The Capstone

Forecasting Market Volatility for GBP/USD Using Time Series

March 2024

ISSUED BY
Fahmi Roslan

Introduction

Volatility is one of the most prominent terms you'll hear on any trading floor – and for good reason. In financial markets, volatility captures the amount of fluctuation in prices. High volatility is associated to periods of market turbulence and to large price swings, while low volatility describes more calm and quiet markets.

There are few cause of volatility in forex market such as economic data releases, market sentiment (war etc) and of course market liquidity; Low trading volumes or large trades that will lead to more significant price fluctuations. This project however will go deep and to understand the pattern on specific cause of volatility which is 'Market Liquidity' by measuring volume for specific period of time.

This project will only focus on single forex pair which is GBP/USD and will use Time Series Analysis (TSA) in order to forecast the volume over series of time.

This project obtained the data from Kaggle.com, the chosen dataset is a historical data of GBP/USD recorded daily price movement including volume from year 2000-2023. The original downloaded dataset of GBPUSD consist of 40229 rows & 35 columns.

Executive Summary

Based on the data obtained from Kaggle.com. I shortlisted the method to carry on this analysis project and it definitely will involve time series because the focus of this project is to measure volatility against series of time. The aim of this project :

1. To evaluate the change on participant volume by year in GBP/USD
2. To identify the high & low volatility every specific of time
3. To understand the trend of market liquidity
4. To enhance better trading performance

However, the risk or limitation that will affect the findings of this project is the **validity** of the data. When we are discussing about financial market data there's high percentage that the data are all classified. The initial subject or financial instrument that I want to focus on my capstone project is actually Crude Oil (CL) but due to setback of data I proceed with forex pairs with leverage datasets available.

But, forex pairs are different from stocks or commodities, there are no exchanger that organize and control trading or investing activities. Every traders participate in the market through private brokers. The datasets I obtained definitely from broker. Each broker will come up with different data but the price is still the same.

Hence, when we measuring the volume we need bare in mind that volume of broker A is different from broker B due to number of members.

The metrics that I will use for this project will not be an accuracy. Accuracy is still can be measured using comparison method with different sets of data. But, the key component of this project is not to find the end result but to build a forecasting system that we could rely on to analyse and measure volatility of specific financial instrument subject to availability of data sets. Hence the metrics that I will focus on will be

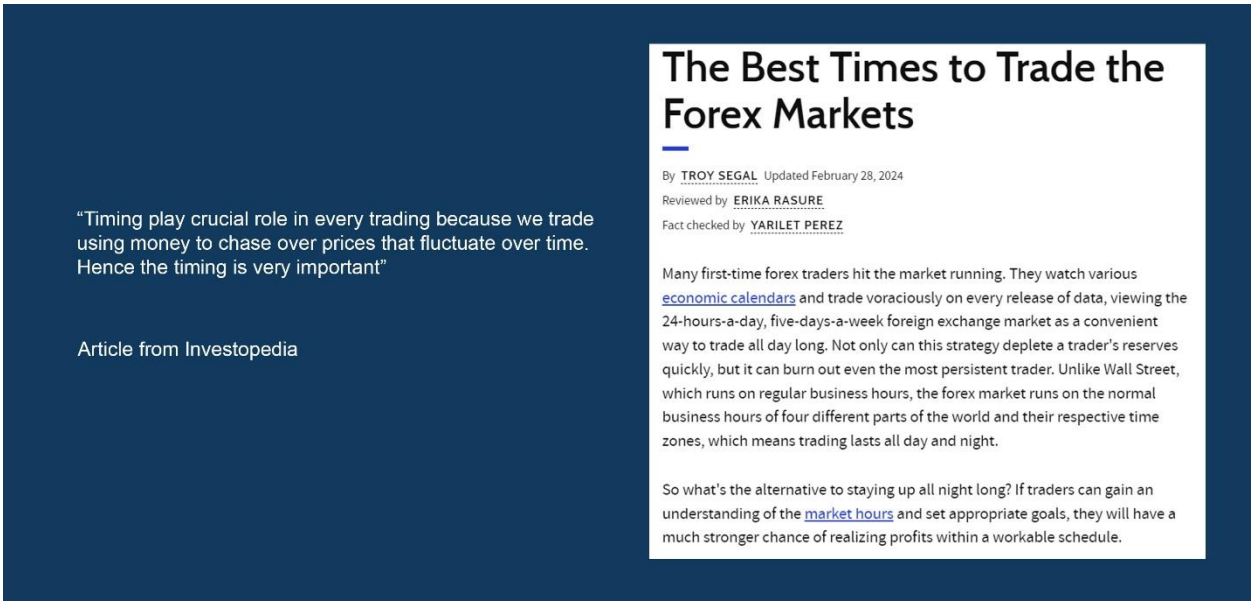
- Accuracy
  Consistency measures how individual data points pulled from two or more sets of data synchronize with one another. If two data points are in conflict, it indicates that one of both of the records are inaccurate.

