

1.0-prof-data-acquisition

June 29, 2023

```
[ ]: import pandas as pd
import seaborn as sns
import numpy as np
import pymysql

from sqlalchemy import create_engine
from tqdm import tqdm
from binance import Client
sns.set()
pymysql.install_as_MySQLdb()

[ ]: engine = create_engine('sqlite:///Binance_daily.db')

[ ]: client = Client()

[ ]: symbols = ['BTCUSDT', 'ETHUSDT', 'ADAUSDT', 'BNBUSDT']

[ ]: def get_data(symbol, interval = client.KLINE_INTERVAL_12HOUR, start = '2018-06-01'):
    temp = client.get_historical_klines(symbol, interval, start)
    temp = pd.DataFrame(temp)
    temp = temp.iloc[:, :6]
    temp.columns = ["Date", "Open", "High", "Low", "Close", "Volume"]
    temp.Date = pd.to_datetime(temp.Date, unit = "ms")
    temp = temp.set_index("Date", drop = True)
    temp = temp.astype(float)
    return temp

[ ]: for coin in tqdm(symbols):
    df = get_data(coin)
    df.to_sql(coin, if_exists= 'replace', con = engine)

100%|      | 4/4 [00:38<00:00,  9.70s/it]

[ ]: def qry(symbol):
    df = pd.read_sql(symbol, engine)
    df = df.set_index('Date')
```

```
df = df.asfreq('12H')
return df.Close
```

```
[ ]: def apply_technical(df):

    for column in df.columns.to_list():
        #calculating returns
        df[f'Ret_{column}'] = df[column].pct_change().mul(100)
        df[f'Ret_cum_{column}'] = df[f'Ret_{column}'].cumsum()
        #normalizing values
        benchmark = df.loc[:,column][0]
        df[f'Norm_{column}'] = df[column].div(benchmark).mul(100)
    return df
```

```
[ ]: def process_coin(symbols):
    df_list = []

    for coin in symbols:
        df = qry(coin)
        df.name = coin
        df_list.append(df)
    df = pd.DataFrame(df_list).T
    df = apply_technical(df)

    return df
```

```
[ ]: df = process_coin(symbols = symbols)
```

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

```
[ ]:
```

	BTCUSDT	ETHUSDT	ADAUSDT	BNBUSDT	Ret_BTCUSDT
Date					
2018-06-01 00:00:00	7361.39	570.63	0.21580	13.7800	NaN \
2018-06-01 12:00:00	7521.01	579.00	0.22038	14.2888	2.168340
2018-06-02 00:00:00	7652.28	592.73	0.22428	14.5000	1.745377
2018-06-02 12:00:00	7640.03	590.85	0.22648	14.6732	-0.160083
2018-06-03 00:00:00	7714.85	619.93	0.23284	14.7861	0.979316

	Ret_cum_BTCUSDT	Norm_BTCUSDT	Ret_ETHUSDT
Date			
2018-06-01 00:00:00	NaN	100.000000	NaN \
2018-06-01 12:00:00	2.168340	102.168340	1.466800
2018-06-02 00:00:00	3.913718	103.951563	2.371330
2018-06-02 12:00:00	3.753635	103.785155	-0.317176
2018-06-03 00:00:00	4.732950	104.801539	4.921723

	Ret_cum_ETHUSDT	Norm_ETHUSDT	Ret_ADAUSDT
Date			
2018-06-01 00:00:00	NaN	100.000000	NaN \
2018-06-01 12:00:00	1.466800	101.466800	2.122335
2018-06-02 00:00:00	3.838130	103.872912	1.769671
2018-06-02 12:00:00	3.520953	103.543452	0.980917
2018-06-03 00:00:00	8.442676	108.639574	2.808195

	Ret_cum_ADAUSDT	Norm_ADAUSDT	Ret_BNBUSDT
Date			
2018-06-01 00:00:00	NaN	100.000000	NaN \
2018-06-01 12:00:00	2.122335	102.122335	3.692308
2018-06-02 00:00:00	3.892006	103.929564	1.478081
2018-06-02 12:00:00	4.872923	104.949027	1.194483
2018-06-03 00:00:00	7.681118	107.896200	0.769430

	Ret_cum_BNBUSDT	Norm_BNBUSDT
Date		
2018-06-01 00:00:00	NaN	100.000000
2018-06-01 12:00:00	3.692308	103.692308
2018-06-02 00:00:00	5.170388	105.224964
2018-06-02 12:00:00	6.364871	106.481858
2018-06-03 00:00:00	7.134301	107.301161

```
[ ]: def train_split(df, size:int):
    percentage = int(len(df) * size)
    data = df.iloc[:percentage]
    data_test = df.iloc[percentage:]

    data.to_csv('../data/processed/data_processed.csv')
    data_test.to_csv('../data/processed/data_test.csv')

    return len(data), len(data_test)
```

```
[ ]: train_split(df, 0.7)
```

```
[ ]: (2596, 1113)
```

```
[ ]:
```

2.0-prof-initial_data_exploration

June 29, 2023

```
[ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import statsmodels
import seaborn as sns
import scipy.stats as ss
import pylab
import pymysql
import statsmodels.graphics.tsaplots as sgt
import statsmodels.tsa.stattools as sts
import arch
import datetime as dt

from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.ensemble import IsolationForest
from statsmodels.tsa.arima.model import ARIMA
from scipy.stats.distributions import chi2
from tqdm import tqdm
from time import sleep
import warnings
warnings.filterwarnings('ignore', category=Warning)
sns.set()
```

```
[ ]: def read_csv(name:str):
    df = pd.read_csv(f'../data/processed/{name}.csv')
    df.Date = pd.to_datetime(df.Date)
    df.set_index('Date', inplace = True)
    df.asfreq('12H')
    return df
```

```
[ ]: data_raw = read_csv('data_processed')
df = data_raw.copy()
```

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

```
[ ]: BTCUSDT ETHUSDT ADAUSDT BNBUSDT Ret_BTCUSDT
Date
2018-06-01 00:00:00 7361.39 570.63 0.21580 13.7800 NaN \
2018-06-01 12:00:00 7521.01 579.00 0.22038 14.2888 2.168340
2018-06-02 00:00:00 7652.28 592.73 0.22428 14.5000 1.745377
2018-06-02 12:00:00 7640.03 590.85 0.22648 14.6732 -0.160083
2018-06-03 00:00:00 7714.85 619.93 0.23284 14.7861 0.979316
```

```
Ret_cum_BTCUSDT Norm_BTCUSDT Ret_ETHUSDT
Date
2018-06-01 00:00:00 NaN 100.000000 NaN \
2018-06-01 12:00:00 2.168340 102.168340 1.466800
2018-06-02 00:00:00 3.913718 103.951563 2.371330
2018-06-02 12:00:00 3.753635 103.785155 -0.317176
2018-06-03 00:00:00 4.732950 104.801539 4.921723
```

```
Ret_cum_ETHUSDT Norm_ETHUSDT Ret_ADAUSDT
Date
2018-06-01 00:00:00 NaN 100.000000 NaN \
2018-06-01 12:00:00 1.466800 101.466800 2.122335
2018-06-02 00:00:00 3.838130 103.872912 1.769671
2018-06-02 12:00:00 3.520953 103.543452 0.980917
2018-06-03 00:00:00 8.442676 108.639574 2.808195
```

```
Ret_cum_ADAUSDT Norm_ADAUSDT Ret_BNBUSDT
Date
2018-06-01 00:00:00 NaN 100.000000 NaN \
2018-06-01 12:00:00 2.122335 102.122335 3.692308
2018-06-02 00:00:00 3.892006 103.929564 1.478081
2018-06-02 12:00:00 4.872923 104.949027 1.194483
2018-06-03 00:00:00 7.681118 107.896200 0.769430
```

```
Ret_cum_BNBUSDT Norm_BNBUSDT
Date
2018-06-01 00:00:00 NaN 100.000000
2018-06-01 12:00:00 3.692308 103.692308
2018-06-02 00:00:00 5.170388 105.224964
2018-06-02 12:00:00 6.364871 106.481858
2018-06-03 00:00:00 7.134301 107.301161
```

```
[ ]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2596 entries, 2018-06-01 00:00:00 to 2021-12-19 12:00:00
Data columns (total 16 columns):
#   Column          Non-Null Count  Dtype
---  -
0   BTCUSDT         2596 non-null   float64
```

```

1  ETHUSDT      2596 non-null  float64
2  ADAUSDT      2596 non-null  float64
3  BNBUSDT      2596 non-null  float64
4  Ret_BTCUSDT  2595 non-null  float64
5  Ret_cum_BTCUSDT 2595 non-null  float64
6  Norm_BTCUSDT 2596 non-null  float64
7  Ret_ETHUSDT  2595 non-null  float64
8  Ret_cum_ETHUSDT 2595 non-null  float64
9  Norm_ETHUSDT 2596 non-null  float64
10 Ret_ADAUSDT  2595 non-null  float64
11 Ret_cum_ADAUSDT 2595 non-null  float64
12 Norm_ADAUSDT 2596 non-null  float64
13 Ret_BNBUSDT  2595 non-null  float64
14 Ret_cum_BNBUSDT 2595 non-null  float64
15 Norm_BNBUSDT 2596 non-null  float64

```

dtypes: float64(16)

memory usage: 344.8 KB

```
[ ]: df.describe()
```

```

[ ]:
count      BTCUSDT      ETHUSDT      ADAUSDT      BNBUSDT  Ret_BTCUSDT  \
mean    19073.491984    928.039823    0.464944    114.735015    0.107102
std     18320.875224   1229.798002    0.715065    181.000502    2.666371
min       3201.240000     83.670000    0.022290     4.516000   -23.536521
25%       6758.510000    179.777500    0.048727    14.843825   -0.907544
50%       9517.580000    265.860000    0.089635    21.840900    0.112227
75%      33116.050000   1372.190000    0.368208    44.755575    1.155585
max      67594.980000   4807.980000    3.048000    676.150000   17.633807

count      Ret_cum_BTCUSDT  Norm_BTCUSDT  Ret_ETHUSDT  Ret_cum_ETHUSDT  \
mean           96.164292    259.101773     0.134348     46.479023
std          113.024360    248.877932     3.436786    155.336940
min          -73.204783     43.486896    -29.813473   -169.725848
25%           3.465213     91.810242    -1.287288    -70.261637
50%          61.656395    129.290528     0.071052    -18.897994
75%          223.565446    449.861371     1.697319    195.277387
max          311.787068    918.236637     18.345390    364.551722

count      Norm_ETHUSDT  Ret_ADAUSDT  Ret_cum_ADAUSDT  Norm_ADAUSDT  Ret_BNBUSDT  \
mean       162.634250     0.150224     61.607932    215.451493     0.212713
std       215.515834     4.066955    197.088596    331.355258     3.791113
min        14.662741    -30.108340   -173.410435     10.329008    -31.200628
25%        31.505091    -1.786669    -88.009456     22.579935    -1.465138
50%        46.590610     0.022198   -38.347904     41.536145     0.083649

```

75%	240.469306	2.013309	192.112207	170.624421	1.879254
max	842.573997	23.258978	466.725618	1412.418906	35.598364

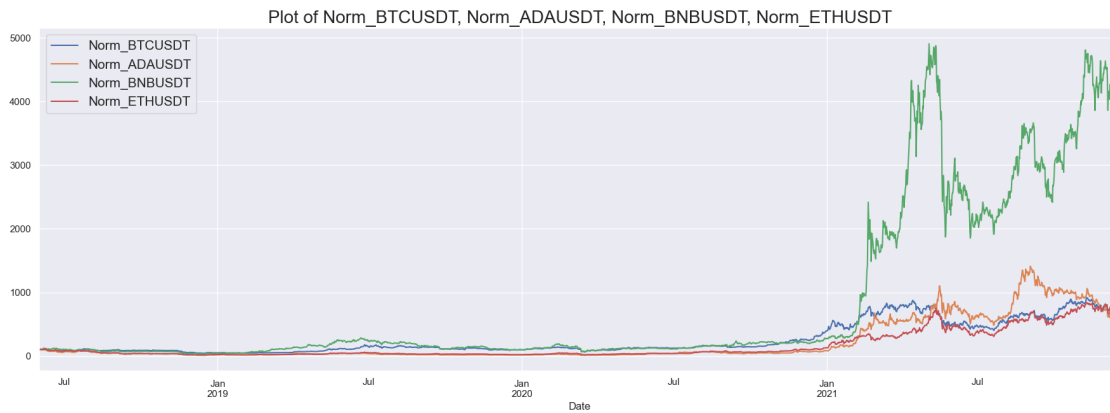
	Ret_cum_BNBUSDT	Norm_BNBUSDT
count	2595.000000	2596.000000
mean	177.478157	832.619844
std	193.880197	1313.501468
min	-91.025486	32.772134
25%	57.958113	107.720065
50%	107.057233	158.497097
75%	220.457945	324.786466
max	569.890575	4906.748911

```
[ ]: df.isna().sum()
```

```
[ ]: BTCUSDT      0
     ETHUSDT      0
     ADAUSDT      0
     BNBUSDT      0
     Ret_BTCUSDT   1
     Ret_cum_BTCUSDT 1
     Norm_BTCUSDT  0
     Ret_ETHUSDT   1
     Ret_cum_ETHUSDT 1
     Norm_ETHUSDT  0
     Ret_ADAUSDT   1
     Ret_cum_ADAUSDT 1
     Norm_ADAUSDT  0
     Ret_BNBUSDT   1
     Ret_cum_BNBUSDT 1
     Norm_BNBUSDT  0
     dtype: int64
```

```
[ ]: def plot(df, columns, cumsum = False):
      fig, ax = plt.subplots(figsize=(22, 7))
      for column in columns:
          if cumsum == True:
              df[column].cumsum().plot(ax=ax, label=column)
          else:
              df[column].plot(ax=ax, label=column)
      plt.title(f"Plot of {'', '.join(columns)}", fontsize = 20)
      plt.legend(fontsize = 15)
      file_name = "_".join(columns) + "_plot.png"
      plt.savefig(f'../reports/figures/{file_name}')
      plt.show()
```

