```python
import pandas as pd
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats
import statistics
import statsmodels.api as sm
from scipy.stats import skew, skewtest, norm
from scipy.stats import kurtosis, skewnorm
import scipy.stats as stats
from scipy.stats import boxcox
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics import tsaplots
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
import seaborn as sns
import yfinance as yf
import warnings
warnings.simplefilter("ignore")
plt.style.use('ggplot')
```

Analysis for Skewness

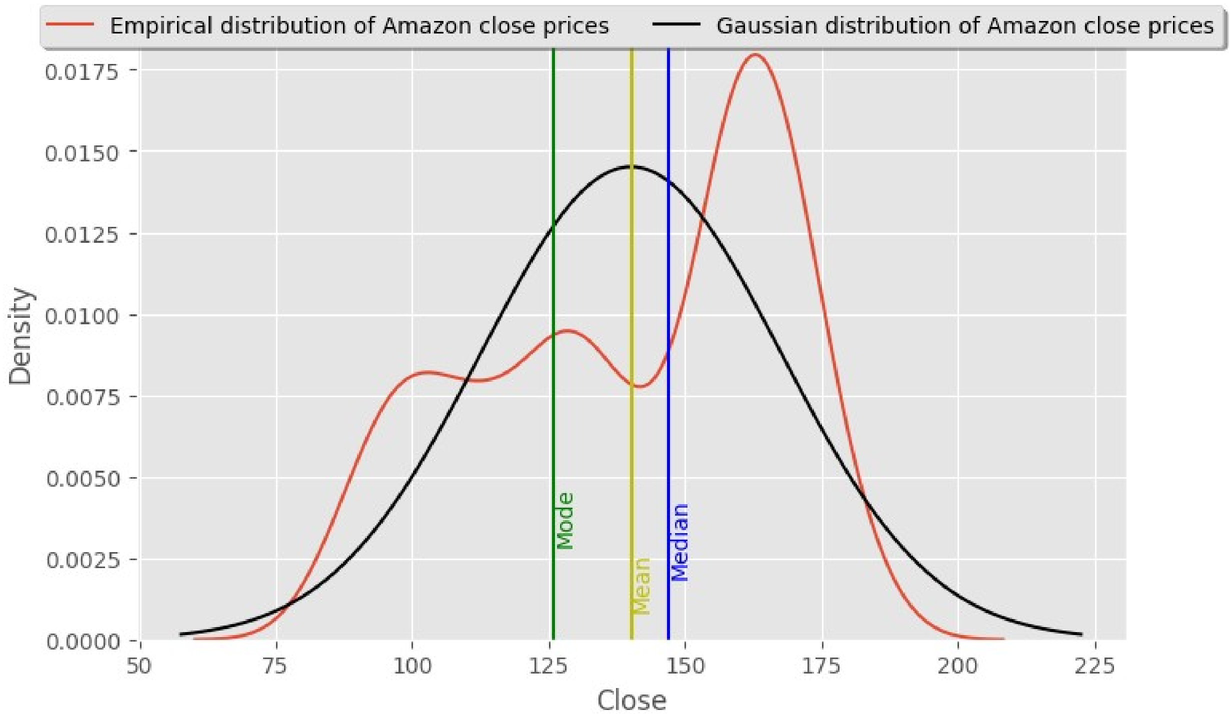
Skewness is a statistical metric that indicates the symmetry of a distribution. One frequently employed method for quantifying skewness is the Fisher-Pearson Standardized Moment Coefficient. Skewness manifests in two forms: left-skewed and right-skewed. A distribution with perfect symmetry exhibits a skewness of zero. When a distribution is positively skewed (skewed to the right), it indicates that the right-side tail is longer or more dispersed compared to the left side. Conversely, if a distribution is negatively skewed (skewed to the left), it signifies that the left-side tail is longer or more dispersed than the right side.

We can examine the symmetry of the distribution by utilizing the closing prices of META Platforms over the past few years as examples. It involves plotting the distribution of META close prices against a Gaussian or Normal distribution for comparison.

```python
amzn = yf.Ticker("AMZN")
amzn_close = amzn.history(start="2020-09-01",  end="2023-10-31")
["Close"]
amzn_close.head()
```

```
Date
2020-09-01 00:00:00-04:00    174.955994
2020-09-02 00:00:00-04:00    176.572495
2020-09-03 00:00:00-04:00    168.399994
2020-09-04 00:00:00-04:00    164.731003
2020-09-08 00:00:00-04:00    157.492004
Name: Close, dtype: float64
```

```
mean = amzn_close.mean()
std = amzn_close.std()
x = np.linspace(mean - 3*std, mean + 3*std, 100)

plt.figure(figsize=(8, 5))
sns.distplot(amzn_close, hist=False, label="Empirical distribution of
Amazon close prices")

plt.plot(x, scipy.stats.norm.pdf(x, mean, std), label="Gaussian
distribution of Amazon close prices", color='k')

# Add Mean
plt.axvline(x=statistics.mean(amzn_close), color='y')
plt.text(statistics.mean(amzn_close), 0.001, "Mean", rotation=90,
color='y')

# Add Median
plt.axvline(x=statistics.median(amzn_close), color='b')
plt.text(statistics.median(amzn_close), 0.002, "Median", rotation=90,
color='b')

# Add mode
plt.axvline(x=statistics.mode(amzn_close), color='g')
plt.text(statistics.mode(amzn_close), 0.003, "Mode", rotation=90,
color='g')

plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.05), ncol=2,
fancybox=True, shadow=True)
plt.show()
```



Based on the depicted plots, normal distributions exhibit symmetrical patterns with the peak situated at the center of the distribution. In contrast, the AMZN close price distribution displays a peak located in the right region, indicating a left-skewed distribution.

To further substantiate the skewness of the distribution, various mathematical equations can be employed, such as the Fisher-Pearson Standardized Moment Coefficient, as outlined in the provided definition. These calculations, which incorporate the mean, median, and standard deviation of the dataset, offer a numerical representation of the distribution's skewness. A symmetric distribution is indicated by a coefficient value of 0. A negative value signifies a left-skewed distribution, while a positive value denotes a right-skewed distribution. The coefficient compares the dataset with a normal distribution, and a larger coefficient value suggests a greater deviation from normality. In our case, the skewness of the AMZN close price is calculated as -0.3886, confirming a left-skewed distribution.

```
mean = statistics.mean(amzn_close)
median = statistics.median(amzn_close)
mode = statistics.mode(amzn_close)
print('\n mean: ', mean, '\n median: ', median, '\n mode: ',mode)


 mean:  140.09588769692272
 median:  147.09200286865234
 mode:  125.9800033569336

amzn_skew = skew(amzn_close, axis=0, bias=True)
print('skewness: ', amzn_skew)

skewness:  -0.3886130280877852
```

The negative value of skew indicates a left-skew distribution.

