ARIMA Model and Box-Jenkins Methodology

The ARIMA (AutoRegressive Integrated Moving Average) model is a popular time series forecasting method that captures the relationship between an observation and a number of lagged observations, as well as the lagged forecast errors. The model is defined as:

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t$$

Where:

- Y_t is the actual value at time t.
- $\phi_1, \phi_2, \ldots, \phi_p$ are the autoregressive coefficients.
- $\theta_1, \theta_2, \ldots, \theta_q$ are the moving average coefficients.
- ϵ_t is the error term (white noise).
- p is the order of the autoregressive part.
- q is the order of the moving average part.
- d is the number of differencing required to make the series stationary.

The Box-Jenkins methodology involves the following steps to identify the best ARIMA model:

1. **Model Identification**: Determine the values of p, d, and q using tools like the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots.
2. **Parameter Estimation**: Estimate the coefficients ϕ and θ using methods like Maximum Likelihood Estimation (MLE).
3. **Model Checking**: Validate the model by analyzing residuals and ensuring they behave like white noise.
4. **Forecasting**: Use the fitted model to forecast future values.

Matrices and Equations

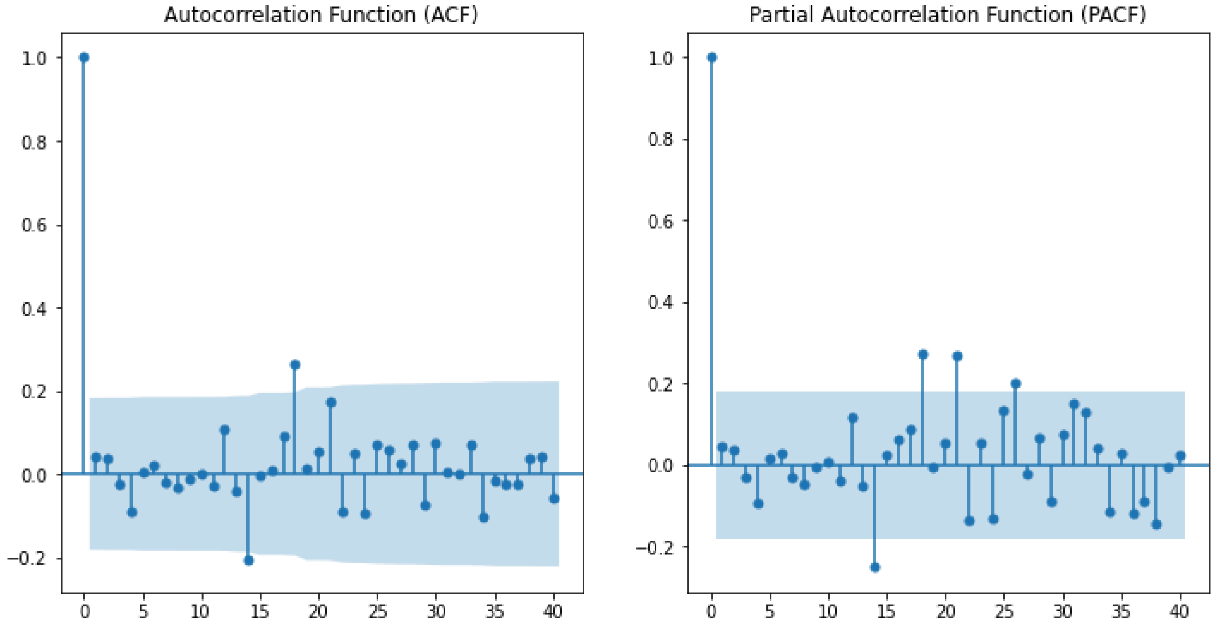
For AR(1) process:
$Y_t = \phi_1 Y_{t-1} + \epsilon_t$
Matrix form:

$$\begin{pmatrix} Y_t \\ Y_{t-1} \\ \vdots \\ Y_1 \end{pmatrix} = \begin{pmatrix} \phi_1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} Y_{t-1} \\ Y_{t-2} \\ \vdots \\ Y_0 \end{pmatrix} + \begin{pmatrix} \epsilon_t \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