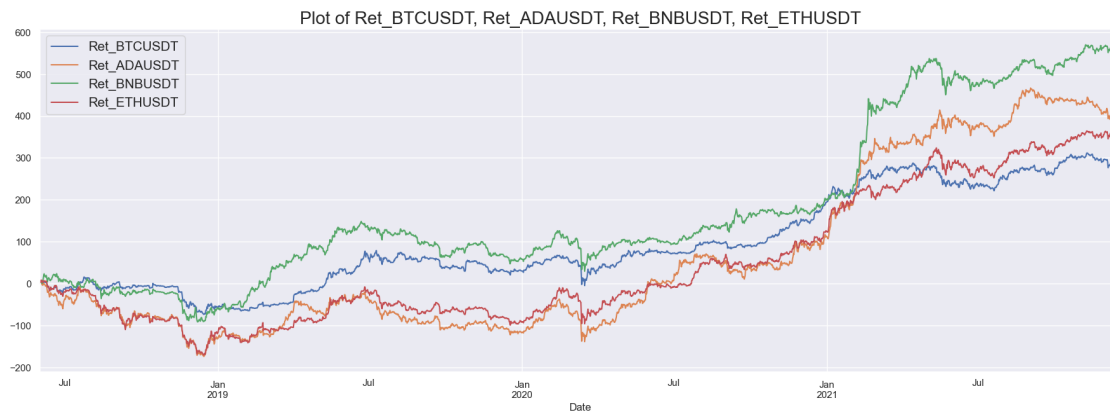
```
[ ]: plot(df, ['Norm_BTCUSDT', 'Norm_ADAUSDT', 'Norm_BNBUSDT', 'Norm_ETHUSDT'])
```



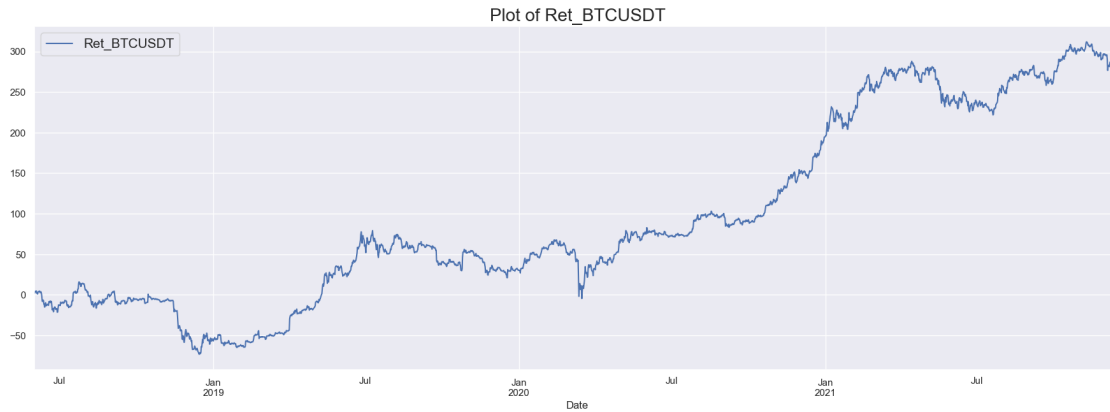
```
[ ]: df.BNBUSDT.nlargest()
```

```
[ ]: Date
2021-05-03 12:00:00    676.15
2021-05-11 12:00:00    671.91
2021-05-09 00:00:00    668.61
2021-05-10 00:00:00    664.77
2021-05-12 00:00:00    663.99
Name: BNBUSDT, dtype: float64
```

```
[ ]: plot(df, ['Ret_BTCUSDT', 'Ret_ADAUSDT', 'Ret_BNBUSDT', 'Ret_ETHUSDT'],
          ↪cumsum=True)
```



```
[ ]: plot(df, ['Ret_BTCUSDT'], cumsum=True)
```



0.1 QQ Plot

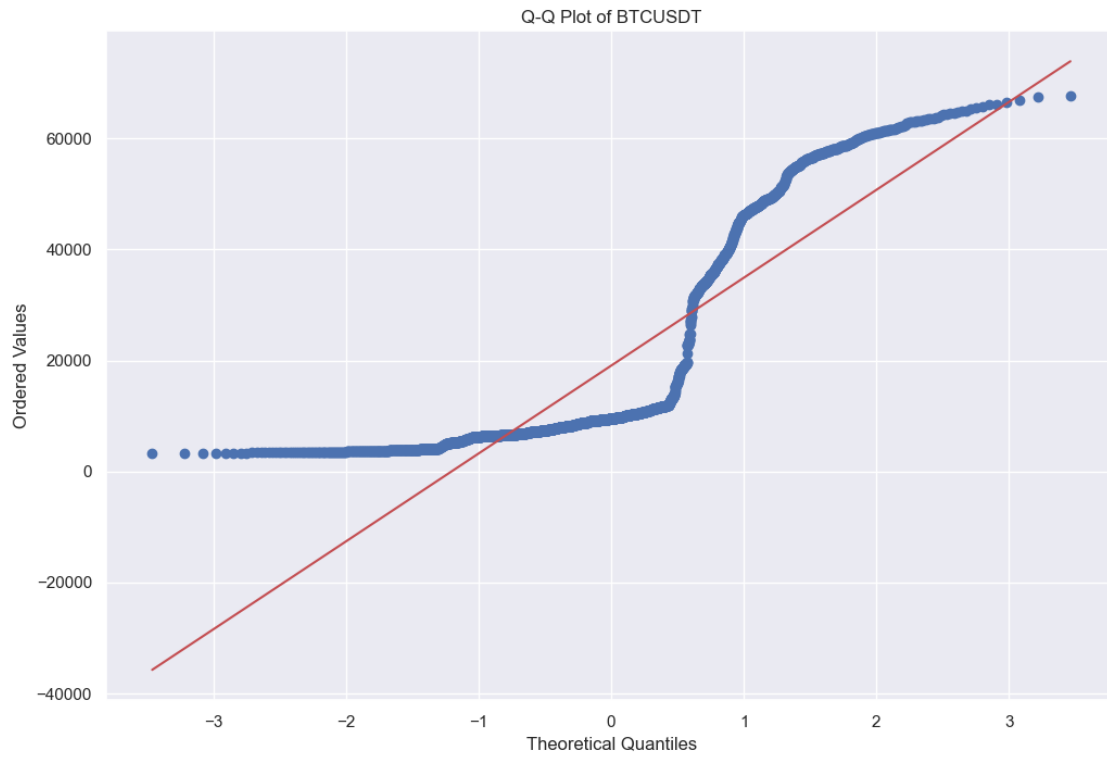
The QQ Plot below indicates that values are not normally distributed

```
[ ]: def qqplot(df, column):
    fig, ax = plt.subplots(figsize=(12, 8))
    ss.probplot(df[column], plot=plt)
    plt.title(f"Q-Q Plot of {column}")
    plt.xlabel("Theoretical Quantiles")
    plt.ylabel("Ordered Values")

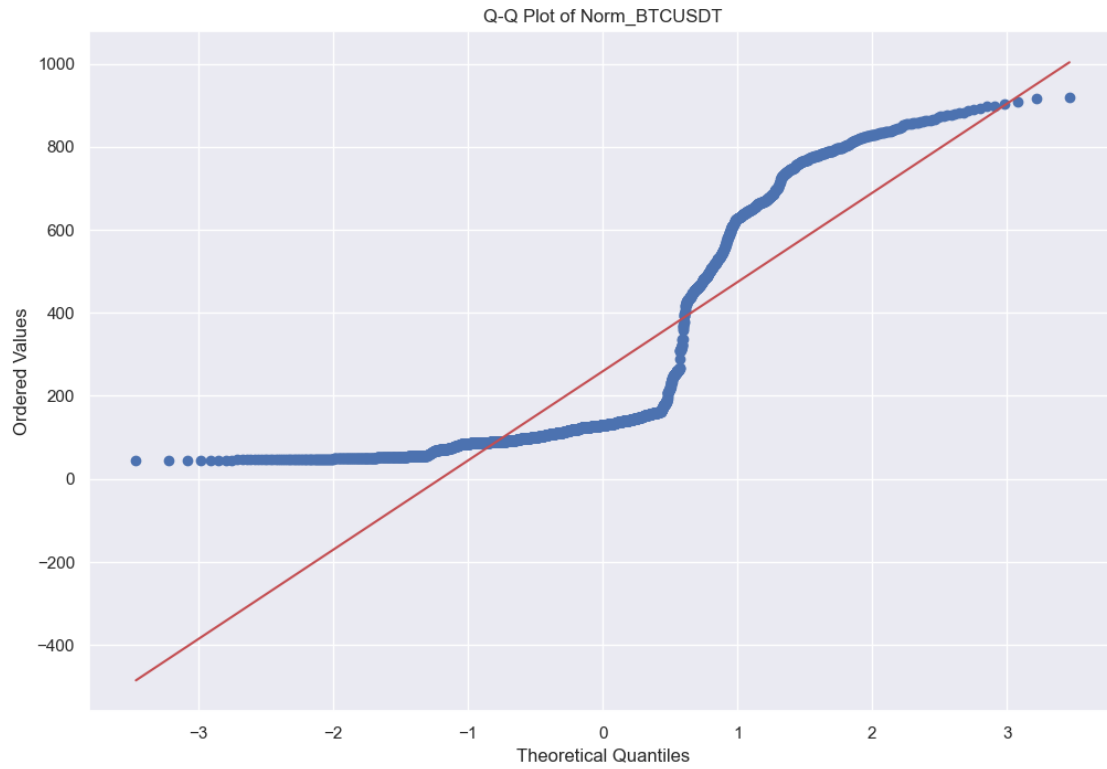
    # Generate the file name based on the column name
    file_name = f"{column}_qqplot.png"

    # Save the figure in the specified directory
    plt.savefig(f'../reports/figures/{file_name}')
    plt.show()
```

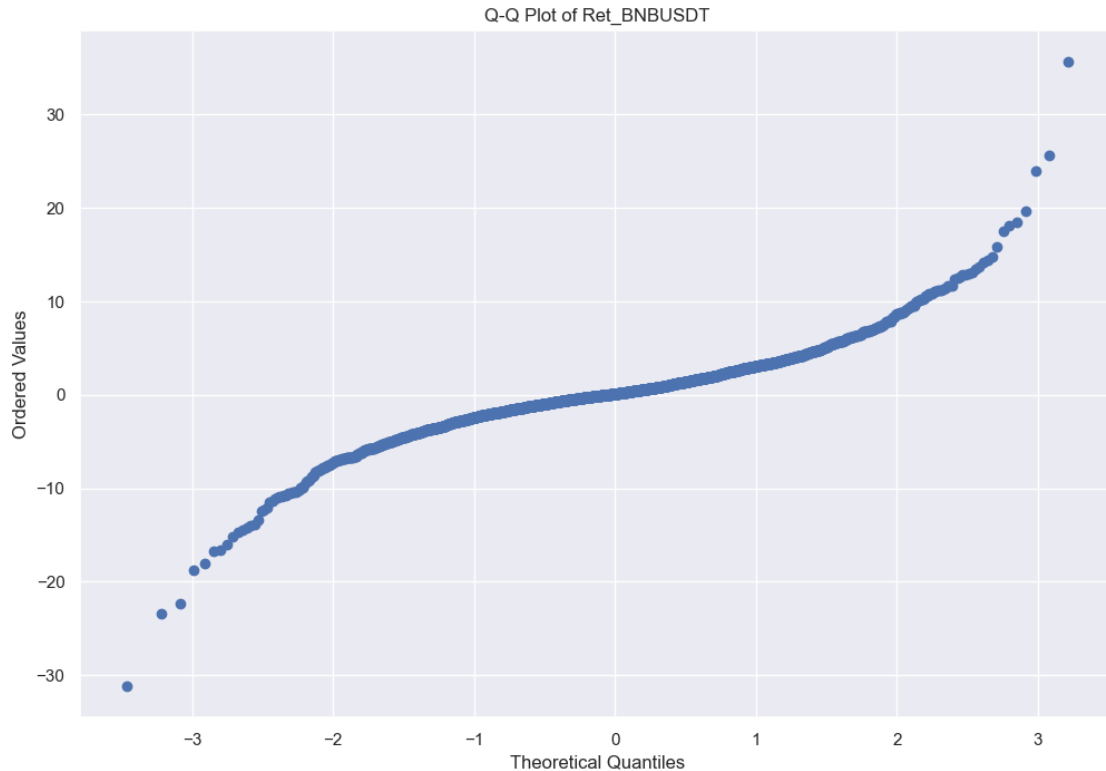
```
[ ]: qqplot(df, 'BTCUSD' )
```



```
[ ]: qqplot(df, 'Norm_BTCUSDT')
```



```
[ ]: qqplot(df, 'Ret_BNBUSDT')
```



0.2 Test of stationality

ADF Test Results for BTCUSDT Closing Prices The Augmented Dickey-Fuller (ADF) test was conducted on the data. The purpose of this test is to determine whether the series is stationary or non-stationary.

- The test statistic is insignificant at the 5% level.
- These results indicate that there is insufficient evidence to reject the null hypothesis of non-stationarity.