Some notable consequences of skewness in a distribution include:

1. Misleading to depend on Central Tendency: Skewed distributions can introduce a discrepancy between the mean and the median. In such instances, relying solely on the mean as a measure of central tendency can be misleading, as the skewness pulls the mean in the direction of the longer tail, affecting the perceived center of the distribution.

2. Error in Forecasting: Forecasting models often assume normally distributed residuals. If the residuals exhibit skewness, the predictive performance of the model may be compromised. Skewness can result in underestimation or overestimation of future values, influencing decision-making and planning based on flawed forecasts. Understanding and addressing skewness in data is crucial for obtaining accurate insights and making informed decisions in various analytical and modeling contexts.

In the case of a skewed distribution, various transformation methods can be applied, commonly including log and Box-Cox transformations. Specifically, the log transformation is useful for compressing large value ranges, enhancing pattern visibility, and facilitating data visualization and analysis.

```
amzn_close.isna().sum()
```

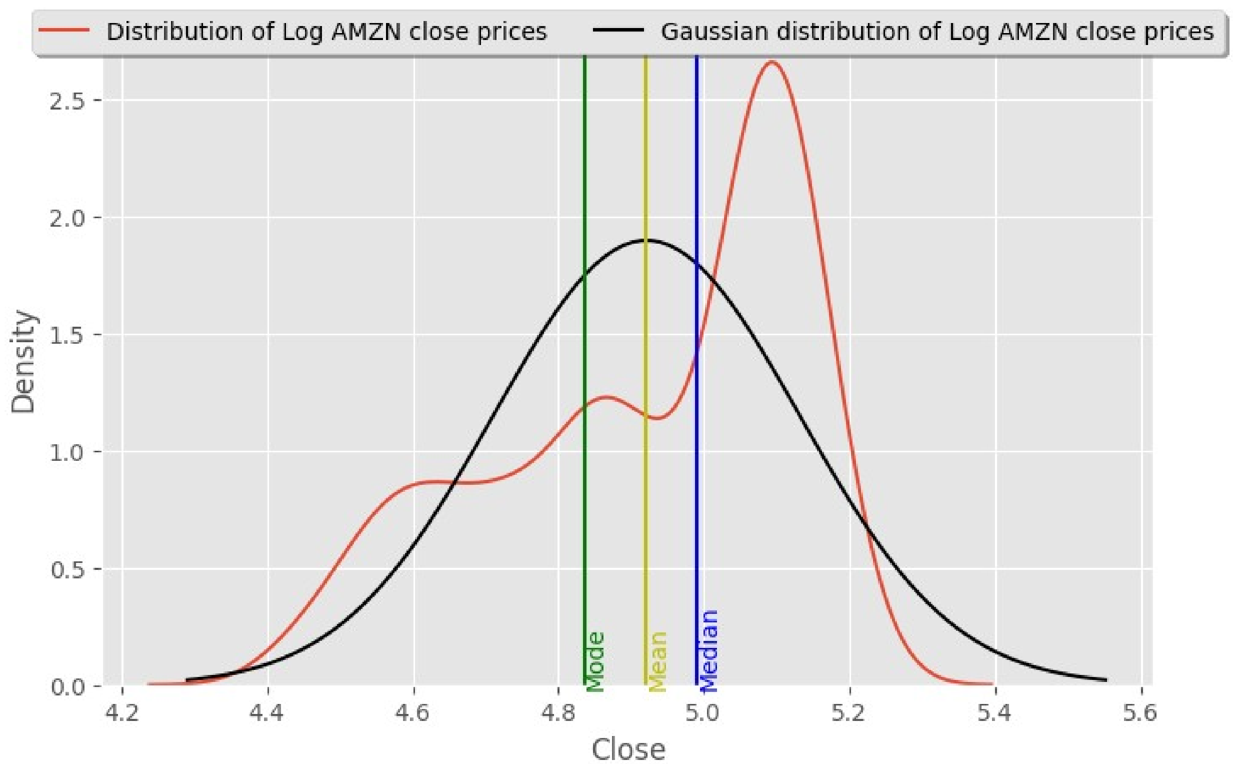
```
0
```

```
log_amzn_close = np.log(meta_close)
log_amzn_close.head()
```

```
Date
2020-09-01 00:00:00-04:00    5.164534
2020-09-02 00:00:00-04:00    5.173732
2020-09-03 00:00:00-04:00    5.126342
2020-09-04 00:00:00-04:00    5.104314
2020-09-08 00:00:00-04:00    5.059375
Name: Close, dtype: float64
```

```
mean = log_amzn_close.mean()
std = log_amzn_close.std()
x = np.linspace(mean - 3*std, mean + 3*std, 100)

plt.figure(figsize=(8, 5))
sns.distplot(log_amzn_close, hist=False, label="Distribution of Log AMZN close prices")

plt.plot(x, scipy.stats.norm.pdf(x, mean, std), label="Gaussian distribution of Log AMZN close prices", color='k')

# Add Mean
plt.axvline(x=statistics.mean(log_amzn_close), color='y')
plt.text(statistics.mean(log_amzn_close), 0.001, "Mean", rotation=90,
```

```
color='y')

# Add Median
plt.axvline(x=statistics.median(log_amzn_close), color='b')
plt.text(statistics.median(log_amzn_close), 0.002, "Median",
rotation=90, color='b')

# Add mode
plt.axvline(x=statistics.mode(log_amzn_close), color='g')
plt.text(statistics.mode(log_amzn_close), 0.003, "Mode", rotation=90,
color='g')

plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.05), ncol=2,
fancybox=True, shadow=True)
plt.show()
```




```
log_skew = skew(log_amzn_close, axis=0, bias=True)
print('before log transform skewness: ', amzn_skew)
print('after log transform skewness: ', log_skew)

before log transform skewness:  -0.3886130280877852
after log transform skewness:  -0.6249954044570926
```

