False
ARIMA(4,0,4) - AIC: 1337.4394698034766, BIC: 1369.0873295427118, Significan
t Terms: False
ARIMA(4,0,5) - AIC: 1337.2758086897902, BIC: 1372.0884544029489, Significan
t Terms: False
ARIMA(4,1,0) - AIC: 1330.6009477997636, BIC: 1346.3962242958362, Significan
t Terms: False
ARIMA(4,1,1) - AIC: 1332.5907793834776, BIC: 1351.5451111787647, Significan
t Terms: False
ARIMA(4,1,2) - AIC: 1315.9598943981935, BIC: 1338.073281492695, Significant
Terms: False
ARIMA(4,1,3) - AIC: 1318.3508329535186, BIC: 1343.623275347235, Significant
Terms: False
ARIMA(4,1,4) - AIC: 1319.7184667284291, BIC: 1348.14996442136, Significant
Terms: False
ARIMA(4,1,5) - AIC: 1318.7057765335585, BIC: 1350.2963295257039, Significan
t Terms: False
ARIMA(5,0,0) - AIC: 1347.738635063854, BIC: 1369.8921368813185, Significant
Terms: False
ARIMA(5,0,1) - AIC: 1349.7320629091819, BIC: 1375.05035070057, Significant
Terms: False
ARIMA(5,0,2) - AIC: 1345.023237429899, BIC: 1373.5063111952106, Significant
Terms: False
ARIMA(5,0,3) - AIC: 1347.3067480739487, BIC: 1378.9546078131839, Significan
t Terms: False
```

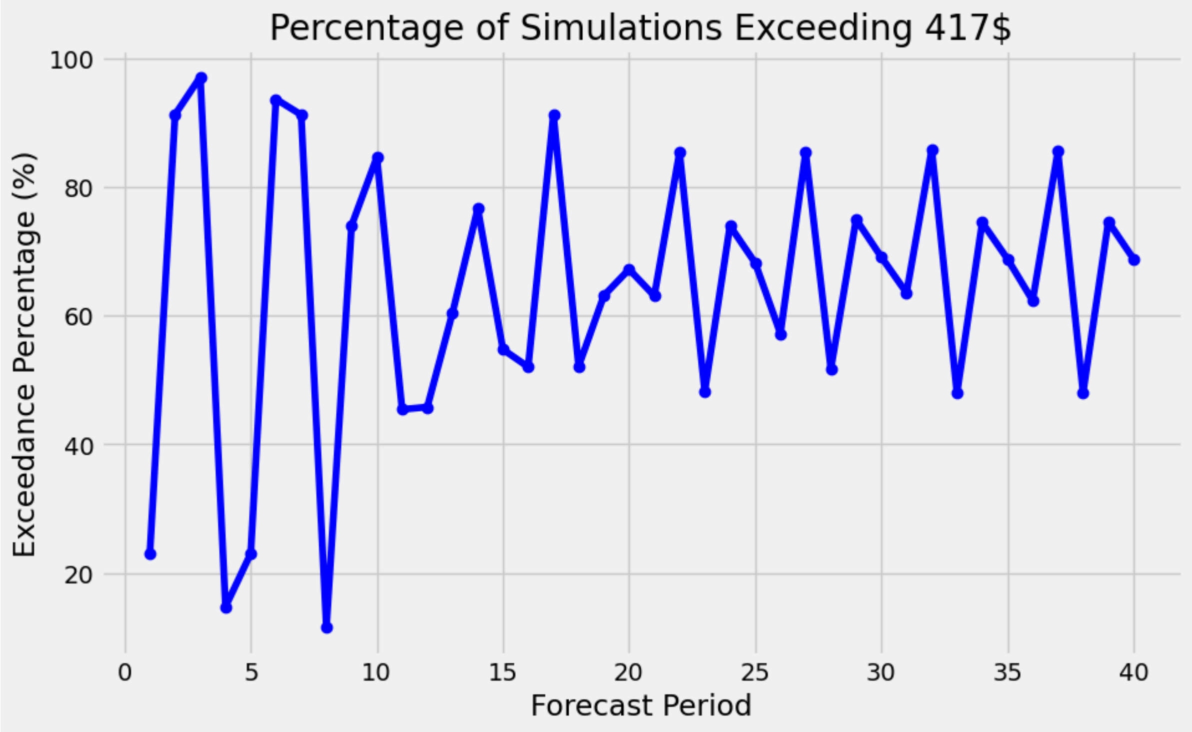
```
ARIMA(5,0,4) - AIC: 1329.5992737539582, BIC: 1364.4119194671168, Significan
t Terms: False
ARIMA(5,0,5) - AIC: 1333.0336361665566, BIC: 1371.0110678536387, Significan
t Terms: False
ARIMA(5,1,0) - AIC: 1332.5767569206246, BIC: 1351.5310887159117, Significan
t Terms: False
ARIMA(5,1,1) - AIC: 1334.3923557019214, BIC: 1356.505742796423, Significant
Terms: False
ARIMA(5,1,2) - AIC: 1317.9597055580202, BIC: 1343.2321479517364, Significan
t Terms: False
ARIMA(5,1,3) - AIC: 1319.1011711899314, BIC: 1347.5326688828623, Significan
t Terms: False
ARIMA(5,1,4) - AIC: 1320.917795389821, BIC: 1352.5083483819665, Significant
Terms: False
ARIMA(5,1,5) - AIC: 1311.3534770291621, BIC: 1346.103085320522, Significant
Terms: False