Therefore, based on the ADF test, the data are likely non-stationary, suggesting the presence of trends or other forms of non-random behavior. These further guides our model choice which according to the data below should be Arimax or sarimax

```
[ ]: sts.adfuller(df.BTCUSDT)
```

```
[ ]: (-0.7299171849555073,
      0.838847397321332,
      23,
      2572,
      {'1%': -3.4328950382761456,
       '5%': -2.862664398228024,
       '10%': -2.567368558377212},
```

```
41475.08601964069)
```

```
[ ]: sts.adfuller(df.BNBUSDT)
```

```
[ ]: (-0.3970086199232143,  
      0.9105970339172873,  
      28,  
      2567,  
      {'1%': -3.432900000469521,  
       '5%': -2.8626665895880508,  
       '10%': -2.567369725077316},  
      18633.803068122936)
```

```
[ ]: sts.adfuller(df.ADAUSD)
```

```
[ ]: (-1.199566751689236,  
      0.6737820566420791,  
      23,  
      2572,  
      {'1%': -3.4328950382761456,  
       '5%': -2.862664398228024,  
       '10%': -2.567368558377212},  
      -9684.84142999204)
```

```
[ ]: sts.adfuller(df.ETHUSD)
```

```
[ ]: (0.3859014334148559,  
      0.9809599813512939,  
      25,  
      2570,  
      {'1%': -3.432897020834196,  
       '5%': -2.8626652737482425,  
       '10%': -2.5673690245121046},  
      27929.469537041532)
```

0.3 Test of Seasonality

- This shows that the data doesn't follow a seasonal trend
- looking into the residuals it indicates the crypto boom in 2021 and beyond

```
[ ]: def seasonal_decompose_plots(df, column):  
      result = seasonal_decompose(df[column].iloc[1:], model='additive')  
  
      fig, axes = plt.subplots(4, 1, figsize=(12, 8))  
  
      axes[0].plot(result.observed)  
      axes[0].set_ylabel('Observed')
```

```

axes[1].plot(result.trend)
axes[1].set_ylabel('Trend')

axes[2].plot(result.seasonal)
axes[2].set_ylabel('Seasonal')

axes[3].plot(result.resid)
axes[3].set_ylabel('Residual')

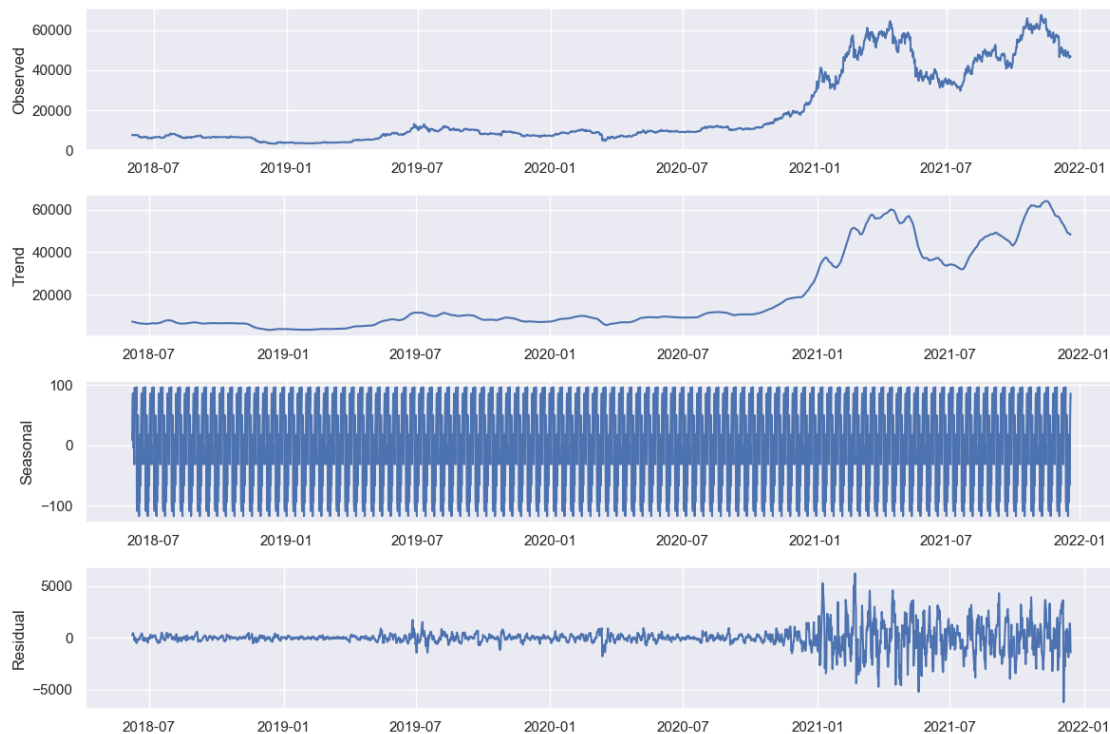
plt.tight_layout()

# Generate the file name based on the column name
file_name = f"{column}_decompose_additive.png"

# Save the figure in the specified directory
plt.savefig(f'../reports/figures/{file_name}')
plt.show()

```

```
[ ]: seasonal_decompose_plots(df, 'BTCUSDT')
```



```
[ ]: seasonal_decompose_plots(df, 'BNBUSDT')
```



```
[ ]: seasonal_decompose_plots(df, 'ADAUSDTCryptocurrency')
```

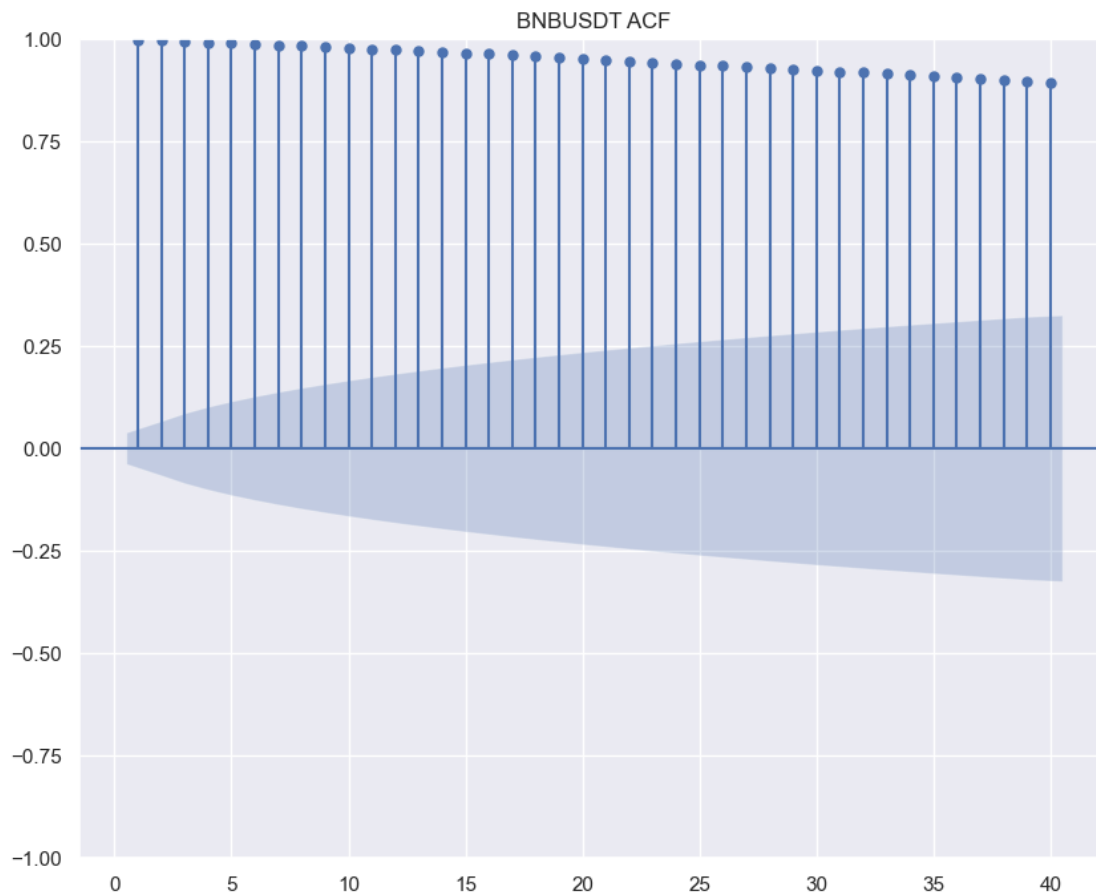


0.3.1 seasonal decompose plot show that there is no seasonality in the data which means that the btcusdt is not a seasonal data also the residual shows that there was some instablity in the data in 2021 up untill early 2022

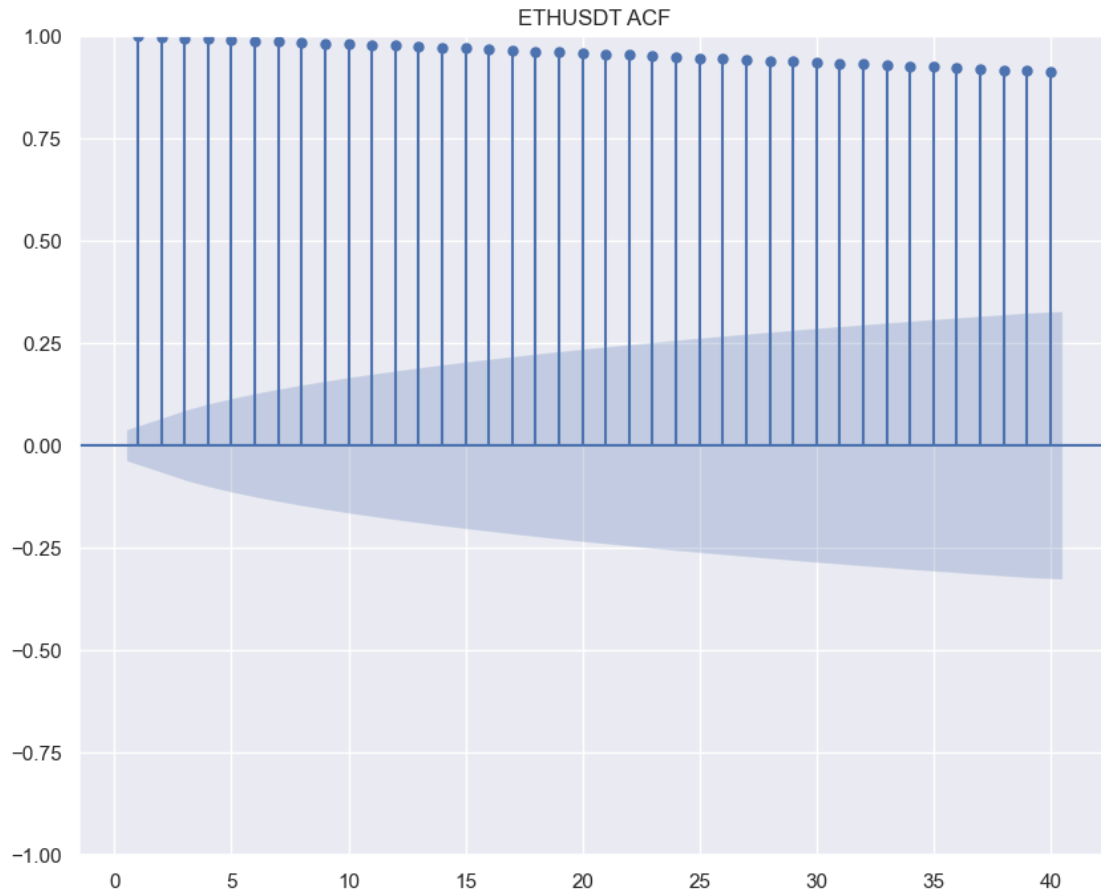
0.4 Plotting ACF

```
[ ]: def acf_plot(df, column, lags = 40):  
    fig, ax = plt.subplots(figsize=(10, 8))  
    sgt.plot_acf(df[column], lags = lags, zero=False, ax=ax)  
    plt.title(f"{column} ACF")  
  
    # Generate the file name based on the column name  
    file_name = f"{column}_acf.png"  
  
    # Save the figure in the specified directory  
    plt.savefig(f'../reports/figures/{file_name}')  
    plt.show()
```

```
[ ]: acf_plot(df, 'BNBUSDT')
```



```
[ ]: acf_plot(df, 'ETHUSDT', lags = 40)
```



0.4.1 This acf suggest that coefficient are significant which shows time dependance in the data also the shows that prices 30 days ago and beyond isnt really a good indicator of current prices