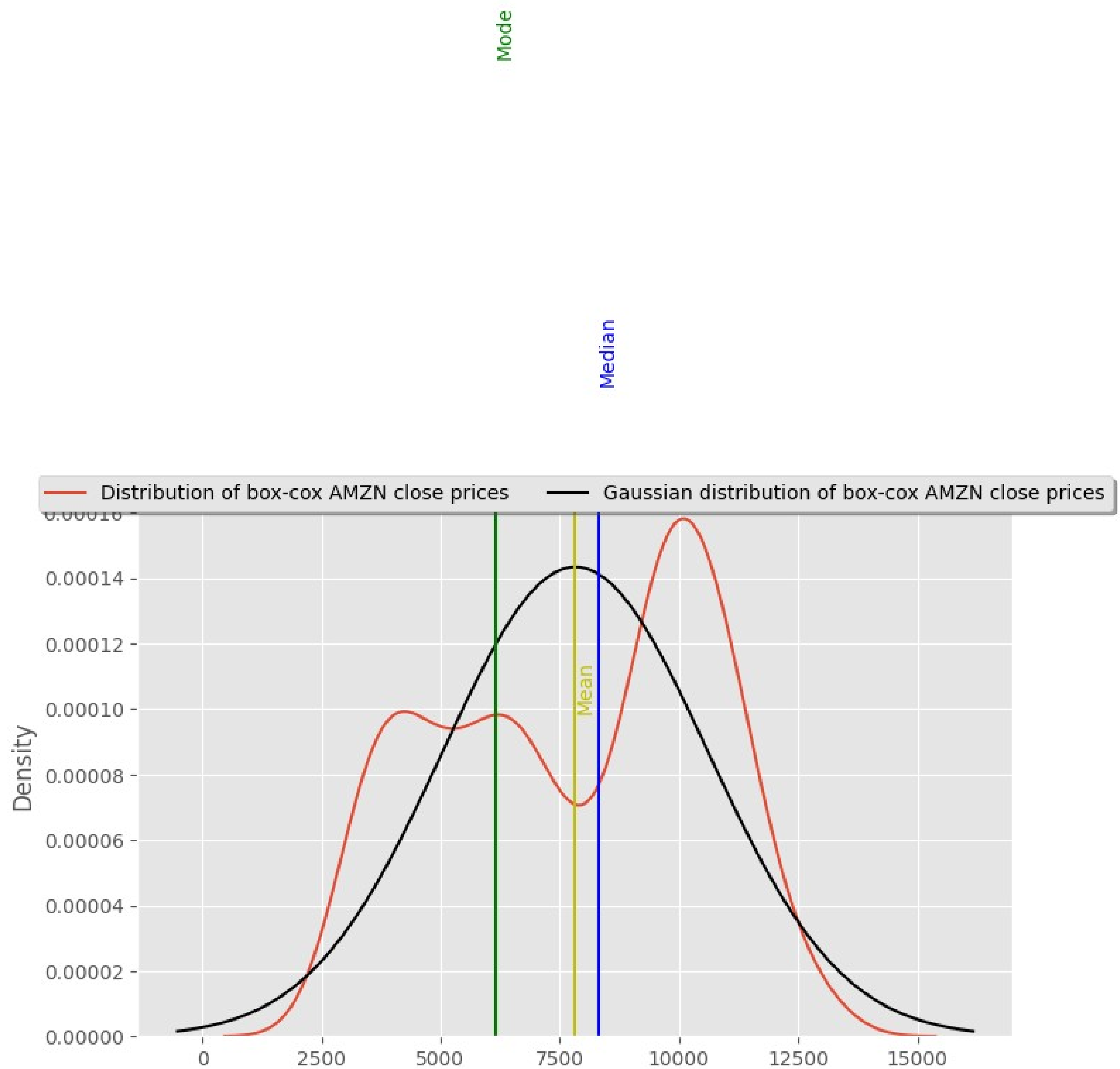
In the case of a skewed distribution, various transformation methods can be applied, commonly including log and Box-Cox transformations. Specifically, the log transformation is useful for compressing large value ranges, enhancing pattern visibility, and facilitating data visualization and analysis.

In our analysis, both transformation methods were tested. Surprisingly, the log transformation exacerbated the skewness issue. Consequently, we present only the successful results achieved through the Box-Cox transformation. The Box-Cox method is a variable transformation technique designed to convert a non-normally distributed variable into one that follows a normal distribution.

```
bc_amzn_close, bc_lambda = boxcox(amzn_close)
print('best lambda parameter in box-cox method: ', bc_lambda)
```

```
best lambda parameter in box-cox method:  1.9412341671853788
```

```
mean = bc_amzn_close.mean()
std = bc_amzn_close.std()
x = np.linspace(mean - 3*std, mean + 3*std, 100)

plt.figure(figsize=(8, 5))
sns.distplot(bc_amzn_close, hist=False, label="Distribution of box-cox
AMZN close prices")

plt.plot(x, scipy.stats.norm.pdf(x, mean, std), label="Gaussian
distribution of box-cox AMZN close prices", color='k')

# Add Mean
plt.axvline(x=statistics.mean(bc_amzn_close), color='y')
plt.text(statistics.mean(bc_amzn_close), 0.0001, "Mean", rotation=90,
color='y')

# Add Median
plt.axvline(x=statistics.median(bc_amzn_close), color='b')
plt.text(statistics.median(bc_amzn_close), 0.0002, "Median",
rotation=90, color='b')

# Add mode
plt.axvline(x=statistics.mode(bc_amzn_close), color='g')
plt.text(statistics.mode(bc_amzn_close), 0.0003, "Mode", rotation=90,
color='g')

plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.05), ncol=2,
fancybox=True, shadow=True)
plt.show()
```




```
print("Skew before box cox Transformation: %f" % skew(amzn_close))
print("Skew after box cox Transformation: %f" % skew(bc_amzn_close))

Skew before box cox Transformation: -0.388613
Skew after box cox Transformation: -0.183657
```

Indeed, the Box-Cox transformation has proven effective in mitigating the skewness issue of the META stock close price. The skewness value has decreased notably, shifting from -0.388613 to -0.183657. As a result of this transformation, the center of the peak in the distribution has moved towards the center, and the distribution now exhibits a slight left-skew. This adjustment reflects a more symmetrical and normalized distribution, addressing the initial skewness concern.

Analysis for Non-Stationarity

Non-stationarity in a time series implies that the mean, variance, and autocorrelation structure of the series undergo changes over time. In simpler terms, the characteristics of the time series are not consistent across different time points. Conversely, a stationary time series maintains constant statistical properties, including mean and variance, over time.