  For example, you may combine volume data from two different sources. When merging the data for a GBPUSD, you want all the fields to agree – the Open, High, Low, Close and Volume. If the two sources have different of any of that component, the data is inconsistent.

Project Description

Problem Statement :

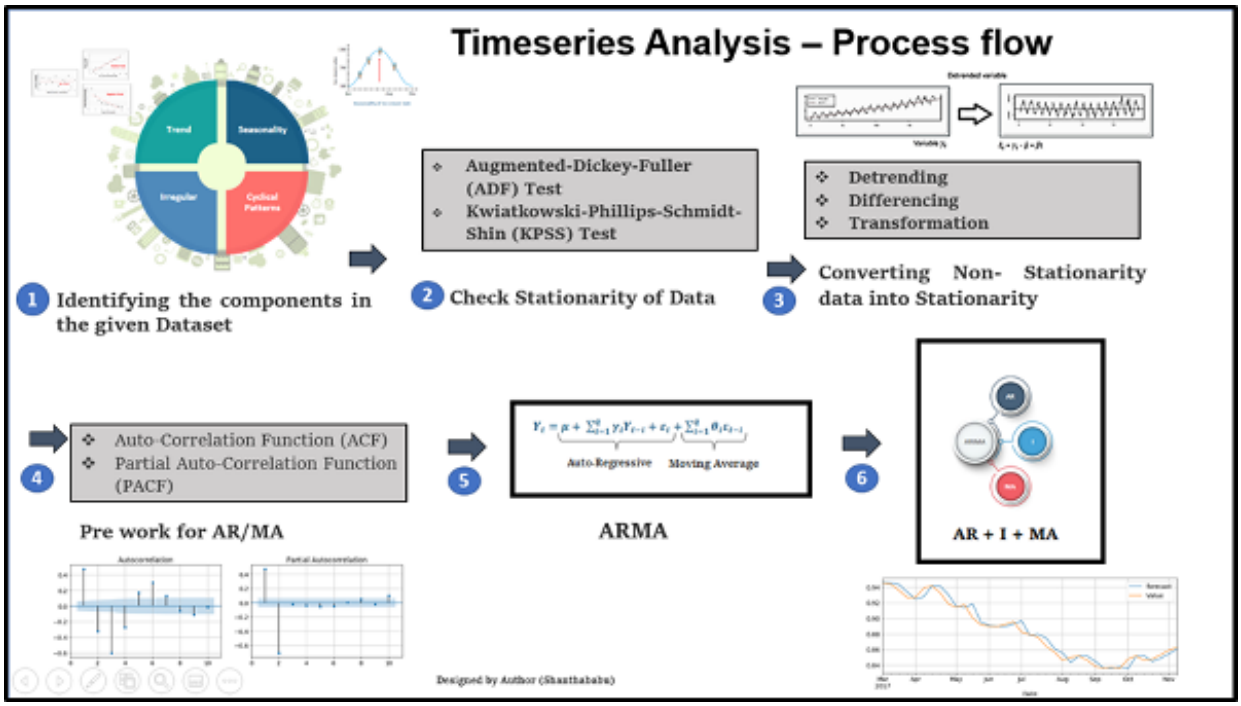
“To understand the long term volatility trend and forecast the short term volume to enhance better trading or investment performance”



Impact on Investors:

- Risk and Uncertainty: Volatility introduces risk and uncertainty into investing. Prices can fluctuate rapidly, leading to potential losses or gains for investors.
- Emotional Stress: High volatility can create emotional stress for investors, potentially leading to impulsive decisions.
- Opportunities: Some investors view volatility as an opportunity to buy undervalued assets during market downturns and sell during upswings.