0.5 PACF

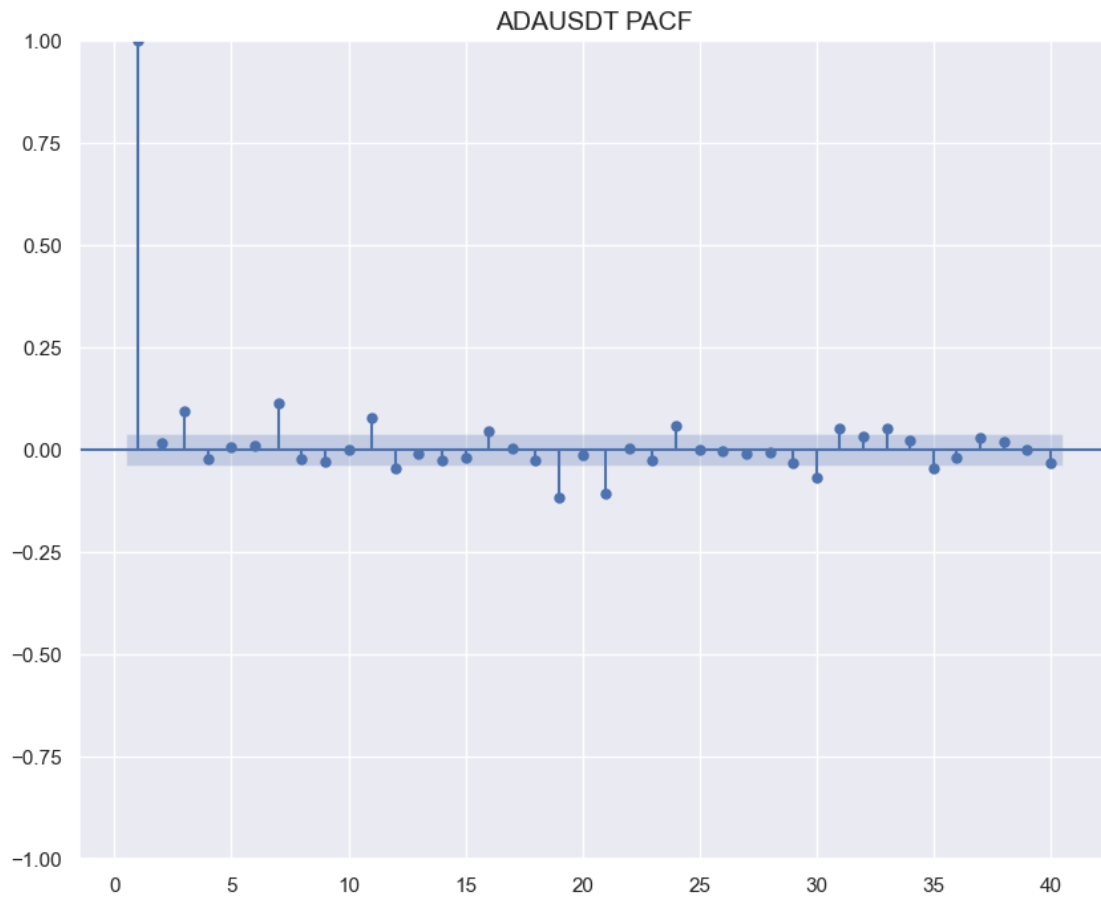
```
[ ]: def pacf_plot(df, column):
    fig, ax = plt.subplots(figsize=(10, 8))
    sgt.plot_pacf(df[column], lags=40, zero=False, ax=ax, method='ols')
    plt.title(f"{column} PACF", size=14)

    # Generate the file name based on the column name
    file_name = f"{column}_pacf.png"

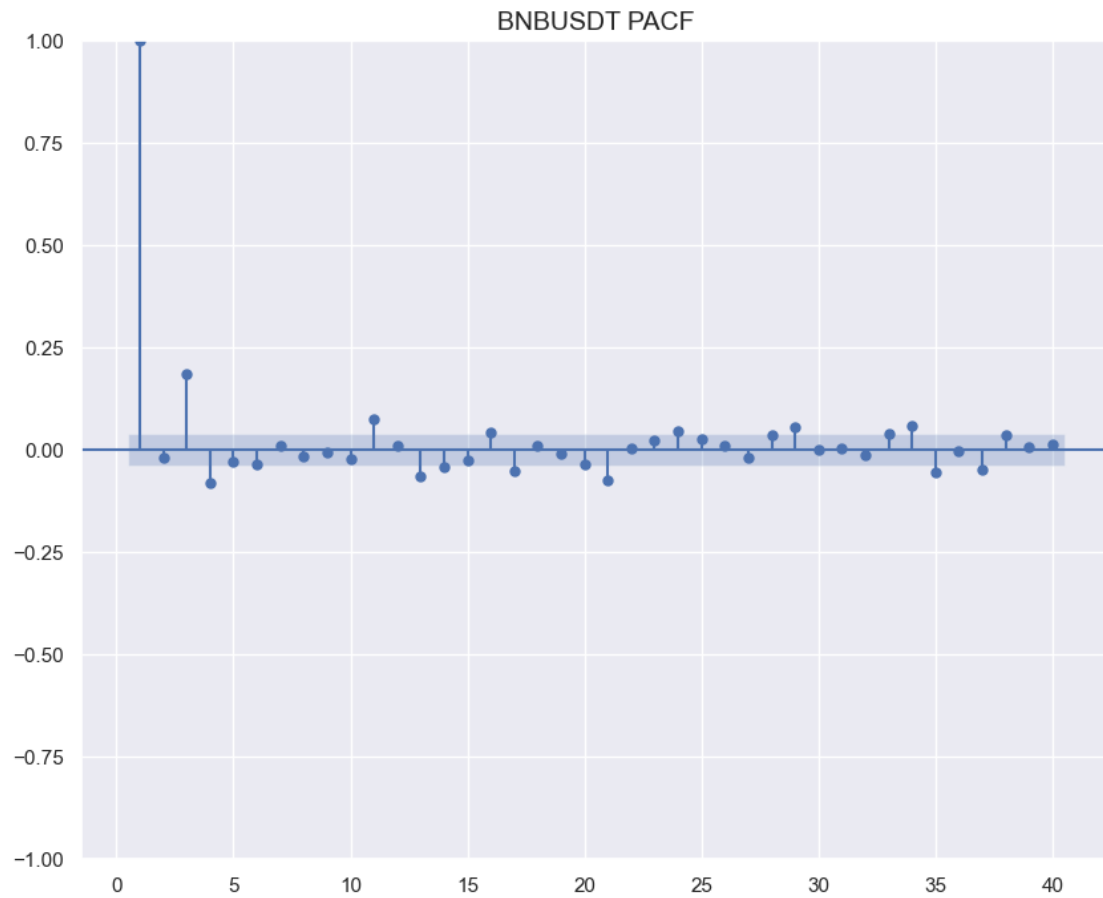
    # Save the figure in the specified directory
```

```
plt.savefig(f'../reports/figures/{file_name}_pacf.png')  
plt.show()
```

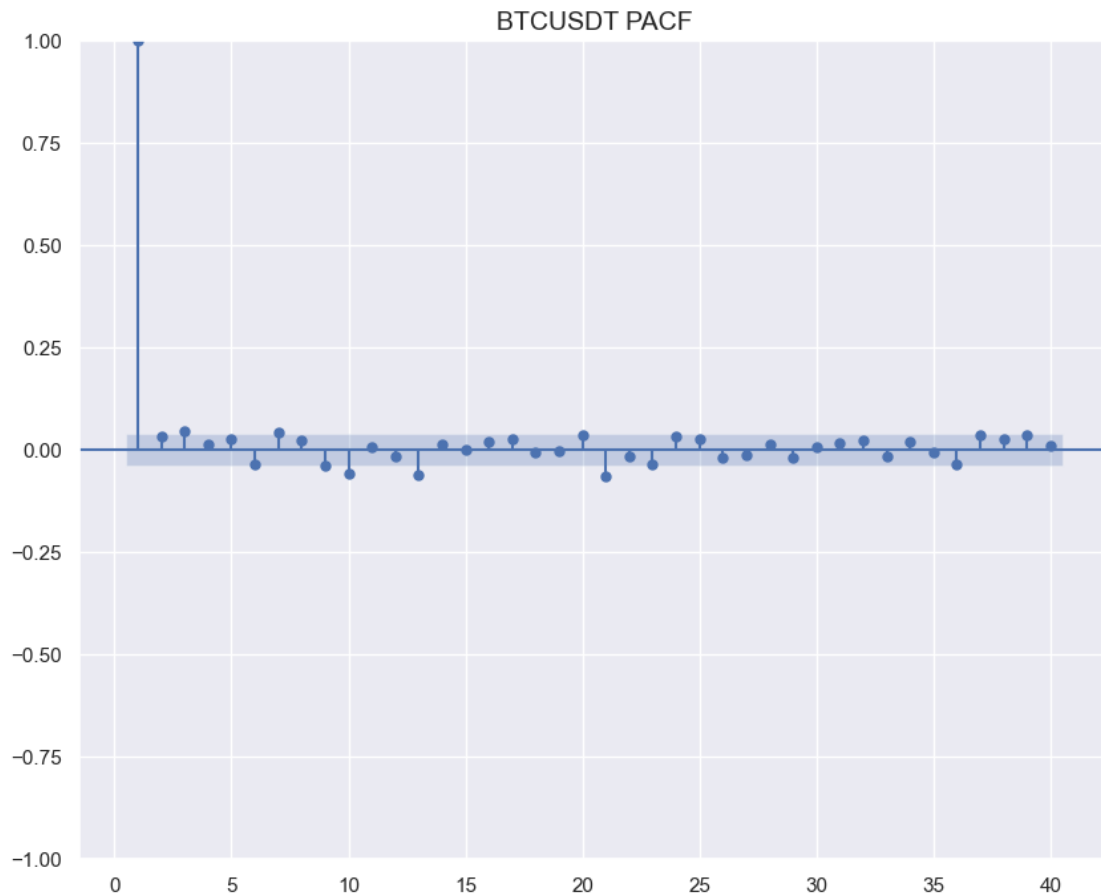
```
[ ]: pacf_plot(df, 'ADAUSDT')
```



```
[ ]: pacf_plot(df, 'BNBUSDT')
```



```
[ ]: pacf_plot(df, 'BTCUSDT')
```



0.5.1 the pacf shows that the previous lags has little to no effect on the current price

```
[ ]: def plot_correlation_heatmap(df, columns):
    selected_df = df[columns]

    # Compute the correlation matrix
    corr_matrix = selected_df.corr()

    # Create the heatmap plot
    plt.figure(figsize=(10, 8))
    sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')

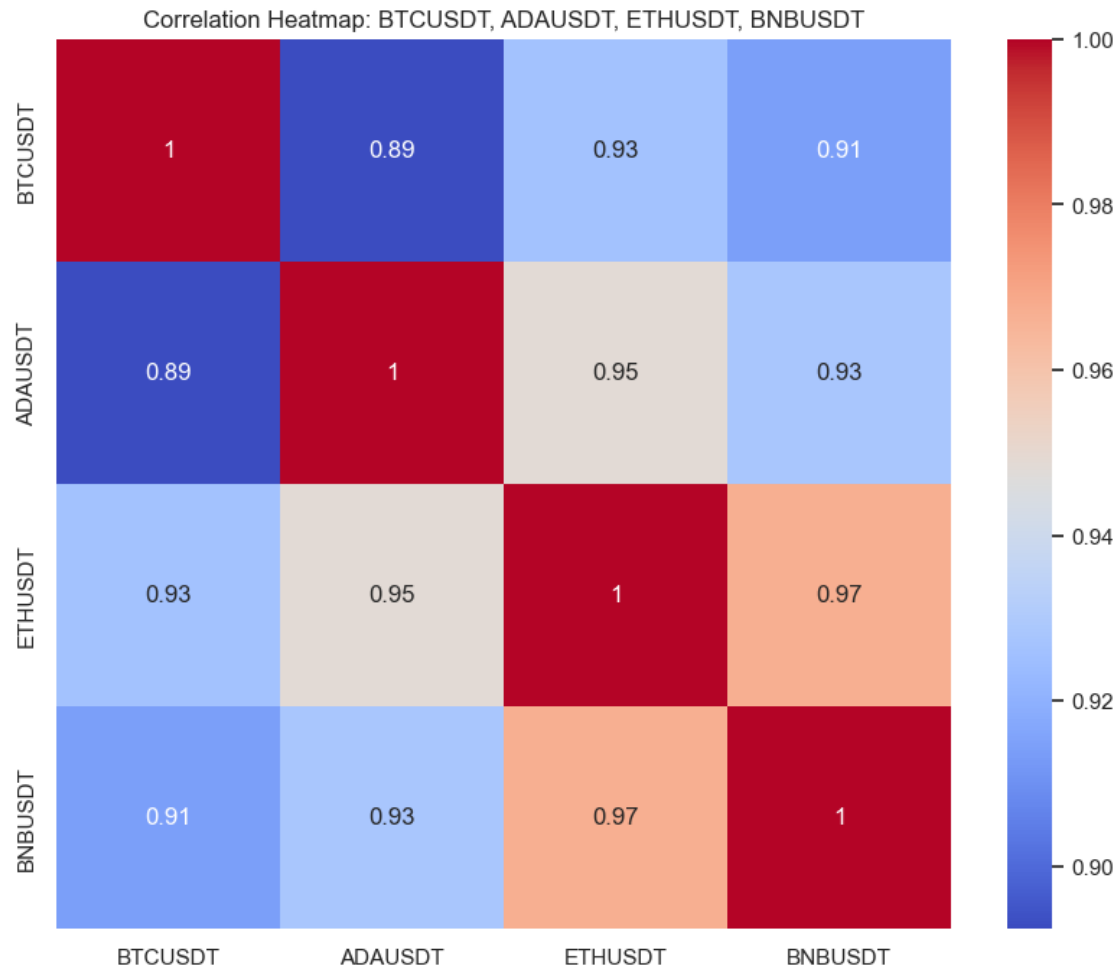
    # Set the dynamic title based on the column names
    title = "Correlation Heatmap: " + ', '.join(columns)
    plt.title(title)

    file_name = "correlation_heatmap_" + '_'.join(columns) + ".png"
```

```
plt.savefig(f'../reports/figures/{file_name}')

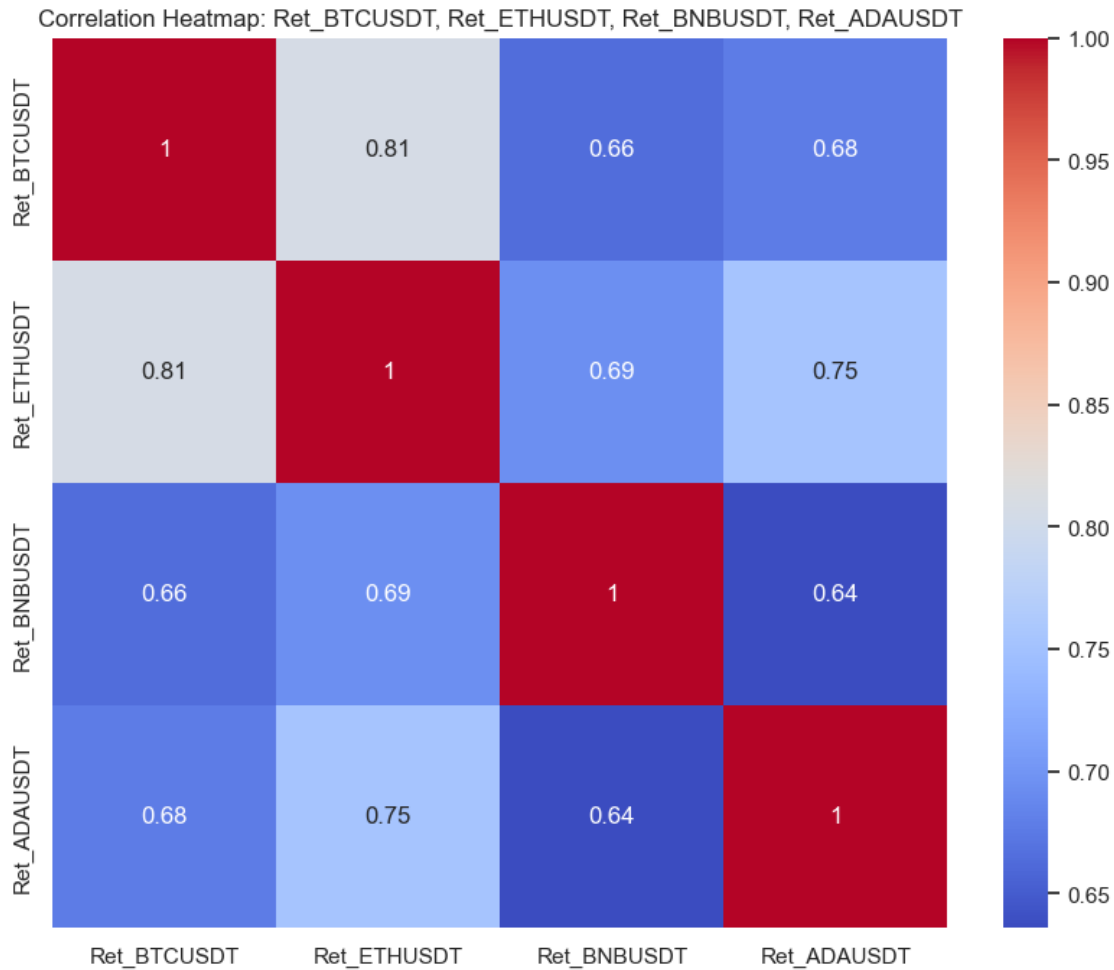
plt.show()
```

```
[ ]: plot_correlation_heatmap(df, ['BTCUSDT', 'ADAUSDT', 'ETHUSDT', 'BNBUSDT'])
```