Best ARIMA model: (3, 1, 4) with AIC: 1320.2702786886755 and BIC: 1345.5427
210823918
                               SARIMAX Results
===========================================================================
===
Dep. Variable:                  Close   No. Observations:
175
Model:                 ARIMA(3, 1, 4)   Log Likelihood                 -652.
135
Date:                Fri, 04 Oct 2024   AIC                            1320.
270
Time:                        13:04:10   BIC                            1345.
543
Sample:                    01-01-2010   HQIC                           1330.
522
                         - 07-01-2024
Covariance Type:                  opg
===========================================================================
===
                 coef    std err          z      P>|z|      [0.025      0.9
75]
---------------------------------------------------------------------------
---
ar.L1         -0.5737      0.100     -5.739      0.000      -0.770      -0.
378
ar.L2         -0.7718      0.079     -9.799      0.000      -0.926      -0.
617
ar.L3         -0.2194      0.100     -2.195      0.028      -0.415      -0.
024
ma.L1          0.6242      0.092      6.753      0.000       0.443       0.
805
ma.L2          0.9646      0.077     12.564      0.000       0.814       1.
115
ma.L3          0.3164      0.094      3.354      0.001       0.132       0.
501
ma.L4          0.5099      0.056      9.104      0.000       0.400       0.
620
sigma2       104.3253      6.228     16.751      0.000      92.119     116.
532
===========================================================================
========
Ljung-Box (L1) (Q):                   0.43   Jarque-Bera (JB):
168.45
Prob(Q):                              0.51   Prob(JB):
0.00
Heteroskedasticity (H):              64.14   Skew:
0.84
```

```
Prob(H) (two-sided):                  0.00   Kurtosis:
7.52
===========================================================================
========

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comp
lex-step).
```




```
MSE: 108.32661475707457
RMSE: 10.408007242362707
MAE: 6.431025581883379
MAPE: 5.913646149256075%
```






```
     Period 1    Period 2    Period 3    Period 4    Period 5    Period 6  \
0  411.905331  423.617515  429.128290  406.538058  412.426086  427.417902
1  410.628460  434.845781  426.738483  404.107106  416.671519  426.383835
2  404.847283  428.342928  429.224819  414.053609  412.226931  419.411844
3  409.494750  422.432571  436.146928  406.235524  409.418999  390.173170
4  421.024568  428.219336  417.968057  393.194445  409.209998  428.694773

     Period 7    Period 8    Period 9   Period 10  ...   Period 31  \
0  430.476714  404.980907  416.278142  427.440426  ...  430.666721
1  422.719645  420.742940  418.117213  423.474754  ...  445.923272
2  424.305722  406.675374  418.603663  435.362414  ...  410.153184
3  441.792376  396.169511  405.549817  416.153372  ...  412.824148
4  419.248448  407.370714  418.709094  423.194993  ...  418.170282

    Period 32   Period 33   Period 34   Period 35   Period 36   Period 37  \
0  426.668906  428.500530  418.478222  410.223294  414.540133  422.723830
1  418.272868  418.305970  418.920153  413.003976  434.461321  418.339802
2  422.436534  413.583257  418.114614  430.721440  412.443663  417.544422
3  402.561884  439.061624  440.804643  392.681362  442.195481  458.769876
4  419.100745  416.934026  413.017803  420.432386  442.195481  458.769876

    Period 38   Period 39   Period 40
0  420.020595  420.642022  428.211493
1  436.318796  431.877681  423.022442
2  410.200329  422.670719  427.546049
3  410.906463  432.121040  424.964650
4  410.906463  432.121040  424.964650

[5 rows x 40 columns]
```

How to Define the Best p-Value Threshold by Looking at the Distribution of Stock

Defining the best p-value threshold for ARIMA model selection by analyzing the distribution of stock data requires balancing statistical significance and the underlying characteristics of the stock's time series. Here's how you can approach it:

Steps to Define the Best p-Value Threshold:

1. **Analyze the Stock Price Distribution**:

    - **Look for Normality**: If the stock price distribution is approximately normal, smaller p-value thresholds (e.g., 0.01, 0.05) may work well because you're focusing on strict significance. However, stock prices often exhibit non-normality, volatility, and trends, meaning that you might need a more lenient threshold (e.g., 0.10 or higher) to capture important model features.
    - **Skewness and Kurtosis**: If the distribution is highly skewed or has fat tails, consider higher p-value thresholds (0.10 or 0.15) since some features might still be important but show higher p-values due to market anomalies.
2. **Plot the Distribution**:

    - Plot a histogram of the stock price returns (not prices) or use a kernel density estimate (KDE) to visualize the underlying distribution.
    - You can also plot QQ plots to compare the stock return distribution against a normal distribution.
3. **Consider Volatility and Outliers**:

    - If the stock has high volatility (common in stocks like TSLA), lower p-value thresholds might exclude too many relevant terms, leading to underfitting. In such cases, a slightly higher p-value (e.g., 0.1) might be more appropriate.
    - For stable stocks, stricter thresholds (0.05 or lower) might work better because significant patterns will emerge more clearly.
4. **Compare Model Performance with Different p-Values**:

    - Run your ARIMA model with different p-value thresholds (e.g., 0.01, 0.05, 0.10, 0.15) and compare the models' error metrics (e.g., AIC, RMSE, MAPE).
    - Look for a threshold that balances a good fit (lower AIC/BIC) without overfitting (keeping the number of terms reasonable).
5. **Cross-Validation**:

    - Use time-series cross-validation to evaluate model performance across different thresholds. The best p-value threshold would be the one that gives the lowest validation error.
6. **Evaluate Statistical Significance**:

    - **Lower p-values (e.g., 0.01)** indicate that the terms are strongly significant and help reduce the chance of including irrelevant terms. However, this could cause the model to miss important, but less significant, terms.
    - **Higher p-values (e.g., 0.10–0.15)** allow more flexibility but could increase the risk of overfitting by including noise.

In [ ]:

```

```