Time Series Analysis



Time Series Analysis -Process Flow

Multiple components are used to answer the objective of this analysis:

1. Obtain and import data set before start the processing
2. Need to undergo ADF Test (Augmented Dickey Fuller test) to obtain the P-Value
3. Converting the nonstationary into stationary with eliminate the trend and seasonality by differencing method
4. Using ARIMA (Autoregression, Integration & Moving Average)
5. Check accuracy of forecast using accuracy metrics

Part 1 : Exploratory Data

1) Data Processing

- Importing data
- Identify components within dataset
- Set up the index

In [7]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
```

In [8]:
```
# Load data
df = pd.read_csv('gbpusd.csv')
df.head()
```

Out[8]:

	Date	Open	High	Low	Close	Volume
0	2000.01.03	1.6146	1.6400	1.6138	1.6361	4444
1	2000.01.04	1.6359	1.6415	1.6310	1.6373	6141
2	2000.01.05	1.6376	1.6450	1.6354	1.6419	6504
3	2000.01.06	1.6421	1.6511	1.6411	1.6470	6473
4	2000.01.07	1.6476	1.6499	1.6360	1.6394	4754

In [10]:
```
# Change date column to be datetime dtype
df['Date'] = pd.to_datetime(df['Date'])
```

In [11]:
```
# Set Date to be in the index
df.set_index('Date', inplace=True)
```

In [12]:
```
df.head()
```

Out[12]:

Date	Open	High	Low	Close	Volume
2000-01-03	1.6146	1.6400	1.6138	1.6361	4444
2000-01-04	1.6359	1.6415	1.6310	1.6373	6141
2000-01-05	1.6376	1.6450	1.6354	1.6419	6504
2000-01-06	1.6421	1.6511	1.6411	1.6470	6473
2000-01-07	1.6476	1.6499	1.6360	1.6394	4754

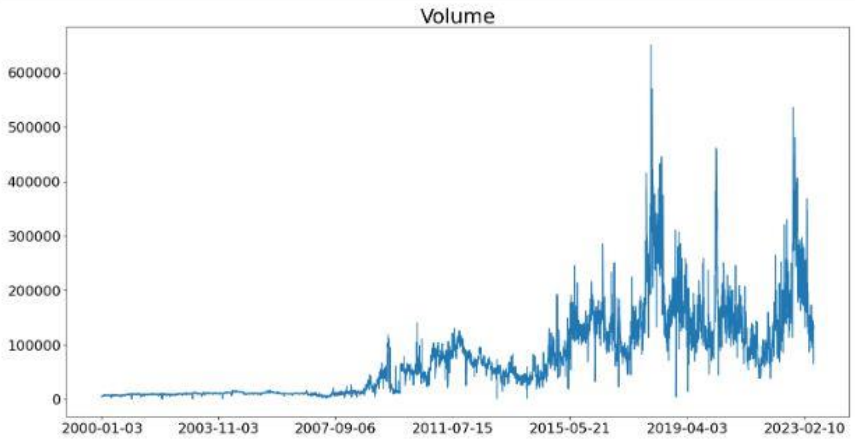
In [9]:
```
df.dtypes
```

Out[9]:
```
Date       object
Open      float64
High      float64
Low       float64
Close     float64
Volume      int64
dtype: object
```

2) Check Stationary of Data

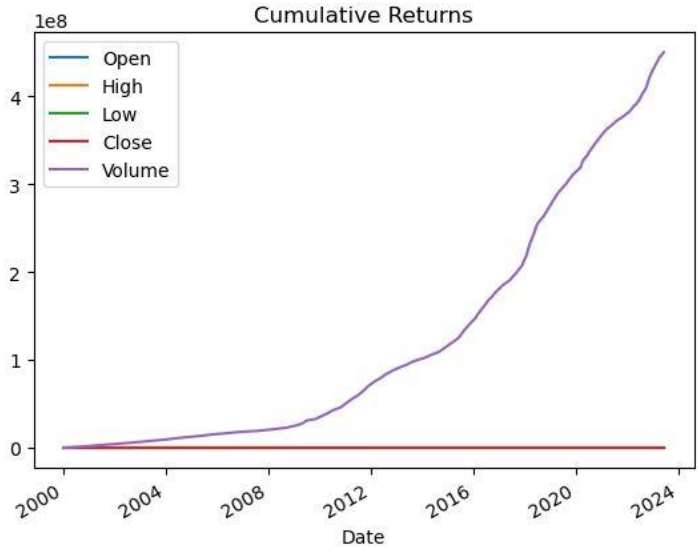
- Plotting the visualization of chosen variable which is 'Volume'
- Test the models with ADF/KPSS test
- Check in the visual data (mean/variance/covariance) over time
- Plot autocorrelation of the variable