This heatmap shows that the cryptocurrencies in view are highly correlated with each other, which is not unexpected. Looking closely, ADA and BTC are the least correlated.

```
[ ]: plot_correlation_heatmap(df, ['Ret_BTCUSDT', 'Ret_ETHUSDT', 'Ret_BNBUSDT', 'Ret_ADAUSDT'])
```



```
[ ]: def outlier_detection(df, target_column, contamination=0.1):

    model = IsolationForest(contamination=contamination, random_state=42)

    # Fit the model and predict outliers
    df.dropna(inplace = True)
    df[f'outr_{target_column}'] = model.fit_predict(df[[target_column]])
```

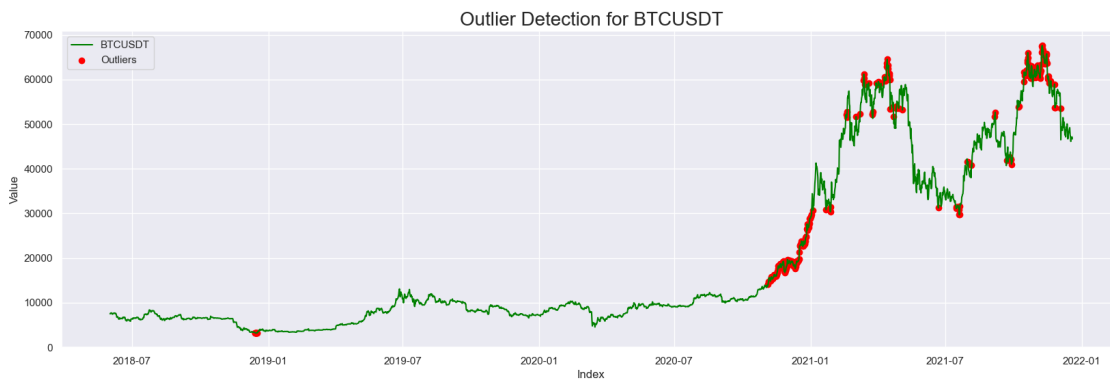
```
[ ]: outlier_detection(df, 'BTCUSDT')
outlier_detection(df, 'BNBUSDT')
outlier_detection(df, 'ETHUSDT')
outlier_detection(df, 'ADAUSDT')
```

```
[ ]: #for returns
outlier_detection(df, 'Ret_cum_BTCUSDT')
outlier_detection(df, 'Ret_cum_ADAUSDT')
outlier_detection(df, 'Ret_cum_ETHUSDT')
```

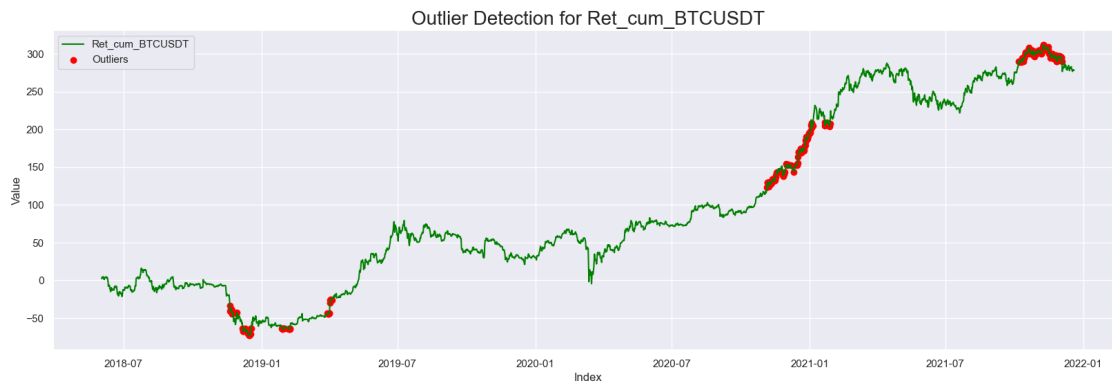
```
outlier_detection(df, 'Ret_cum_BNBUSDT')
```

```
[ ]: def plot_outliers(df, target_column):  
  
    fig, ax = plt.subplots(figsize=(20, 6))  
  
    # Plot the data  
    ax.plot(df.index, df[target_column], label= target_column, c = 'green')  
  
    # Mark the outliers  
    outliers = df[df[f'outr_{target_column}'] == -1]  
    ax.scatter(outliers.index, outliers[target_column], color='red',  
              ↪label='Outliers')  
  
    ax.set_xlabel('Index')  
    ax.set_ylabel('Value')  
    ax.set_title(f'Outlier Detection for {target_column}', fontsize = 20)  
    ax.legend()  
    plt.savefig(f'../reports/figures/{target_column}_outliers.png')  
    plt.show()
```

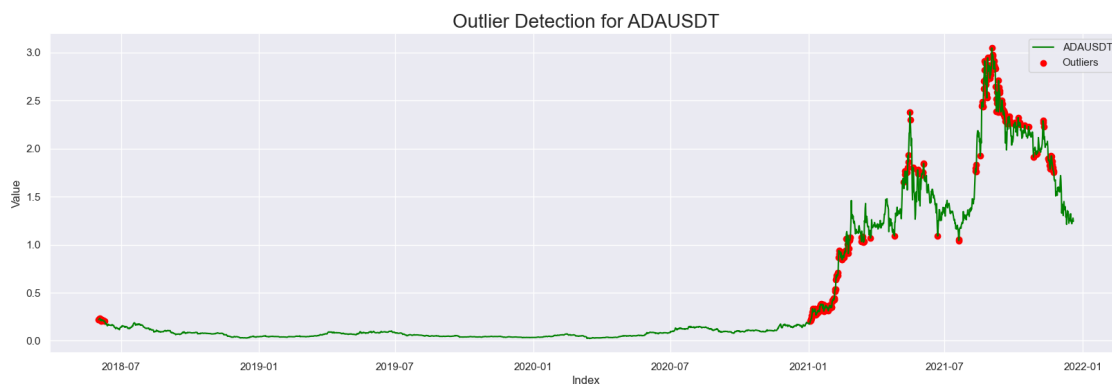
```
[ ]: plot_outliers(df, 'BTCUSDT')
```



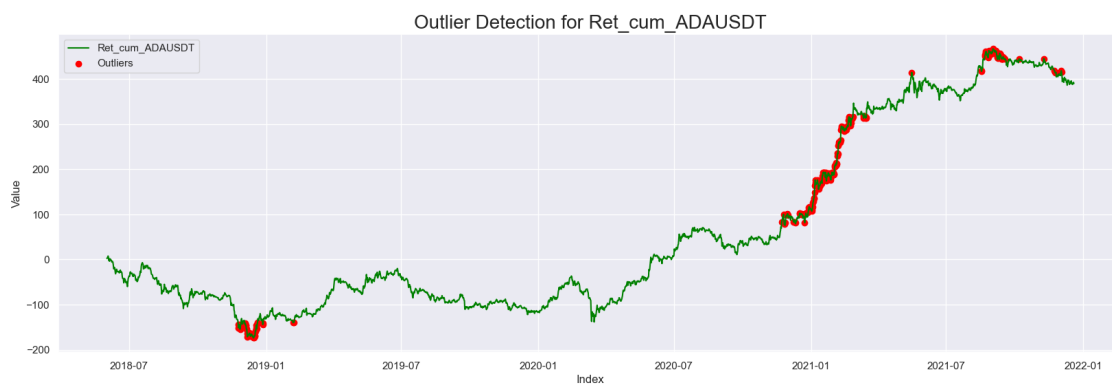
```
[ ]: plot_outliers(df, 'Ret_cum_BTCUSDT')
```



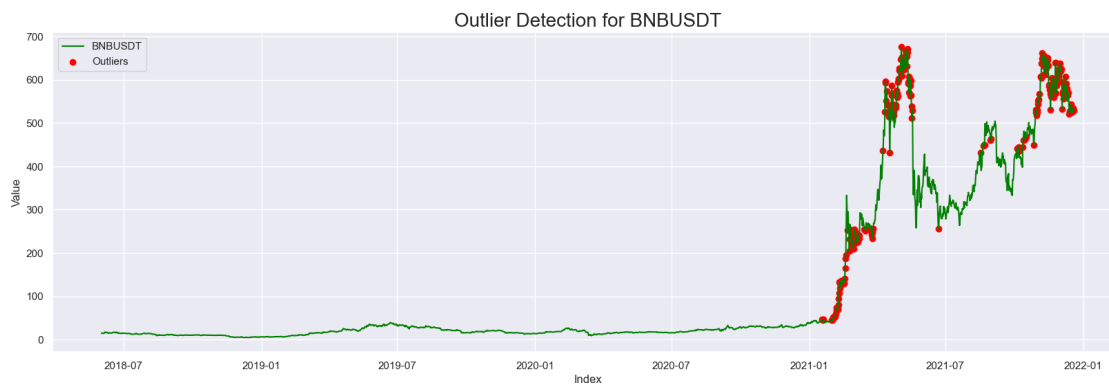
```
[ ]: plot_outliers(df, 'ADAUSDT')
```



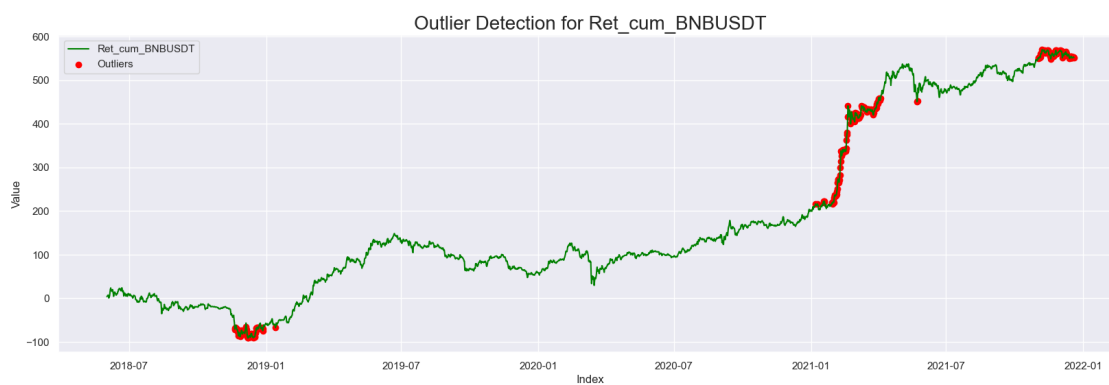
```
[ ]: plot_outliers(df, 'Ret_cum_ADAUSDT')
```