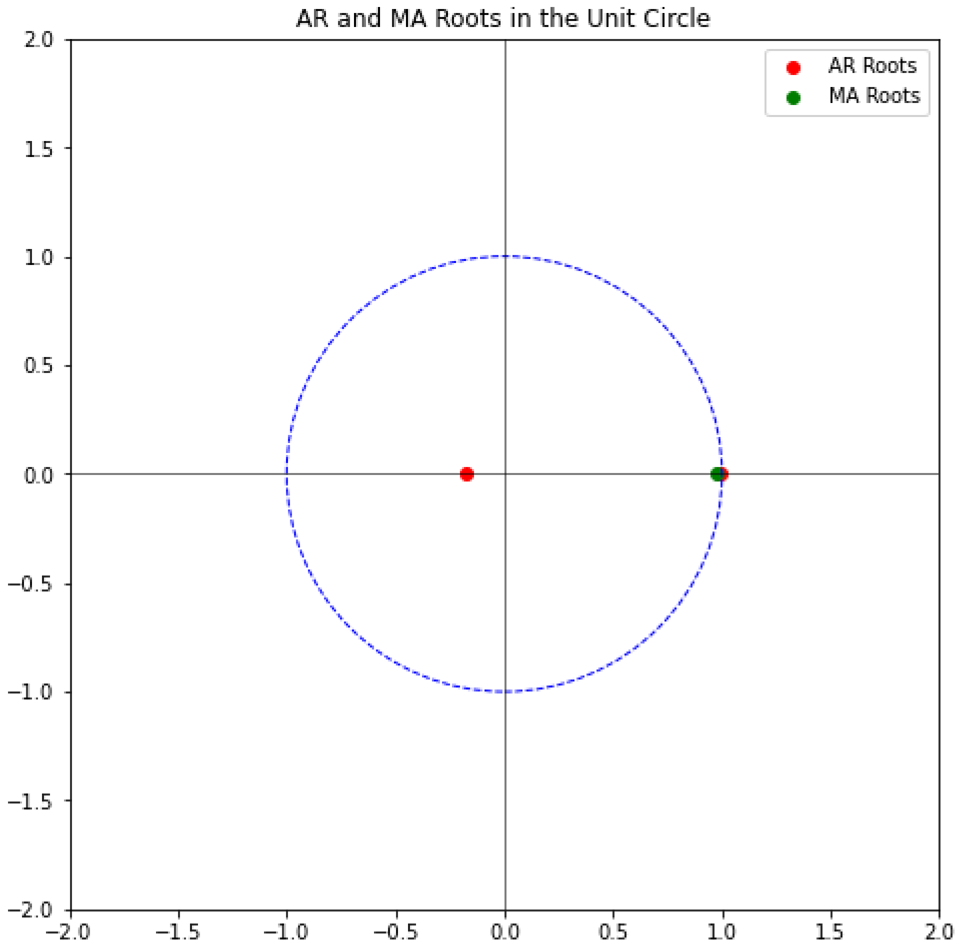
In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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

# Fetch the data from Yahoo Finance
ticker = 'MSFT'
data = yf.download(ticker, start='2010-01-01', end='2024-07-31', interval='1mo')

# Keep only the 'Close' column
data = data['Close']

# Differencing to make the series stationary
data_diff = data.diff().dropna()

# ACF and PACF plots
plt.figure(figsize=(12, 6))
plt.subplot(121)
plot_acf(data_diff, ax=plt.gca(), lags=40)
plt.title('Autocorrelation Function (ACF)')

plt.subplot(122)
plot_pacf(data_diff, ax=plt.gca(), lags=40)
plt.title('Partial Autocorrelation Function (PACF)')
plt.show()

# Model Identification: Using AIC/BIC to select the best ARIMA model
best_aic = np.inf
best_order = None
best_model = None

for p in range(5):
    for d in range(2):
        for q in range(5):
            try:
                model = ARIMA(data, order=(p, d, q)).fit()
                if model.aic < best_aic:
                    best_aic = model.aic
                    best_order = (p, d, q)
                    best_model = model
            except Exception as e:
                print(f"ARIMA({p},{d},{q}) failed to fit: {e}")
                continue

if best_model is not None:
    print(f'Best ARIMA model: {best_order} with AIC: {best_aic}')
else:
    print("No suitable ARIMA model found.")