```
In [15]: # Generate a time plot of our data.
         plot_series(df, ['Volume'], title = "Volume", steps=1000)
```



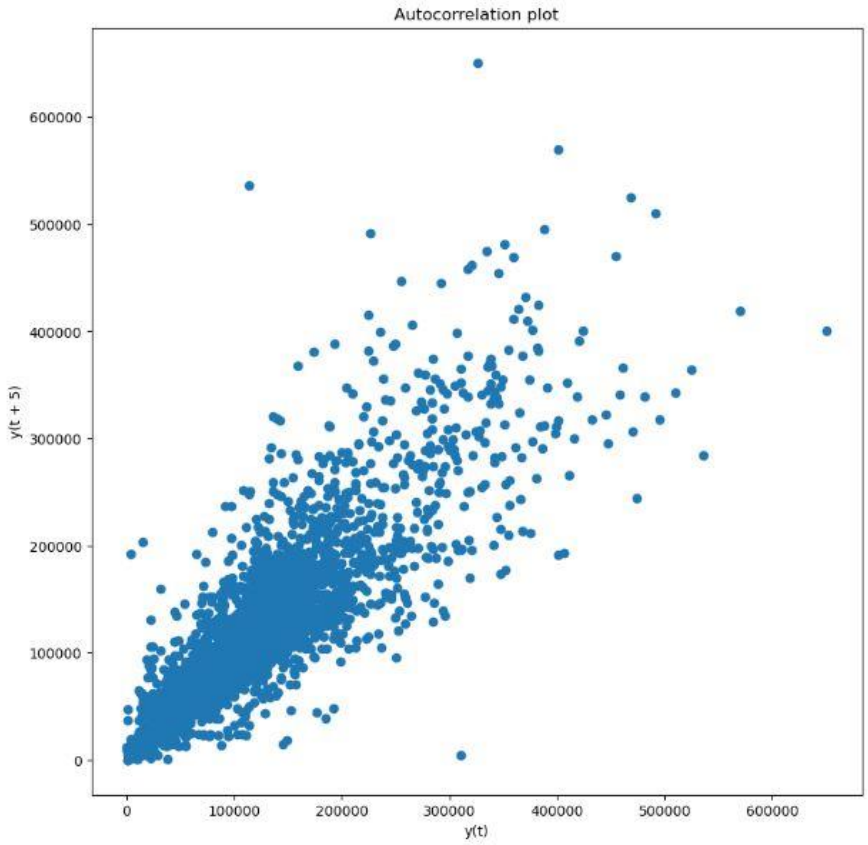

```
In [17]: # Comulative Return
         dr = df.cumsum()
         dr.plot()
         plt.title('Cumulative Returns')
```