We can analyse share prices and volumes of Apple (AAPL) over the years and can see that while the volumnes are stationary, the close prices are not. Viewing the graphs itself provides a clue to this behavior

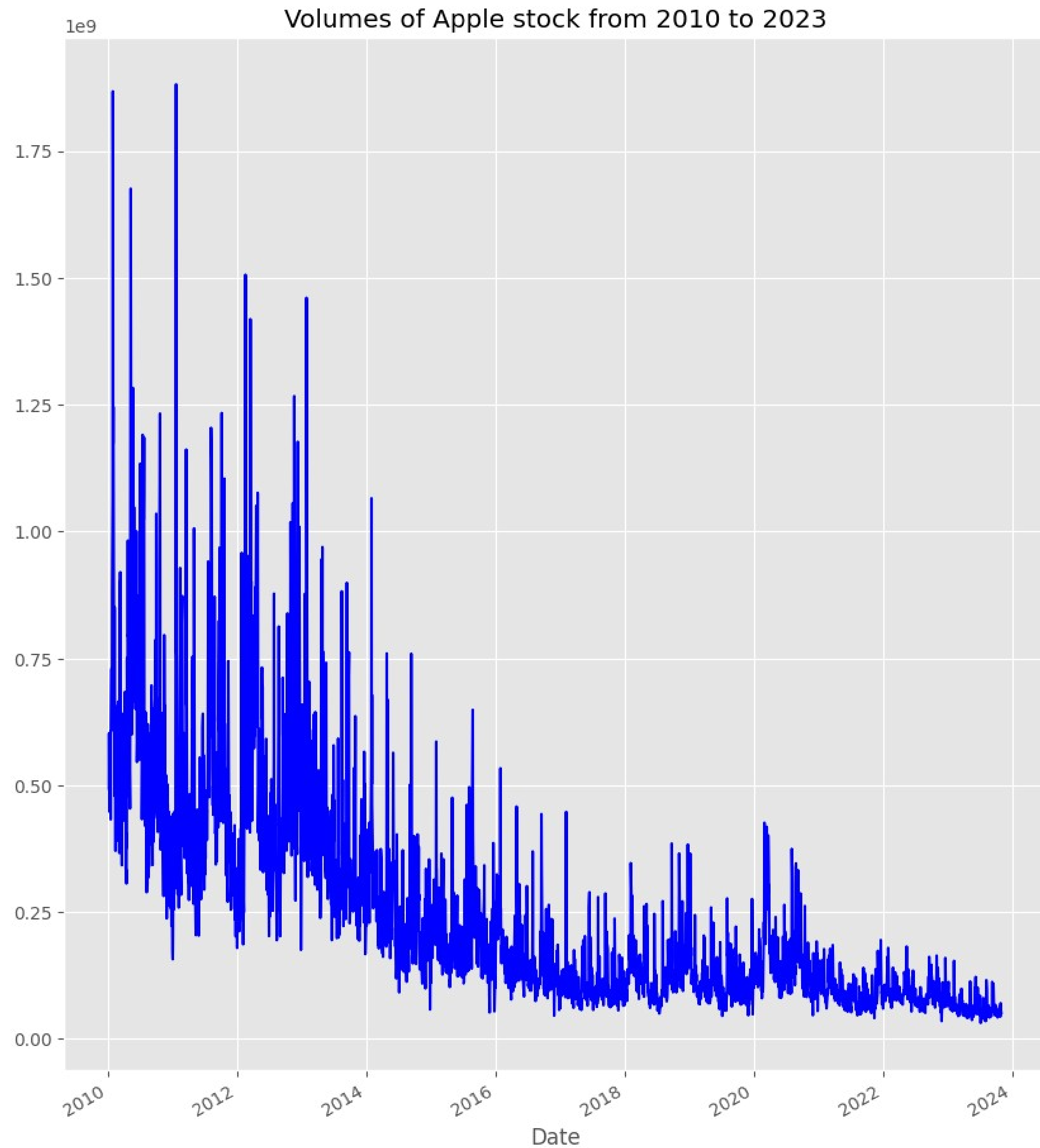
```
start_date = dt.date(2010, 1, 1)
end_date = dt.date(2023, 10, 31)
AAPL_data = yf.download(tickers = "AAPL" ,start = start_date, end =
end_date)
AAPL_data.head()
```

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

```
                Open      High       Low     Close  Adj Close
Volume
Date

2010-01-04  7.622500  7.660714  7.585000  7.643214   6.478998
493729600
2010-01-05  7.664286  7.699643  7.616071  7.656429   6.490201
601904800
2010-01-06  7.656429  7.686786  7.526786  7.534643   6.386965
552160000
2010-01-07  7.562500  7.571429  7.466071  7.520714   6.375157
477131200
2010-01-08  7.510714  7.571429  7.466429  7.570714   6.417540
447610800
```

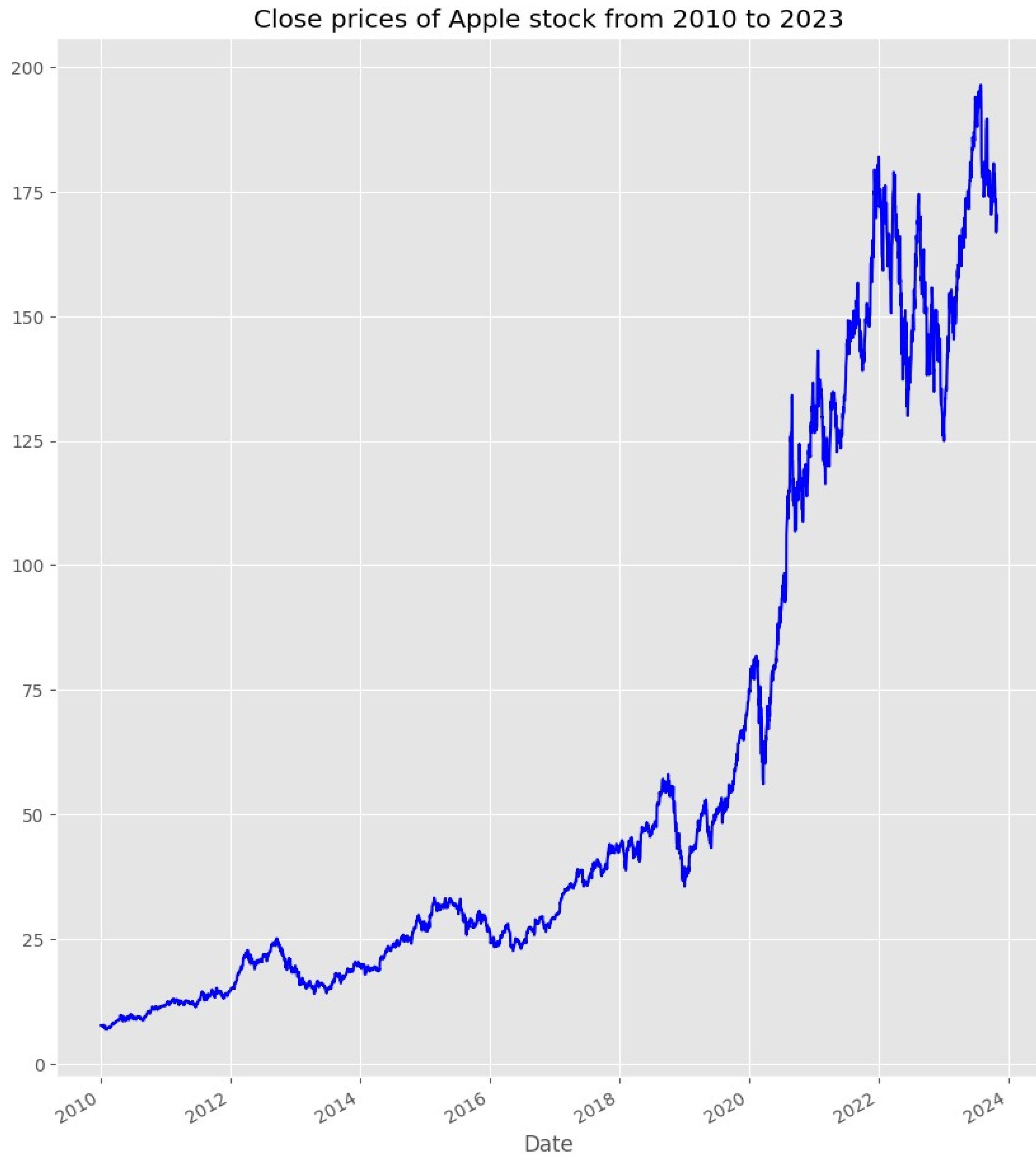
First, let's consider the share volumes and see how the trend can signal stationarity.

```
AAPL_data["Volume"].plot(subplots=True, figsize=(10,12), color="blue")
plt.title('Volumes of Apple stock from 2010 to 2023')
plt.show()
```