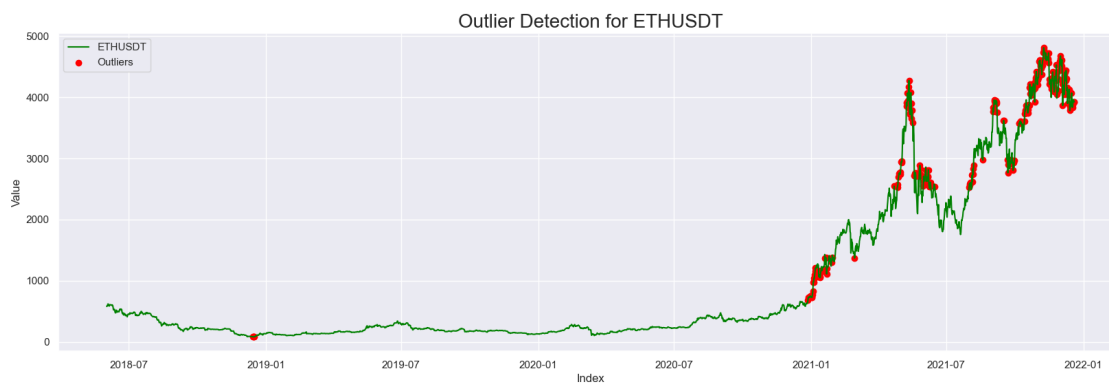
```
[ ]: plot_outliers(df, 'BNBUSDT')
```



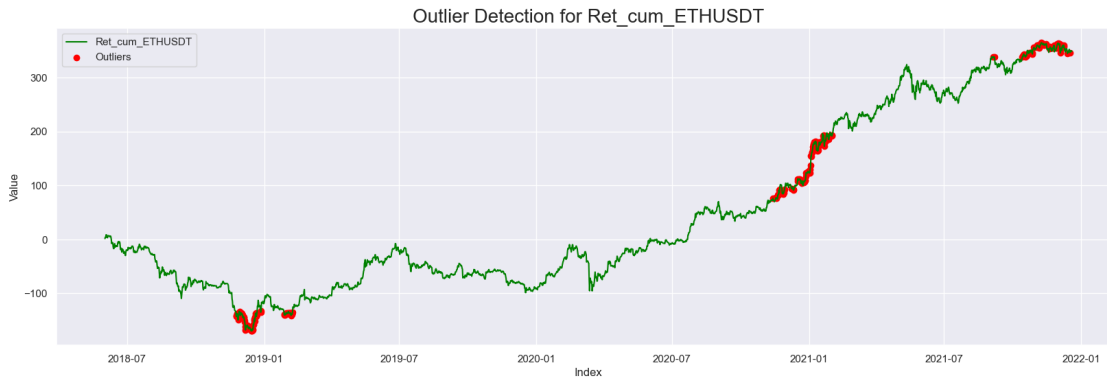
```
[ ]: plot_outliers(df, 'Ret_cum_BNBUSDT')
```



```
[ ]: plot_outliers(df, 'ETHUSDT')
```



```
[ ]: plot_outliers(df, 'Ret_cum_ETHUSDT')
```



Analysing volatility

```
[ ]: def garch_volatility(df, column):
    """
    Fit and plot the GARCH model's estimated volatility.

    Parameters:
        df (DataFrame): DataFrame containing the return series.
        column (str): Name of the column with the return series.

    """
    # Create a GARCH model
    model = arch.arch_model(df[f'Ret_{column}'], vol='Garch', p=1, q=1)

    # Fit the model
    results = model.fit(update_freq=5)

    # Print the model summary
    print(results.summary())

    # Plot the estimated volatility
    fig, ax = plt.subplots(figsize=(20, 6))
    ax.plot(results.conditional_volatility, color='blue', label=f'Estimated_
↳Volatility for {column}')
    ax.set_xlabel('Time', fontsize = 15)
    ax.set_ylabel('Volatility', fontsize = 17)
    ax.set_title(f'GARCH Estimated Volatility {column}', fontsize = 20)
    ax.legend()
    plt.savefig(f'../reports/figures/Ret_{column}_garch.png')
    plt.show()
```

```
[ ]: garch_volatility(df, 'BTCUSD')
```

Iteration: 5, Func. Count: 41, Neg. LLF: 6048.300257481001

```

Iteration:      10,  Func. Count:      65,  Neg. LLF: 6048.1191191605285
Optimization terminated successfully (Exit mode 0)
Current function value: 6048.1191191622565
Iterations: 10
Function evaluations: 65
Gradient evaluations: 10

```

Constant Mean - GARCH Model Results

```

=====
Dep. Variable:      Ret_BTCUSDT      R-squared:      0.000
Mean Model:      Constant Mean      Adj. R-squared:      0.000
Vol Model:      GARCH      Log-Likelihood:      -6048.12
Distribution:      Normal      AIC:      12104.2
Method:      Maximum Likelihood      BIC:      12127.7
No. Observations:      2595
Date:      Thu, Jun 29 2023      Df Residuals:      2594
Time:      12:14:13      Df Model:      1

```

Mean Model

```

=====
              coef      std err          t      P>|t|      95.0% Conf. Int.
-----
mu              0.1030   4.420e-02      2.330   1.982e-02   [1.635e-02,  0.190]

```

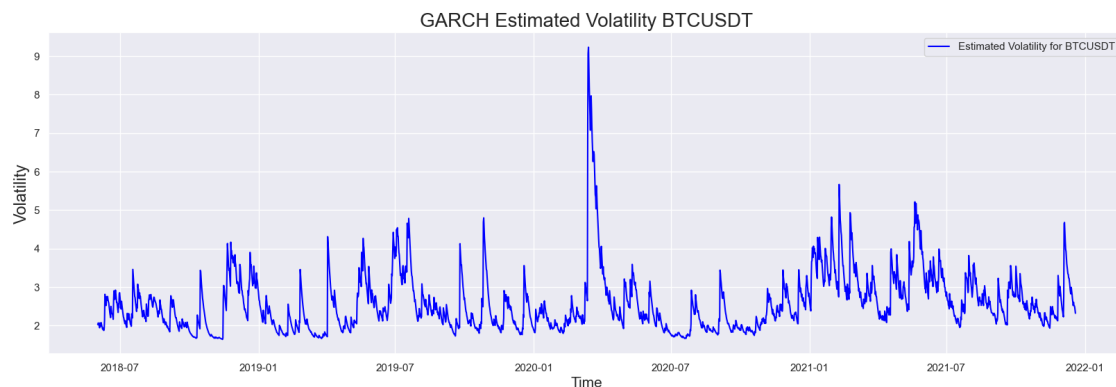
Volatility Model

```

=====
              coef      std err          t      P>|t|      95.0% Conf. Int.
-----
omega           0.3088      0.120      2.582   9.809e-03   [7.445e-02,  0.543]
alpha[1]         0.0819   2.433e-02      3.368   7.580e-04   [3.425e-02,  0.130]
beta[1]           0.8780   2.879e-02     30.496   2.901e-204   [ 0.822,  0.934]

```

Covariance estimator: robust



```
[ ]: garch_volatility(df, 'ADAUSDT')
```

Iteration: 5, Func. Count: 32, Neg. LLF: 7377.427459970775
 Iteration: 10, Func. Count: 60, Neg. LLF: 7158.610315989126
 Optimization terminated successfully (Exit mode 0)
 Current function value: 7158.610273854514
 Iterations: 13
 Function evaluations: 74
 Gradient evaluations: 13

Constant Mean - GARCH Model Results

```
=====
Dep. Variable:      Ret_ADAUSD   R-squared:          0.000
Mean Model:        Constant Mean  Adj. R-squared:     0.000
Vol Model:         GARCH          Log-Likelihood:    -7158.61
Distribution:      Normal         AIC:              14325.2
Method:           Maximum Likelihood  BIC:             14348.7
                                     No. Observations:  2595
Date:             Thu, Jun 29 2023  Df Residuals:         2594
Time:             12:14:16          Df Model:           1
=====
```

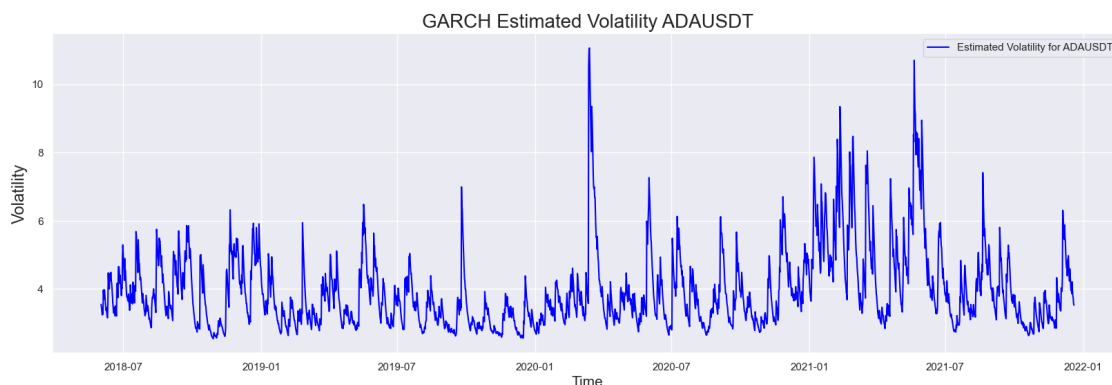
Mean Model

```
=====
              coef      std err          t      P>|t|     95.0% Conf. Int.
-----
mu           0.0649   6.973e-02     0.931     0.352  [-7.174e-02,  0.202]
=====
```

Volatility Model

```
=====
              coef      std err          t      P>|t|     95.0% Conf. Int.
-----
omega        0.8997     0.261       3.444  5.738e-04  [ 0.388,  1.412]
alpha[1]     0.1080    1.939e-02     5.573  2.500e-08  [7.005e-02,  0.146]
beta[1]       0.8420    2.787e-02    30.210  1.758e-200  [ 0.787,  0.897]
=====
```

Covariance estimator: robust



```
[ ]: garch_volatility(df, 'BNBUSDT')
```

Iteration: 5, Func. Count: 37, Neg. LLF: 6793.254751741339

Iteration: 10, Func. Count: 62, Neg. LLF: 6793.127806352441

Optimization terminated successfully (Exit mode 0)

Current function value: 6793.127806351044

Iterations: 10

Function evaluations: 62

Gradient evaluations: 10

Constant Mean - GARCH Model Results

```
=====
Dep. Variable:      Ret_BNBUSDT      R-squared:      0.000
Mean Model:        Constant Mean    Adj. R-squared: 0.000
Vol Model:         GARCH            Log-Likelihood: -6793.13
Distribution:      Normal           AIC:           13594.3
Method:           Maximum Likelihood BIC:           13617.7
                                           No. Observations: 2595
Date:             Thu, Jun 29 2023   Df Residuals:   2594
Time:             12:14:19           Df Model:       1
=====
```

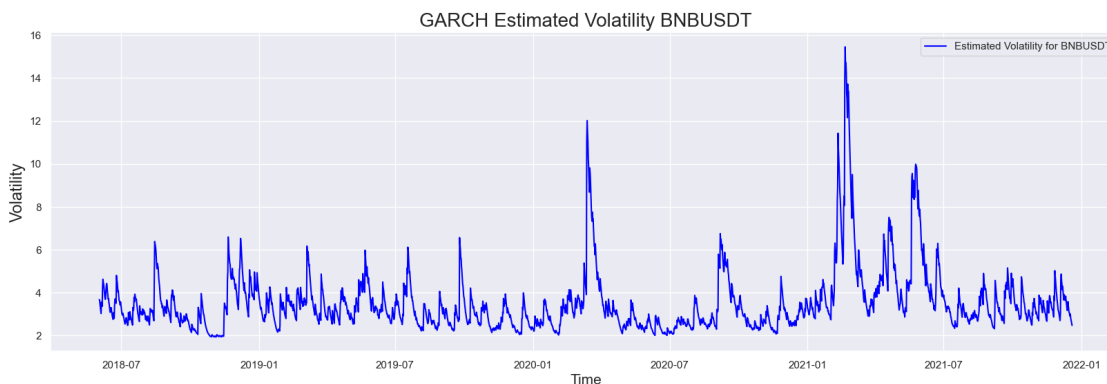
Mean Model

```
=====
              coef      std err          t      P>|t|      95.0% Conf. Int.
-----
mu           0.1472   5.699e-02      2.584   9.780e-03 [3.553e-02, 0.259]
=====
```

Volatility Model

```
=====
              coef      std err          t      P>|t|      95.0% Conf. Int.
-----
omega        0.4565      0.145        3.146   1.655e-03 [ 0.172, 0.741]
alpha[1]     0.1098   1.864e-02      5.891   3.840e-09 [7.328e-02, 0.146]
beta[1]       0.8605   2.174e-02     39.580   0.000      [ 0.818, 0.903]
=====
```

Covariance estimator: robust



```
[ ]: garch_volatility(df, 'ETHUSDT')
```

Iteration: 5, Func. Count: 32, Neg. LLF: 704651870.7921174

Iteration: 10, Func. Count: 60, Neg. LLF: 6714.70195237444

Optimization terminated successfully (Exit mode 0)