```
Out[17]: Text(0.5, 1.0, 'Cumulative Returns')
```



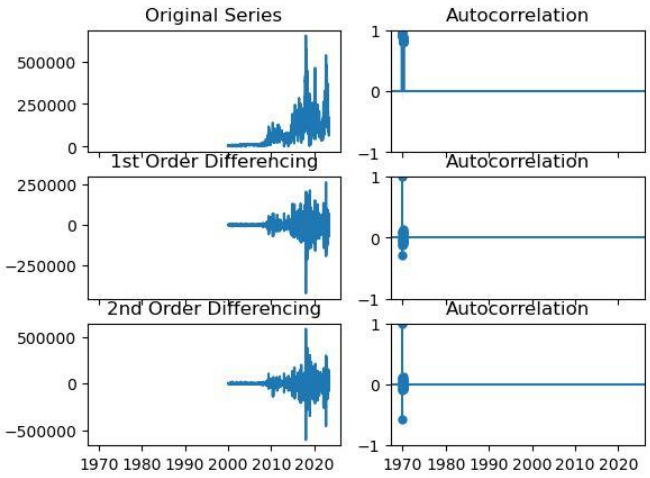
In [27]:
```
plt.figure(figsize=(10,10))
lag_plot(df['Volume'], lag=5)
plt.title('Autocorrelation plot')
```

Out[27]: Text(0.5, 1.0, 'Autocorrelation plot')



Differencing of the model:

```
In [63]: # Original Series
fig, axes = plt.subplots(3, 2, sharex=True)
axes[0, 0].plot(df['Volume']); axes[0, 0].set_title('Original Series')
plot_acf(df['Volume'], ax=axes[0, 1])

# 1st Differencing
axes[1, 0].plot(df['Volume'].diff()); axes[1, 0].set_title('1st Order Differencing')
plot_acf(df['Volume'].diff().dropna(), ax=axes[1, 1])

# 2nd Differencing
axes[2, 0].plot(df['Volume'].diff().diff()); axes[2, 0].set_title('2nd Order Differencing')
plot_acf(df['Volume'].diff().diff().dropna(), ax=axes[2, 1])

plt.show()
```



Checking for stationarity : the Augmented Dickey-Fuller Test (ADF)

```
In [28]: # Import Augmented Dickey-Fuller test.
from statsmodels.tsa.stattools import adfuller

# Run ADF test on original (non-differenced!) data.

adfuller(df['Volume'])
```

```
Out[28]: (-2.943655895014861,
 0.0405018542219686,
 34,
 6044,
 {'1%': -3.4314324089136767,
  '5%': -2.8620183258811283,
  '10%': -2.567024610317412},
 137339.01168246148)
```

3) Feature Engineering (Transforming Non-Stationary)

The result of carried ADF test as below:

Null Hypothesis : The series has a unit root (non stationary)
Test statistic : -2.943656
P-Value : 0.040502
Number of lags : 34
Observations : 6044

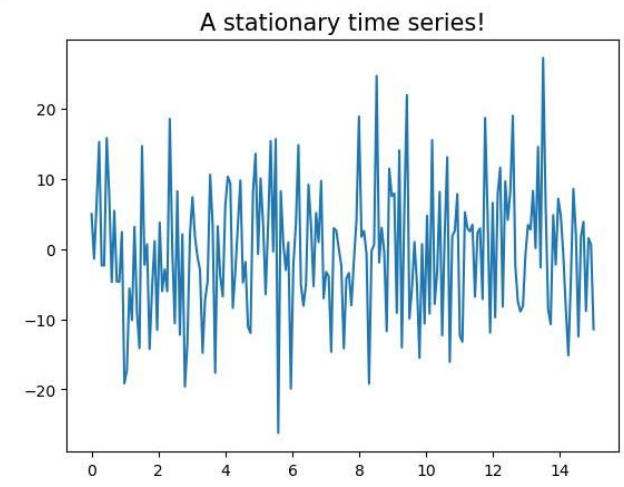
Conclusion : P-Value is smaller than 0.05, we reject the null hypothesis and the series is stationary. Hence the steps as below is not compulsory and the modelling can be start.

Part 2 : Modelling

Modelling has been carried as steps below:

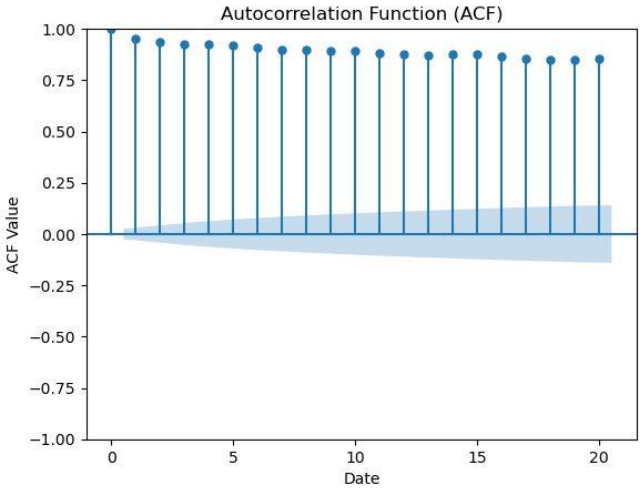
- Plot raw data
- Fitting Model
- Identify parameters
- Manual GridSearch
- Get the p,d,q value of AR1MA

In [35]:
```
# Set seed.
np.random.seed(42)