# Display statistical results of the best ARIMA model
if best_model is not None:
    print(best_model.summary())

# Plot AR and MA roots in the unit circle
if best_model is not None and best_order[0] > 0:  # Only plot if AR or MA parameters
    ar_params = np.r_[1, -best_model.arparams] if best_order[0] > 0 else [1]
    ma_params = np.r_[1, best_model.maparams] if best_order[2] > 0 else [1]

    ar_roots = np.roots(ar_params)
    ma_roots = np.roots(ma_params)

    plt.figure(figsize=(8, 8))
    unit_circle = plt.Circle((0, 0), 1, color='blue', fill=False, linestyle='--')
    plt.gca().add_patch(unit_circle)

    plt.scatter(ar_roots.real, ar_roots.imag, color='red', label='AR Roots')
    plt.scatter(ma_roots.real, ma_roots.imag, color='green', label='MA Roots')

    plt.axhline(0, color='black', linewidth=0.5)
    plt.axvline(0, color='black', linewidth=0.5)
    plt.xlim(-2, 2)
    plt.ylim(-2, 2)
    plt.gca().set_aspect('equal', adjustable='box')

    plt.title('AR and MA Roots in the Unit Circle')
    plt.legend()
    plt.show()

# Unit Root Graph (ADF Test)
adf_test = sm.tsa.adfuller(data_diff)
print(f'ADF Statistic: {adf_test[0]}')
print(f'p-value: {adf_test[1]}')

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

    # Actual vs Forecast
    plt.figure(figsize=(10, 6))
    plt.plot(data, label='Actual')
    plt.plot(best_model.fittedvalues, color='red', label='Fitted')
    plt.legend()
    plt.title('Actual vs Fitted')
    plt.show()

    # Forecasting until the end of 2025
    forecast_steps = 17  # Number of months from August 2024 to December 2025
    forecast = best_model.get_forecast(steps=forecast_steps)
    forecast_index = pd.date_range(start=data.index[-1], periods=forecast_steps+1, f

    # Plot Actual vs Forecast
    plt.figure(figsize=(10, 6))
    plt.plot(data, label='Actual')
    plt.plot(best_model.fittedvalues, color='red', label='Fitted')
    plt.plot(forecast_index, forecast.predicted_mean, color='green', label='Forecast
    plt.fill_between(forecast_index, forecast.conf_int()[:, 0], forecast.conf_int()[
    plt.legend()
    plt.title('Actual vs Fitted and Forecasted Values')
    plt.show()