Current function value: 6714.701888691417

Iterations: 13

Function evaluations: 74

Gradient evaluations: 13

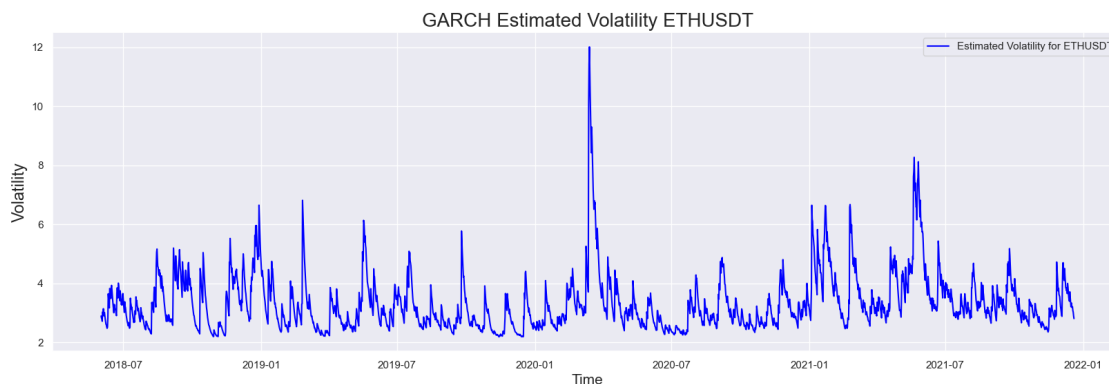
Constant Mean - GARCH Model Results

```
=====
Dep. Variable:      Ret_ETHUSDT    R-squared:      0.000
Mean Model:        Constant Mean  Adj. R-squared:  0.000
Vol Model:         GARCH          Log-Likelihood: -6714.70
Distribution:      Normal         AIC:            13437.4
Method:           Maximum Likelihood BIC:           13460.8
                                           No. Observations: 2595
Date:             Thu, Jun 29 2023 Df Residuals:      2594
Time:             12:14:23         Df Model:         1
                                           Mean Model
=====
```

```
=====
              coef    std err          t      P>|t|      95.0% Conf. Int.
-----
mu           0.1272   5.827e-02      2.183   2.902e-02 [1.301e-02,  0.241]
              Volatility Model
=====
```

```
=====
              coef    std err          t      P>|t|      95.0% Conf. Int.
-----
omega        0.6556     0.216       3.035   2.406e-03 [ 0.232,  1.079]
alpha[1]     0.0987   2.179e-02      4.530   5.890e-06 [5.600e-02,  0.141]
beta[1]      0.8487   3.186e-02     26.636   2.599e-156 [ 0.786,  0.911]
=====
```

Covariance estimator: robust



3.0-prof-data-modeling

June 29, 2023

```
[ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import statsmodels
import seaborn as sns
import scipy.stats as ss
import statsmodels.graphics.tsaplots as sgt
import statsmodels.tsa.stattools as sts
import sklearn
import arch
import datetime as dt

from pmdarima.arima import auto_arima
from statsmodels.tsa.stattools import adfuller
from statsmodels.stats.diagnostic import acorr_ljungbox, het_arch
from statsmodels.stats.stattools import jarque_bera
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from tqdm import tqdm
from time import sleep
import warnings
warnings.filterwarnings('ignore', category=Warning)
sns.set()
```

```
[ ]: def read_csv(name:str):
    df = pd.read_csv(f'../data/processed/{name}.csv')
    df.Date = pd.to_datetime(df.Date)
    df.set_index('Date', inplace = True)
    df.asfreq('12H')
    df.dropna(inplace = True)
    return df
```

```
[ ]: df= read_csv('data_processed')
df_test = read_csv('data_test')
```

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

```
[ ]: BTCUSDT ETHUSDT ADAUSDT BNBUSDT Ret_BTCUSDT
Date
2018-06-01 12:00:00 7521.01 579.00 0.22038 14.2888 2.168340 \
2018-06-02 00:00:00 7652.28 592.73 0.22428 14.5000 1.745377
2018-06-02 12:00:00 7640.03 590.85 0.22648 14.6732 -0.160083
2018-06-03 00:00:00 7714.85 619.93 0.23284 14.7861 0.979316
2018-06-03 12:00:00 7714.26 619.66 0.22659 14.6995 -0.007648
```

```
Ret_cum_BTCUSDT Norm_BTCUSDT Ret_ETHUSDT
Date
2018-06-01 12:00:00 2.168340 102.168340 1.466800 \
2018-06-02 00:00:00 3.913718 103.951563 2.371330
2018-06-02 12:00:00 3.753635 103.785155 -0.317176
2018-06-03 00:00:00 4.732950 104.801539 4.921723
2018-06-03 12:00:00 4.725303 104.793524 -0.043553
```

```
Ret_cum_ETHUSDT Norm_ETHUSDT Ret_ADAUSDT
Date
2018-06-01 12:00:00 1.466800 101.466800 2.122335 \
2018-06-02 00:00:00 3.838130 103.872912 1.769671
2018-06-02 12:00:00 3.520953 103.543452 0.980917
2018-06-03 00:00:00 8.442676 108.639574 2.808195
2018-06-03 12:00:00 8.399123 108.592258 -2.684247
```

```
Ret_cum_ADAUSDT Norm_ADAUSDT Ret_BNBUSDT
Date
2018-06-01 12:00:00 2.122335 102.122335 3.692308 \
2018-06-02 00:00:00 3.892006 103.929564 1.478081
2018-06-02 12:00:00 4.872923 104.949027 1.194483
2018-06-03 00:00:00 7.681118 107.896200 0.769430
2018-06-03 12:00:00 4.996871 105.000000 -0.585685
```

```
Ret_cum_BNBUSDT Norm_BNBUSDT
Date
2018-06-01 12:00:00 3.692308 103.692308
2018-06-02 00:00:00 5.170388 105.224964
2018-06-02 12:00:00 6.364871 106.481858
2018-06-03 00:00:00 7.134301 107.301161
2018-06-03 12:00:00 6.548616 106.672714
```

```
[ ]: def fit_auto(df, target, exogenous=None, exog=True, **kwargs):
    if exog:
        model = auto_arima(df[[f'Ret_{target}']], X=df[exogenous],
                           max_order = None, max_p = 9, max_q = 9, max_d = 2, max_P=
↪ 6, max_Q = 6, max_D = 4,
                           maxiter = 70, trend = 'ct')
        print(model.summary())
```

```

else:
    model = auto_arima(df[[f'Ret_{target}']],
                      max_order = None, max_p = 9, max_q = 9, max_d = 2, max_P=
↪ 6, max_Q = 6, max_D = 4,
                      maxiter = 70, trend = 'ct')
    print(model.summary())

return model

```

```

[ ]: start_date = '2021-12-18 12:00:00'
end_date='2022-06-01 12:00:00'

def fit_pred(model, target, exogenous: list = None, start_date = start_date,
↪end_date= end_date, df_test=df_test, exog=True):
    if exog:
        df_auto_pred = pd.DataFrame(
            model.predict(n_periods=len(df_test[start_date:end_date]),
↪X=df_test[exogenous][start_date:end_date]),
            index=df_test[start_date:end_date].index)
    else:
        df_auto_pred = pd.DataFrame(
            model.predict(n_periods=len(df_test[start_date:end_date]),
↪index=df_test[start_date:end_date].index))

    plt.figure(figsize=(20, 5))
    plt.plot(df_auto_pred, color="red", label="Auto Model Predictions")
    plt.plot(df_test[f'Ret_{target}'][start_date:end_date], color="blue",
↪label="Real Data")
    plt.title(f"Auto Model Predictions vs Real Data for {target}", size=24)
    plt.xlabel("Date", size=16)
    plt.ylabel("Return", size=16)
    plt.legend(fontsize=14)
    plt.savefig(f'../reports/figures/{target}_prediction_vs_real.png')
    plt.show()

```

```

[ ]: def analyze_residuals(model, name:str):
    residuals_df = pd.DataFrame(model.resid())
    print(residuals_df.describe())

    residuals = model.resid()
    lags = [1]
    # Perform Ljung-Box test
    lb_test = acorr_ljungbox(residuals, lags=lags, return_df=True)
    lb_p_value = lb_test.iloc[0, 1]

    # Perform Jarque-Bera test
    jb_stat, jb_p_value, skew, kurtosis = jarque_bera(residuals)

```

```

# Perform heteroskedasticity test
het_test = het_arch(residuals)
het_p_value = het_test[1]

# Perform ADF test
adf_test = adfuller(residuals)
adf_p_value = round(adf_test[1], 3)

# Print summary of analysis
print("Residual Analysis:")
print("====="*2)
print(f"\nLjung-Box (lag 1) p-value: {lb_p_value}")
print(f"\nJarque-Bera p-value: {jb_p_value}")
print(f"\nHeteroskedasticity p-value: {het_p_value}")
print(f"\nADF p-value: {adf_p_value}")
print("====="*2)

print('\n')

# Autocorrelation plot
fig, ax = plt.subplots(figsize=(7, 4))
plot_acf(residuals_df, lags=30, ax=ax)
plt.title(f'Autocorrelation of Residuals_{name}')
plt.xlabel('Lag')
plt.ylabel('Autocorrelation')
plt.savefig(f'../reports/figures/{name}_resid_acf.png')
plt.show()

# Partial autocorrelation plot
fig, ax = plt.subplots(figsize=(7, 4))
plot_pacf(residuals_df, lags=30, ax=ax)
plt.title(f'Partial Autocorrelation of Residuals_{name}')
plt.xlabel('Lag')
plt.ylabel('Partial Autocorrelation')
plt.savefig(f'../reports/figures/{name}_resid_pacf.png')
plt.show()
print('\n')
model.plot_diagnostics(figsize=(20, 11))
plt.savefig(f'../reports/figures/{name}_diagnostics.png')
plt.show()

```

```
[ ]:
```

```
[ ]: model_arima_bnb = fit_auto(df, 'BNBUSDT', exog=False)
```

SARIMAX Results

```

=====
Dep. Variable:          y      No. Observations:      2592
Model:                SARIMAX(0, 0, 2)      Log Likelihood      -7090.591
Date:                Thu, 29 Jun 2023      AIC      14191.182
Time:                12:10:18      BIC      14220.483
Sample:                06-01-2018      HQIC      14201.800
                    - 12-18-2021

```

Covariance Type: opg

```

=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
intercept    -0.0163      0.170     -0.096      0.924     -0.350      0.317
drift         0.0002      0.000      1.704      0.088     -2.63e-05      0.000
ma.L1         0.1337      0.010     13.288      0.000      0.114      0.153
ma.L2        -0.1164      0.013     -9.114      0.000     -0.141     -0.091
sigma2       13.9420      0.172     80.994      0.000     13.605     14.279
=====

```

```

===
Ljung-Box (L1) (Q):          0.01      Jarque-Bera (JB):
10263.80
Prob(Q):                    0.93      Prob(JB):
0.00
Heteroskedasticity (H):      1.92      Skew:
0.18
Prob(H) (two-sided):        0.00      Kurtosis:
12.74
=====

```

Warnings:

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

```
[ ]: analyze_residuals(model_arima_bnb, 'BNBUSDT')
```

```

0
count  2592.000000
mean    0.002685
std     3.731526
min    -31.751767
25%    -1.731326
50%    -0.077065
75%     1.689766
max     35.945056
Residual Analysis:
=====

```

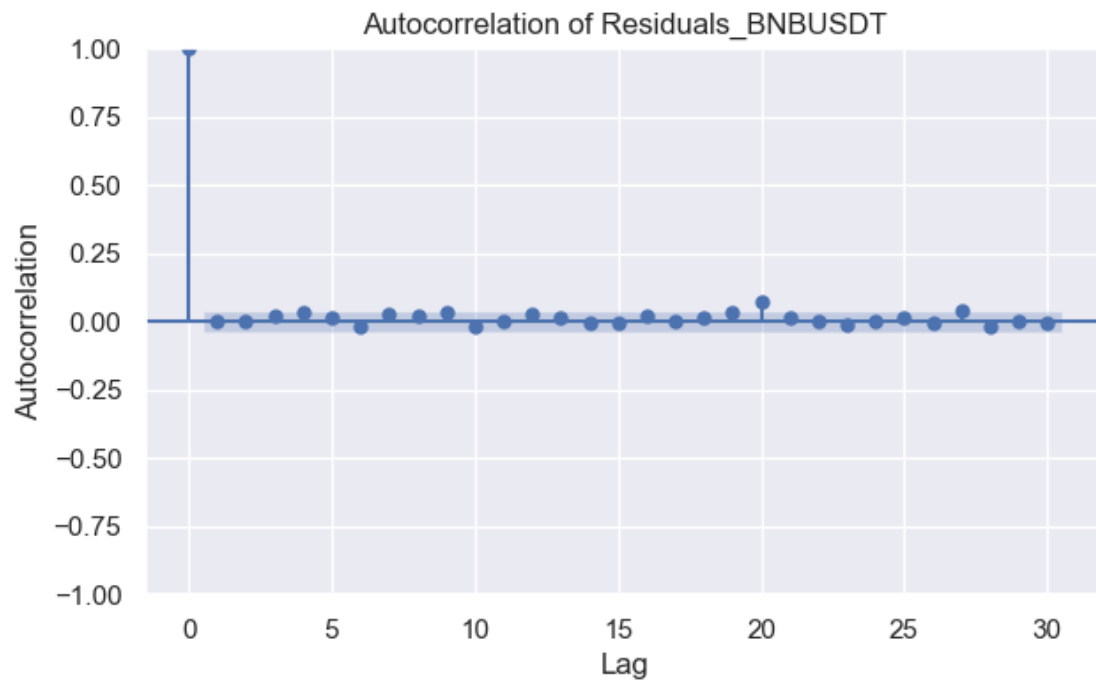
Ljung-Box (lag 1) p-value: 0.9278101753935084