Next, we consider the share prices and see how these show non-stationarity. We can clearly see a trend.

```
AAPL_data["Close"].plot(subplots=True, figsize=(10,12), color="blue")
plt.title('Close prices of Apple stock from 2010 to 2023')
plt.show()
```



The Augmented Dickey Fuller test (ADF Test) is a widely used method to ascertain the stationarity of a given time series. The obtained p-value from this test is compared to a significance level, typically set at 0.05. To reject the null hypothesis that the series is non-stationary, the p-value should be less than this threshold. In our case, p value is lower than 0.05 and hence the volumes are stationary.

```
adf = adfuller(AAPL_data["Volume"])
print("p-value of volumes of AAPL shares: {}".format(float(adf[1])))
```

```
p-value of volumes of AAPL shares: 0.02249274662262558
```

The share prices are however non-stationary as the p value is higher than 0.05.

```
adf = adfuller(AAPL_data["Close"])
print("p-value of close price of Apple shares:
{}".format(float(adf[1])))
```

```
p-value of close price of Apple shares: 0.977691197692373
```

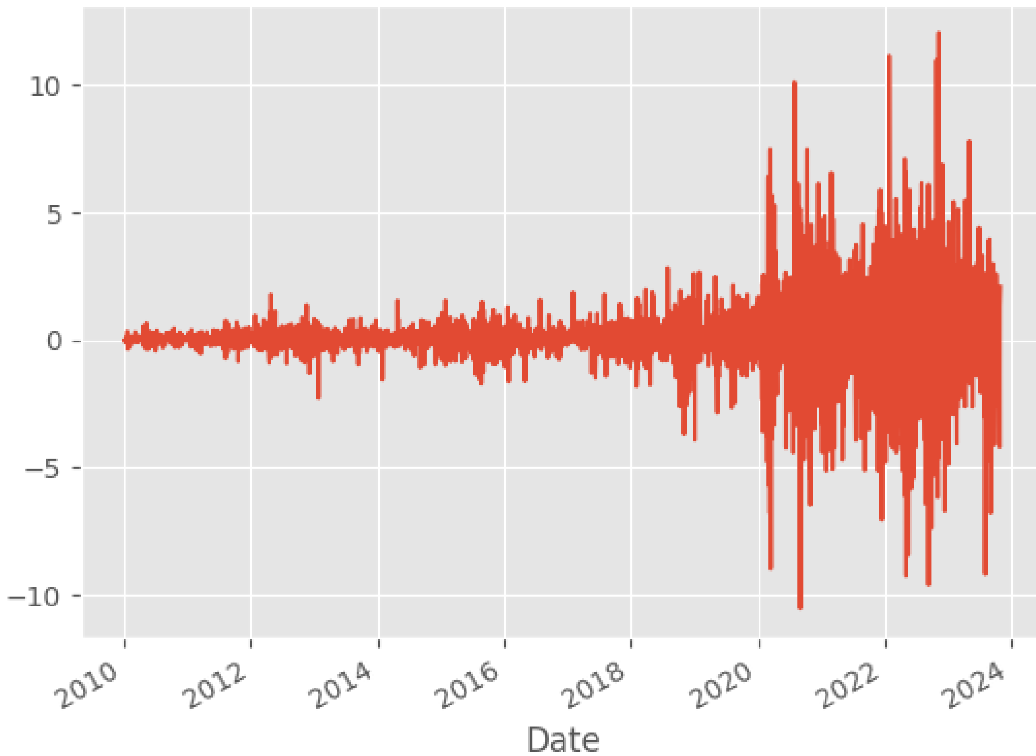
A frequently employed method to deal with non-stationary time series involves differencing the data to attain stationarity. In our analysis, we address the non-stationarity in our data by taking the difference between the variable and its lagged version. This approach has proven effective in rendering the trend of our variable stationary in our case.

```
AAPL_data["diff_Close"] = AAPL_data["Close"].diff().fillna(0)
adf = adfuller(AAPL_data["diff_Close"])
print("p-value of lagged difference of Apple close prices:
{}".format(float(adf[1])))
```

```
p-value of lagged difference of Apple close prices:
4.607571052044991e-25
```