else:
    print("Skipping residual and forecast plots since no model was fitted.")
```

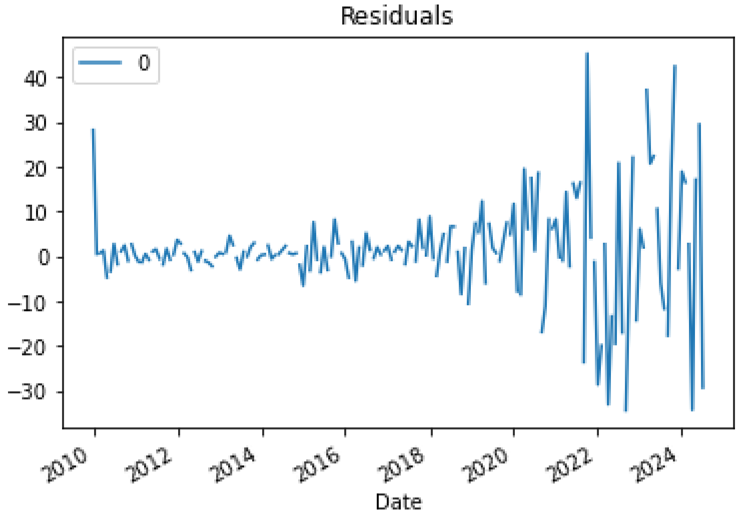
```
[*********************100%***********************]  1 of 1 completed
```

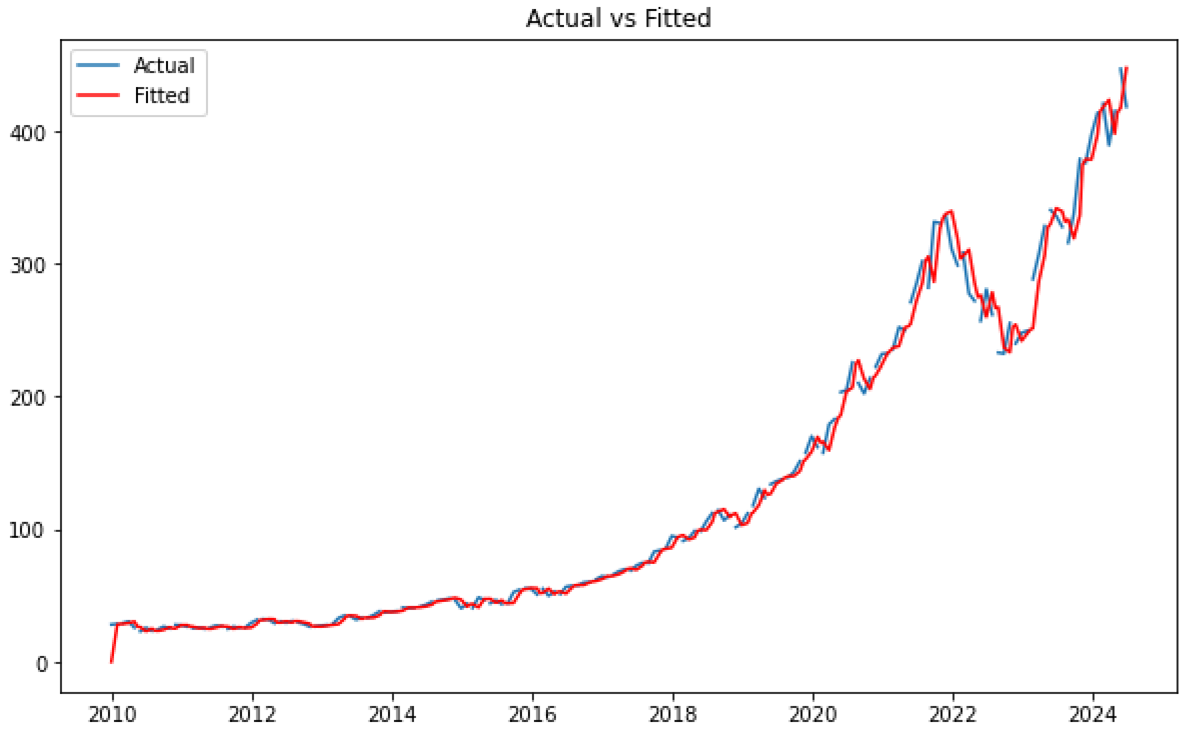


```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978:
UserWarning: Non-invertible starting MA parameters found. Using zeros as starting par
ameters.
  warn('Non-invertible starting MA parameters found.'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\base\model.py:566: Convergence
Warning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:581: Val
ueWarning: A date index has been provided, but it has no associated frequency informa
tion and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\base\model.py:566: Convergence
Warning: Maximum Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
Best ARIMA model: (2, 1, 1) with AIC: 1346.4776082955477
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  233
Model:                 ARIMA(2, 1, 1)   Log Likelihood                -669.239
Date:                Thu, 08 Aug 2024   AIC                           1346.478
Time:                        21:59:27   BIC                           1360.265
Sample:                             0   HQIC                          1352.038
                                - 233
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8181      0.054     15.068      0.000       0.712       0.925
ar.L2          0.1778      0.053      3.358      0.001       0.074       0.282
ma.L1         -0.9783      0.030    -32.550      0.000      -1.037      -0.919
sigma2       106.2550      6.499     16.349      0.000      93.517     118.993
===================================================================================
Ljung-Box (L1) (Q):                   0.09   Jarque-Bera (JB):               488.43
Prob(Q):                              0.76   Prob(JB):                         0.00
Heteroskedasticity (H):             106.62   Skew:                             0.42
Prob(H) (two-sided):                  0.00   Kurtosis:                        10.06
===================================================================================