# Generate 200 steps in time from 0 to 15.
t = np.linspace(0, 15, 200)

# Generate a white noise process with Normally distributed noise.
y = 5 * np.random.normal(0, 2, 200)

# Generate plot.
plt.title('A stationary time series!', fontsize=15)
plt.plot(t, y);
```



In [29]:
```
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.stattools import adfuller
```

In [31]:
```
plot_acf(df['Volume'], lags=20)
plt.title("Autocorrelation Function (ACF)")
plt.xlabel("Date")
plt.ylabel("ACF Value")
plt.show()
```



Manual Gridsearch using this code :

In [15]:
```
# Starting AIC, p, and q.
best_aic = 99 * (10 ** 16)
best_p = 0
best_q = 0

# Use nested for loop to iterate over values of p and q.
for p in range(5):
    for q in range(5):

        # Insert try and except statements.
        try:

            # Fitting an ARIMA(p, 1, q) model.
            print(f'Attempting ARIMA({p}, 1, {q})')

            # Instantiate ARIMA model.
            arima = ARIMA(endog=y_train, order=(p, 1, q))


            # Fit ARIMA model.
            model = arima.fit()

            # Print out AIC for ARIMA(p, 1, q) model.
            print(f'The AIC for ARIMA({p},1,{q}) is: {model.aic}')

            # Is my current model's AIC better than our best_aic?
            if model.aic < best_aic:

                # If so, let's overwrite best_aic, best_p, and best_q.
                best_aic = model.aic
                best_p = p
                best_q = q

        except:
            pass
print()
print()
print('MODEL FINISHED!')
print(f'Our model that minimizes AIC on the training data is the ARIMA({best_p},1,{best_q}).')
print(f'This model has an AIC of {best_aic}.')
```

```
C:\Users\fahmi\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provide
d, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
C:\Users\fahmi\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:966: UserWarning: Non-stationary starting au
toregressive parameters found. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\fahmi\anaconda3\Lib\site-packages\statsmodels\tsa\statespace\sarimax.py:978: UserWarning: Non-invertible starting MA
parameters found. Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'

The AIC for ARIMA(4,1,4) is: 123274.08814598958


MODEL FINISHED!
Our model that minimizes AIC on the training data is the ARIMA(4,1,4).
This model has an AIC of 123274.08814598958.

C:\Users\fahmi\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWarning: Maximum Likelihood optimization
failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