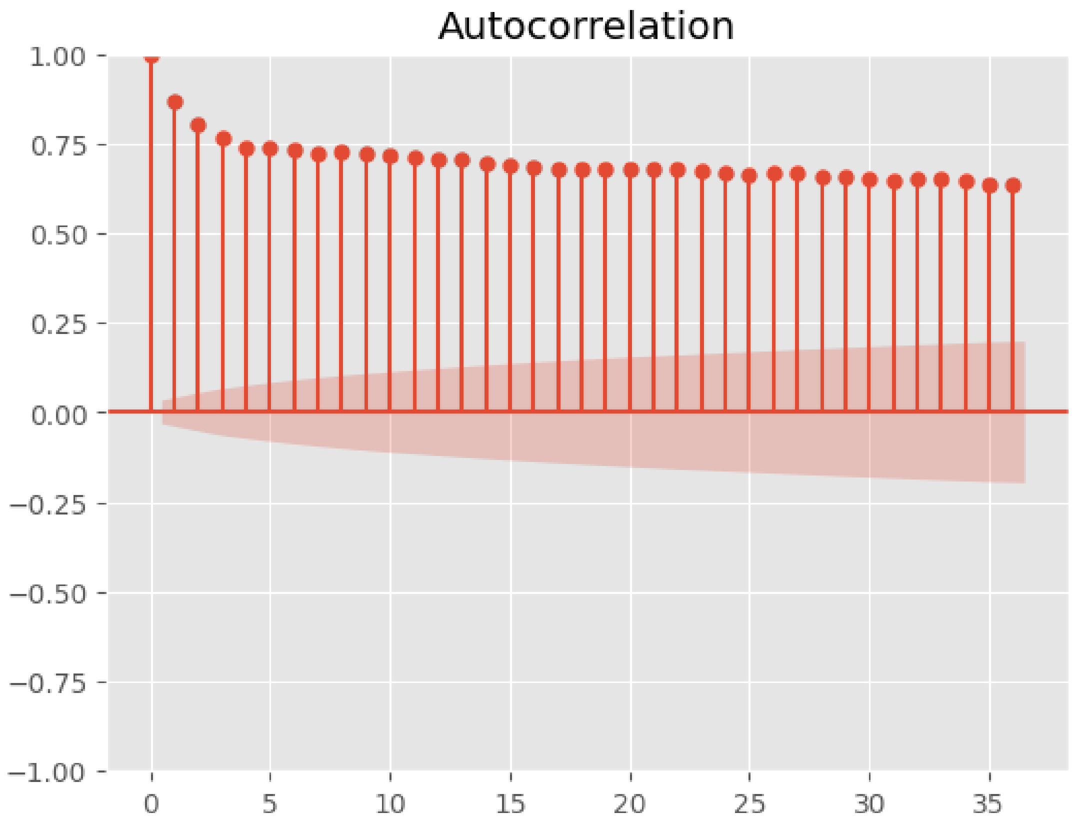
We can also confirm this from the below graph

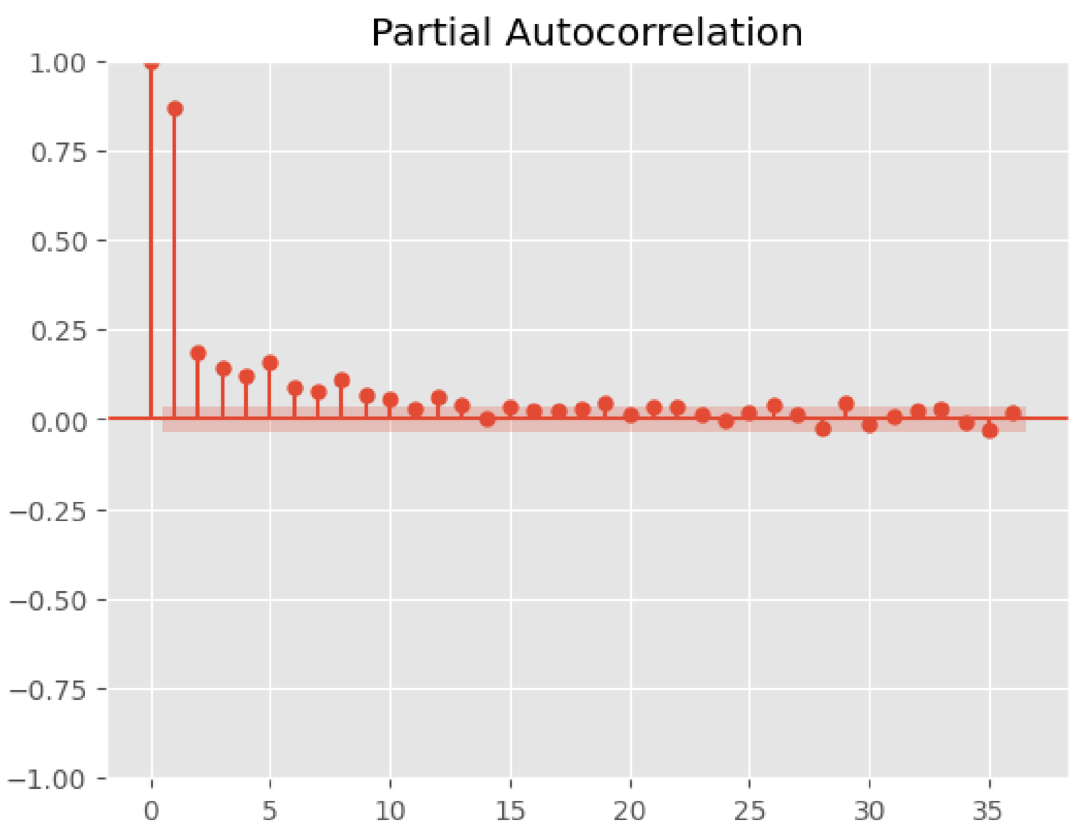
```
AAPL_data["diff_Close"].plot();
```



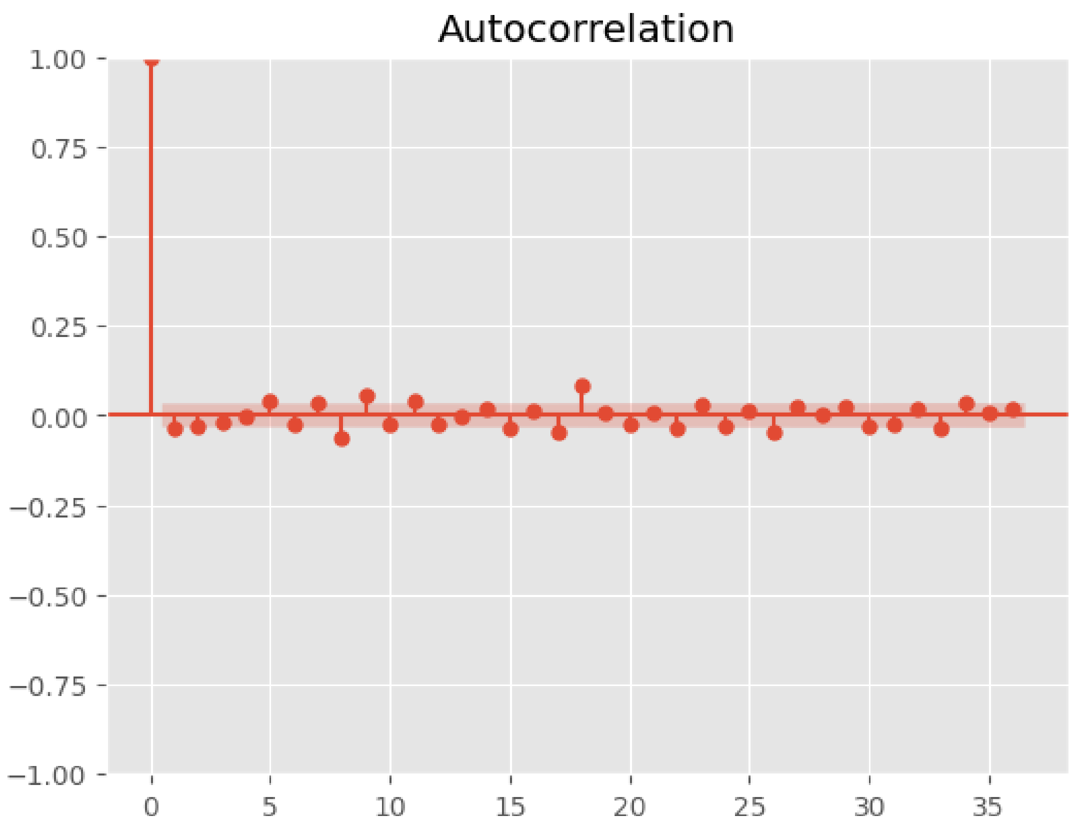
We can also review autocorrelation and partial autocorrelation plots of both volumes and lagged difference of close prices of Apple shares for further idea.