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



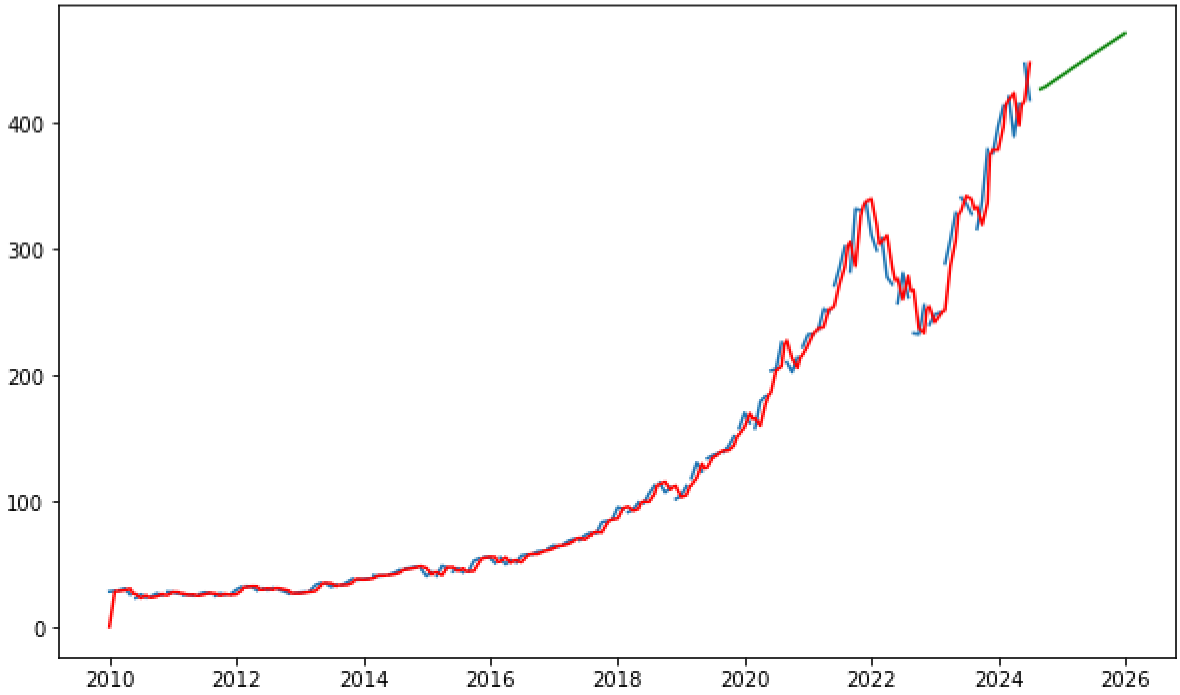
```
ADF Statistic: -9.79453953047507
p-value: 6.224781218168595e-17
```





```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:376: Val
ueWarning: No supported index is available. Prediction results will be given with an
integer index beginning at `start`.
  warnings.warn('No supported index is available.'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_9456/464922073.py in <module>
    109     plt.plot(best_model.fittedvalues, color='red', label='Fitted')
    110     plt.plot(forecast_index, forecast.predicted_mean, color='green', label='F
orecast')
--> 111     plt.fill_between(forecast_index, forecast.conf_int()[:, 0], forecast.conf
_int()[:, 1], color='green', alpha=0.2)
    112     plt.legend()
    113     plt.title('Actual vs Fitted and Forecasted Values')

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self,
key)
   3456             if self.columns.nlevels > 1:
   3457                 return self._getitem_multilevel(key)
-> 3458             indexer = self.columns.get_loc(key)
   3459             if is_integer(indexer):
   3460                 indexer = [indexer]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in get_loc(sel
f, key, method, tolerance)
   3359             casted_key = self._maybe_cast_indexer(key)
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
   3363                 raise KeyError(key) from err

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.ind
ex.IndexEngine.get_loc()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.ind
ex.IndexEngine.get_loc()

TypeError: '(slice(None, None, None), 0)' is an invalid key
```



```
In [ ]:
```