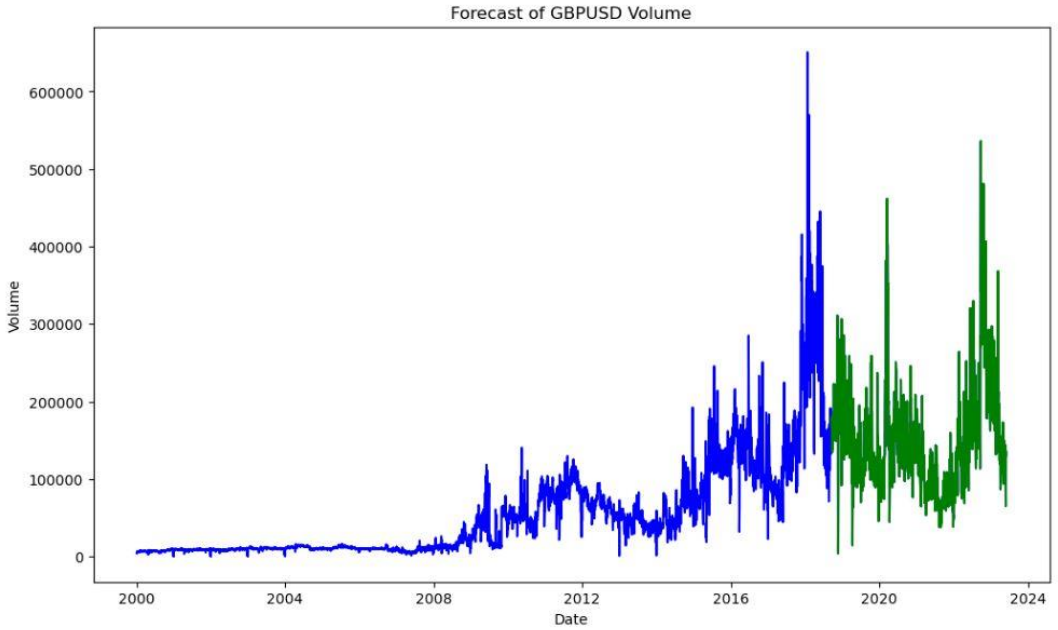
Based on manual GridSearch, we could find the the parameters and value of p-d—q.

The series model that minimizes AIC on the training data is the ARIMA(4,1,4)

p = 4 # Autoregressive order
d = 1 # Differencing order
q = 4 # Moving average order

The AIC for ARIMA 4,1,4 is 123274.08814598958

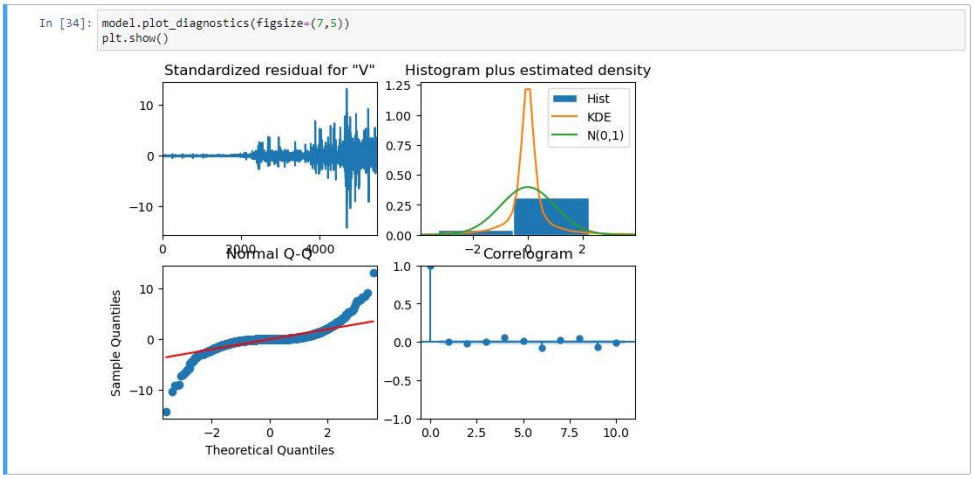
Plotting ARIMA (where green is the prediction on volume) :



Part 3 : Model Evaluation

For evaluation part, there are 2 ways being measured using accuracy in time series forecasting to evaluate performance on predictive models. Accuracy metrics and also interpret using residual plots in ARIMA.

1) **Interpret using Residual plots in ARIMA:**



Top left: The residual errors seem to fluctuate around a mean of zero and have a uniform variance.

Top Right: The density plot suggest normal distribution with mean zero.

Bottom left: All the dots should fall perfectly in line with the red line. Any significant deviations would imply the distribution is skewed.