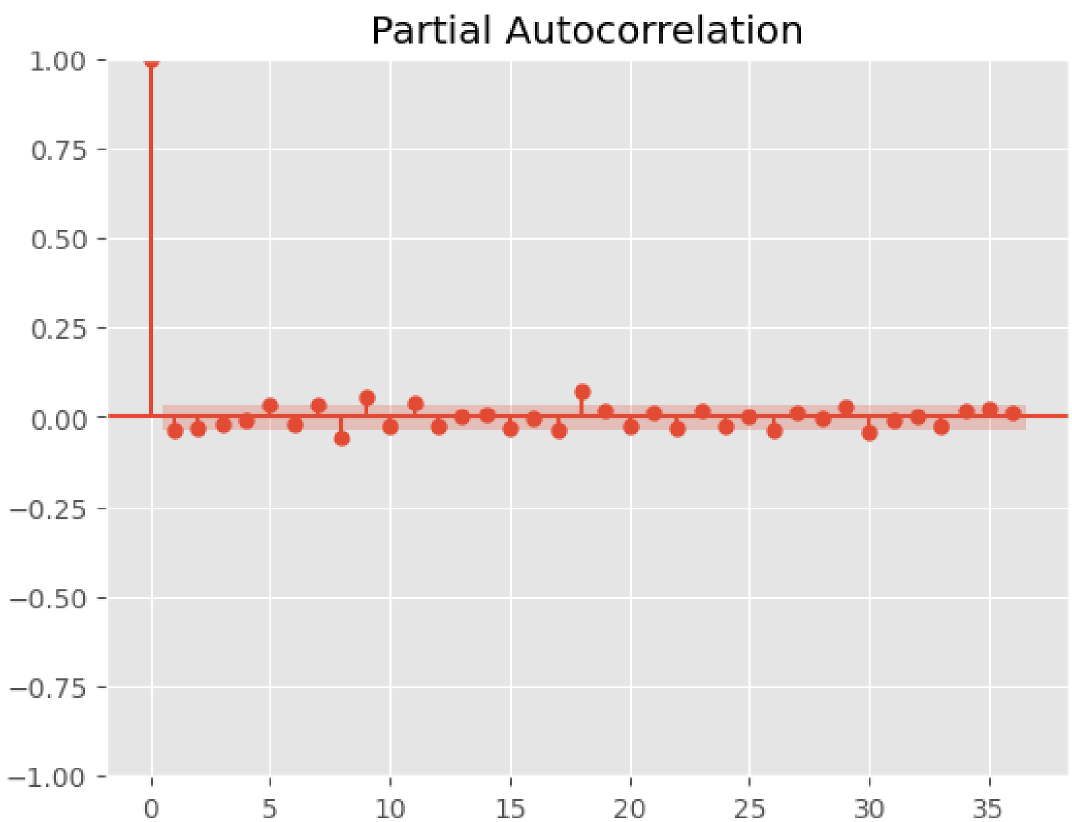
```
plot_acf(AAPL_data["Volume"]);
plot_pacf(AAPL_data["Volume"]);
```


Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35


Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35

```
plot_acf(AAPL_data["diff_Close"]);
plot_pacf(AAPL_data["diff_Close"]);
```





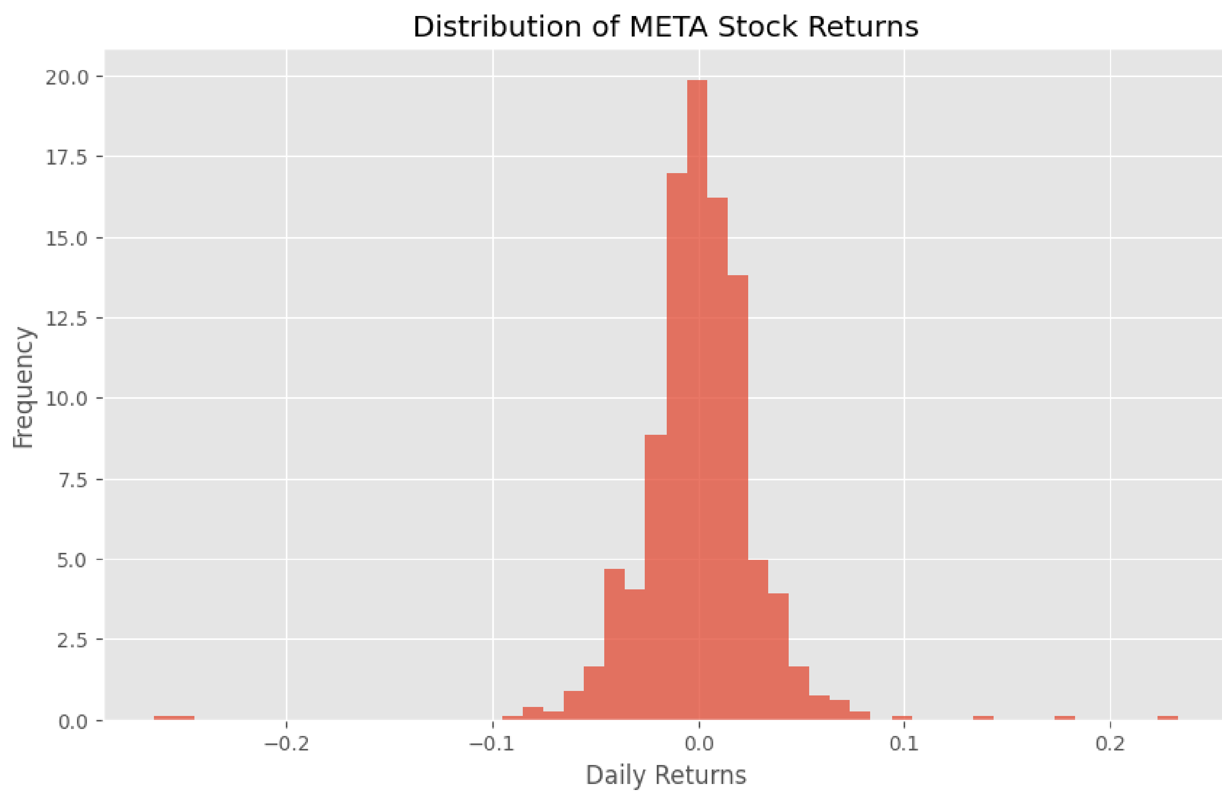
Analysis for Kurtosis

```
import yfinance as yf
from scipy.stats import jarque_bera
import pandas as pd

ticker = "META"
start_date = "2020-09-01"
end_date = "2023-10-31"

stock_data = yf.download(ticker, start=start_date, end=end_date)

stock_data['Daily_Return'] = stock_data['Adj
Close'].pct_change().dropna()

returns = stock_data['Daily_Return'].dropna()

statistic, p_value = jarque_bera(returns)

print(f"Jarque-Bera Test Statistic: {statistic}")
print(f"P-value: {p_value}")

alpha = 0.05
```

```
if p_value < alpha:
    print("The null hypothesis is rejected. The data does not follow a
normal distribution.")
else:
    print("Fail to reject the null hypothesis. The data follows a
normal distribution.")
```

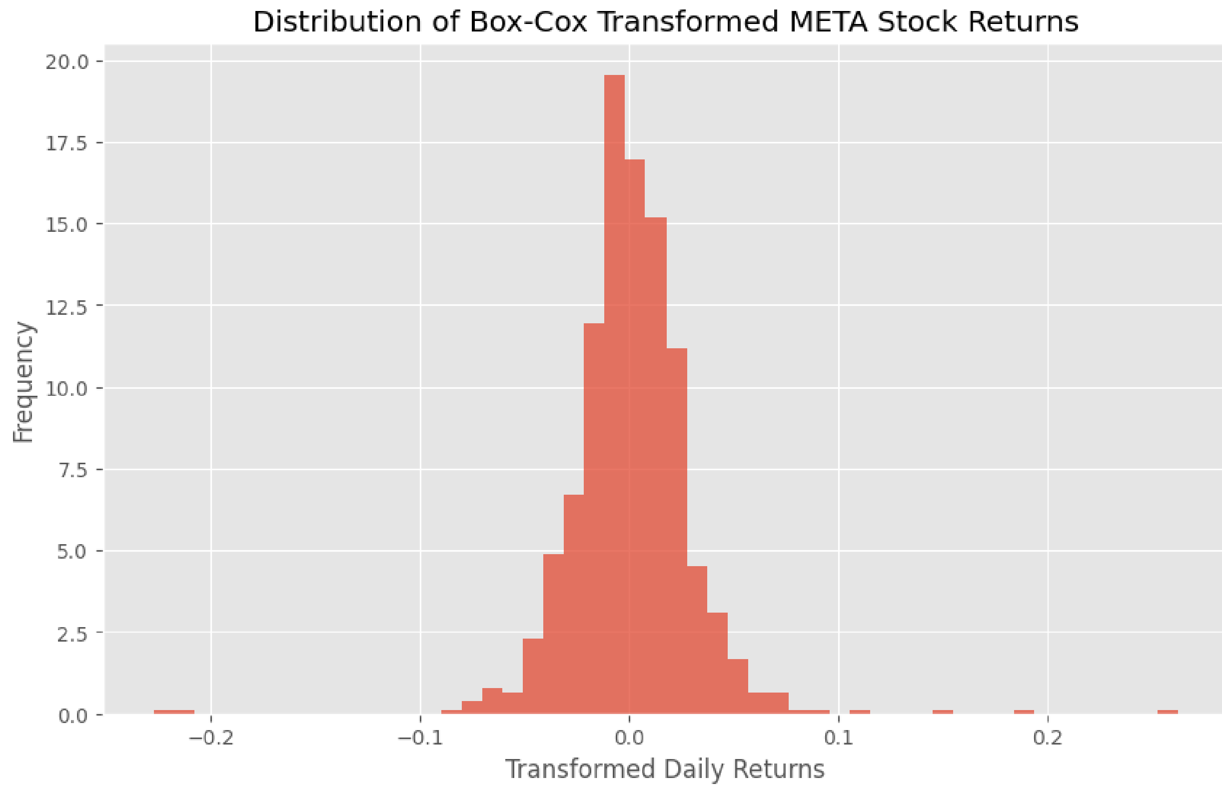
```
 [*********************100%%**********************]  1 of 1 completed
Jarque-Bera Test Statistic: 12781.588713167326
P-value: 0.0
The null hypothesis is rejected. The data does not follow a normal
distribution.
```

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import kurtosis
from datetime import datetime

# Calculate excess kurtosis
excess_kurt = kurtosis(returns, fisher=True)

# Plot the returns distribution
plt.figure(figsize=(10, 6))
plt.hist(returns, bins=50, density=True, alpha=0.75)
plt.title("Distribution of META Stock Returns")
plt.xlabel("Daily Returns")
plt.ylabel("Frequency")
plt.show()

# Display excess kurtosis
excess_kurt
```




```
19.60383946591522
```

```python
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf
from scipy.stats import kurtosis, boxcox
from scipy.special import inv_boxcox

# Box-Cox Transformation
returns_boxcox, lambda_value = boxcox(returns + 1)  # Adding 1 to
handle zero or negative values

# Plot the transformed data
plt.figure(figsize=(10, 6))
plt.hist(returns_boxcox, bins=50, density=True, alpha=0.75)
plt.title("Distribution of Box-Cox Transformed META Stock Returns")
plt.xlabel("Transformed Daily Returns")
plt.ylabel("Frequency")
plt.show()

# Inverse Box-Cox Transformation (if needed)
returns_inverse_boxcox = inv_boxcox(returns_boxcox, lambda_value)

# Calculate excess kurtosis
excess_kurt = kurtosis(returns, fisher=True)
```



Distribution of Box-Cox Transformed META Stock Returns
Frequency
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−0.2
−0.1
0.0
0.1
0.2
Transformed Daily Returns