Bottom Right: The Correlogram, aka, ACF plot shows the residual errors are not autocorrelated. Any autocorrelation would imply that there is some pattern in the residual errors which are not explained in the model. So you will need to look for more X's (predictors) to the model

2) Accuracy Metrics:

Root Mean Squared Error (RMSE): This is the square root of the MSE and provides an interpretable measure of error in the same units as the original data.
RMSE = sqrt(MSE)

```
In [54]: from sklearn.metrics import mean_squared_error

In [61]: from statsmodels.tsa.arima.model import ARIMA
         from sklearn.metrics import mean_squared_error
         import numpy as np

         # Example data
         actual_values = np.array([10, 20, 30, 40, 50])  # Actual values for the period you made predictions for

         # Fit ARIMA model
         model = ARIMA(actual_values, order=(1, 0, 0))  # Example ARIMA model order
         fit_model = model.fit()

         # Make predictions
         forecast = fit_model.forecast(steps=len(actual_values))  # Predictions for the same period as actual values

         # Calculate squared errors
         squared_errors = (actual_values - forecast)**2

         # Calculate mean squared error
         mse = np.mean(squared_errors)

         # Calculate RMSE
         rmse = np.sqrt(mse)

         print("RMSE:", rmse)

         RMSE: 20.73247619699088
```

RMSE: 20.73247619699088

Mean Absolute Error (MAE): This is the average of the absolute errors between the predicted values and the actual values. It provides a simple and interpretable measure of accuracy.
MAE = Σ|Actual - Predicted| / n
 # MAE: 49.99999601481055

Mean Absolute Percentage Error (MAPE): This measures the average absolute percentage difference between the actual and predicted values, making it easy to understand the magnitude of error relative to the actual values.
MAPE = Σ(|Actual - Predicted| / Actual) * 100 / n
 # MAPE: 228.33332232842398

Mean Absolute Scaled Error (MASE) : is a metric used to measure the accuracy of forecasts, particularly in time series forecasting, by comparing the forecast errors to a naive forecast. MASE is robust to different scales and is useful for comparing the accuracy of different forecasting methods on different datasets.
 # MASE: 1.5584414342278614

Findings

After evaluating model I can conclude that this model is not very good based on my the accuracy metrics. My scale of data is quite a big range and the RMSE is not very near to 0. I decided to also use MAPE which involves several considerations, similar to evaluating other error metrics and resulting in high percentage (considered as negative reading.

Hence, my problem statement cannot be answered using this model at this stage. However, the positive positive finding of this Time Series Analysis is actually very significant which is;

To develop a baseline model to be the anchor for my next time series forecasting model that can be use in forecasting any financial instrument over time. With this model I can assess different future developed model to enhance accuracy.

These are few steps to improve model that I have identified:

1. Data Preprocessing:
    - Ensure your time series data is stationary by removing trends and seasonality, if present.
    - Handle missing values appropriately, either by imputation or using models that can handle missing data.
    - Normalize or scale your data if the magnitude of the features varies widely.

2. Assessing Autocorrelation:
    - Autocorrelation plots (ACF and PACF) of the residuals can help identify any remaining autocorrelation in the model's errors. In an ideal scenario, the autocorrelation plot should not show any significant spikes beyond the confidence intervals, indicating that the residuals are uncorrelated.

3. Model Selection:
    - Experiment with different types of time series models, such as ARIMA, SARIMA, exponential smoothing methods (e.g., Holt-Winters), and machine learning algorithms (e.g., decision trees, random forests, gradient boosting).
    - Choose models that are appropriate for the specific characteristics of your data (e.g., seasonality, trend, noise).

4. Parameter Tuning:
    - Optimize the hyperparameters of your chosen models using techniques like grid search, random search, or Bayesian optimization.
    - Fine-tune parameters such as the order of differencing, autoregressive (AR) order, moving average (MA) order, and seasonal components.

5. Cross-Validation:
    - Use cross-validation techniques (e.g., time series cross-validation, rolling origin validation) to evaluate the performance of your models and avoid overfitting.
    - Ensure that your models generalize well to unseen data by testing them on out-of-sample data.

    By implementing these strategies and iteratively refining your models, you can improve the accuracy of your time series forecasts and make more informed decisions based on the predictions. Remember that achieving high accuracy may require a combination of data preprocessing, model selection, and careful parameter tuning.