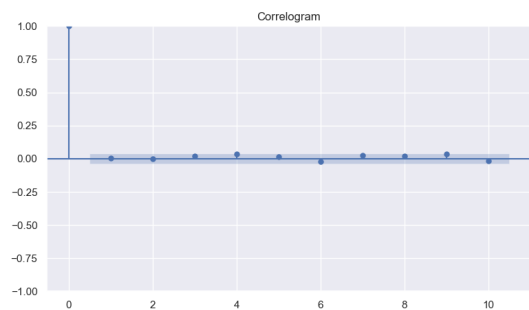
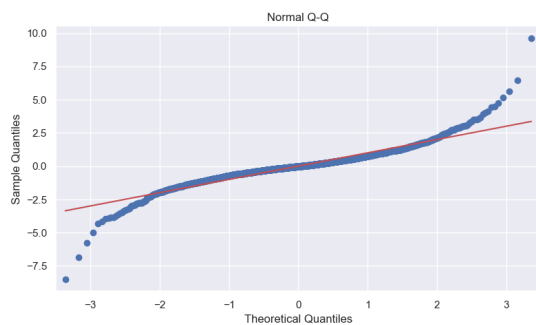
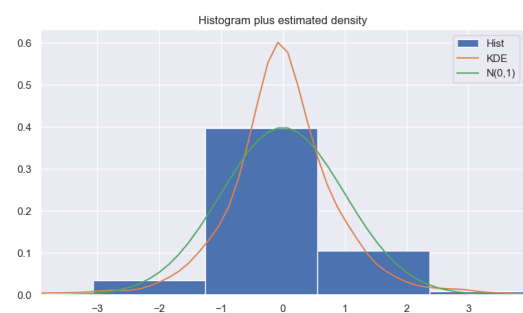
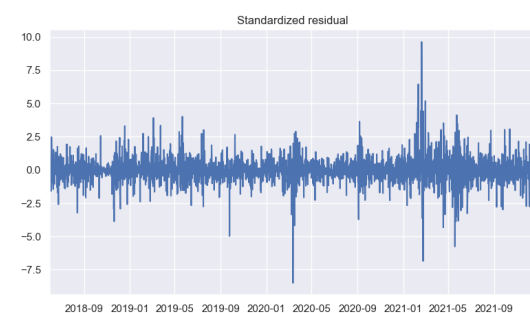
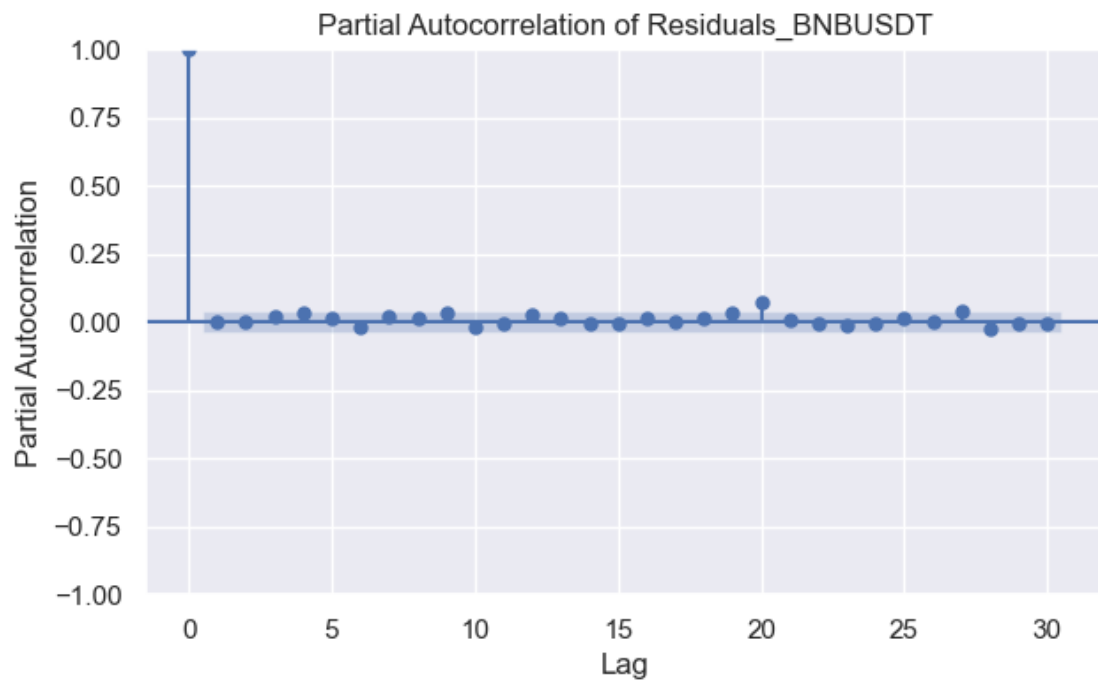
Jarque-Bera p-value: 0.0

Heteroskedasticity p-value: 1.530575495476331e-70

ADF p-value: 0.0

=====





```
[ ]: model_arima_bnb_exog = fit_auto(df, 'BNBUSDT', exogenous= ['Ret_BTCUSDT',
↳ 'Ret_ADAUSDT', 'Ret_ETHUSDT'])
```

SARIMAX Results

```
=====
Dep. Variable:          y      No. Observations:      2592
Model:                SARIMAX(5, 0, 0)  Log Likelihood      -6094.465
Date:                Thu, 29 Jun 2023  AIC              12210.929
Time:                12:13:17    BIC              12275.391
Sample:                06-01-2018    HQIC             12234.289
                        - 12-18-2021
```

Covariance Type: opg

```
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
intercept      0.0631      0.108      0.585      0.558      -0.148      0.274
drift          1.6e-05      7.05e-05      0.227      0.820      -0.000      0.000
Ret_BTCUSDT     0.3600      0.026     13.739      0.000      0.309      0.411
Ret_ADAUSDT     0.2067      0.013     15.763      0.000      0.181      0.232
Ret_ETHUSDT     0.3572      0.024     15.125      0.000      0.311      0.403
ar.L1           0.1757      0.009     19.866      0.000      0.158      0.193
ar.L2          -0.1032      0.012     -8.753      0.000     -0.126     -0.080
ar.L3           0.0552      0.012      4.641      0.000      0.032      0.078
ar.L4           0.0277      0.013      2.111      0.035      0.002      0.053
ar.L5          -0.0388      0.014     -2.774      0.006     -0.066     -0.011
sigma2          6.4615      0.073     88.047      0.000      6.318      6.605
=====
```

```
===
Ljung-Box (L1) (Q):      0.00  Jarque-Bera (JB):
37877.90
Prob(Q):      0.97  Prob(JB):
0.00
Heteroskedasticity (H):      1.46  Skew:
1.97
Prob(H) (two-sided):      0.00  Kurtosis:
21.31
=====
===
```

Warnings:

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

```
[ ]: analyze_residuals(model_arima_bnb_exog, 'BNBUSDT_exog')
```

```
0
count  2592.000000
mean    0.000187
```

std 2.540869
min -13.472579
25% -1.230003
50% -0.198756
75% 0.896538
max 33.653286

Residual Analysis:

=====

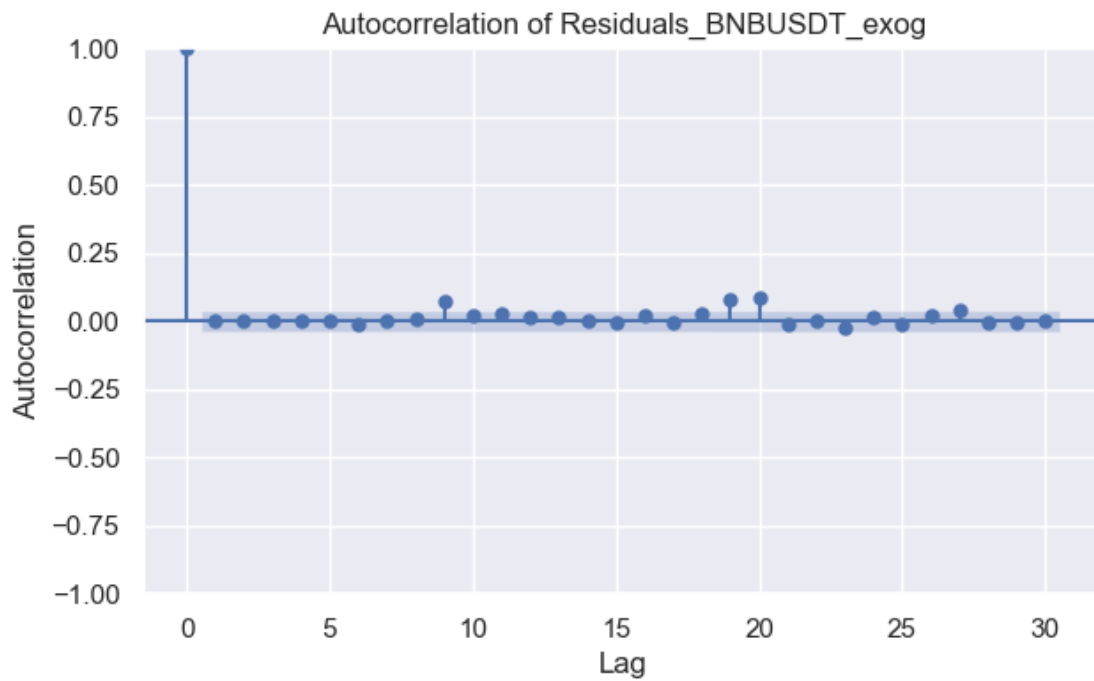
Ljung-Box (lag 1) p-value: 0.9709743058016689

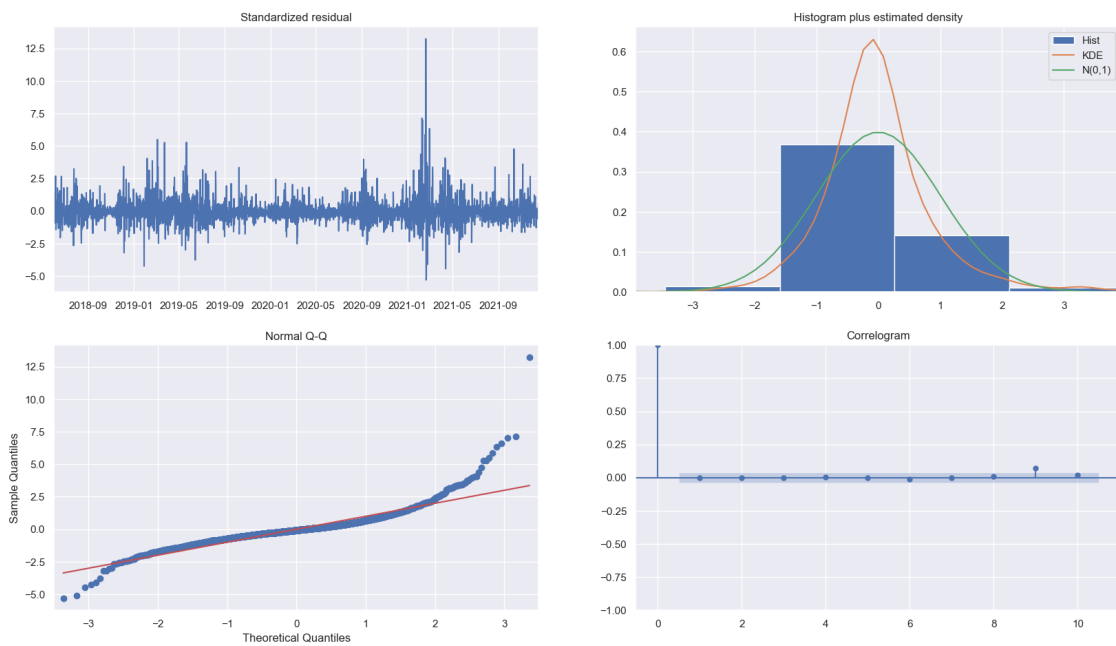
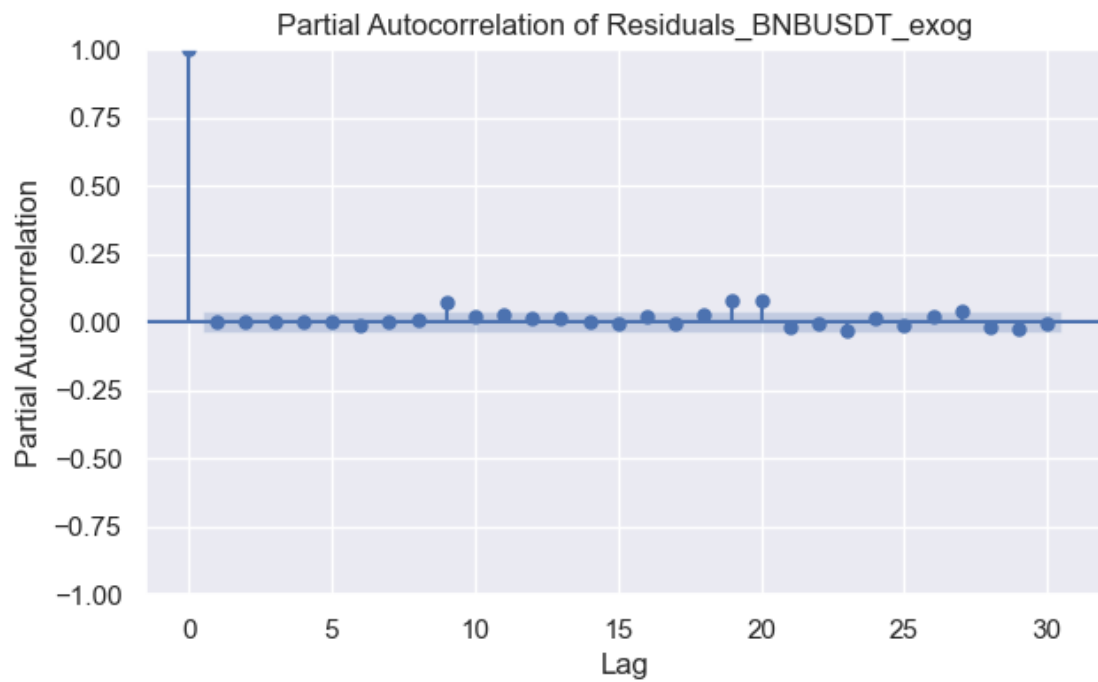
Jarque-Bera p-value: 0.0

Heteroskedasticity p-value: 2.78345921210893e-67

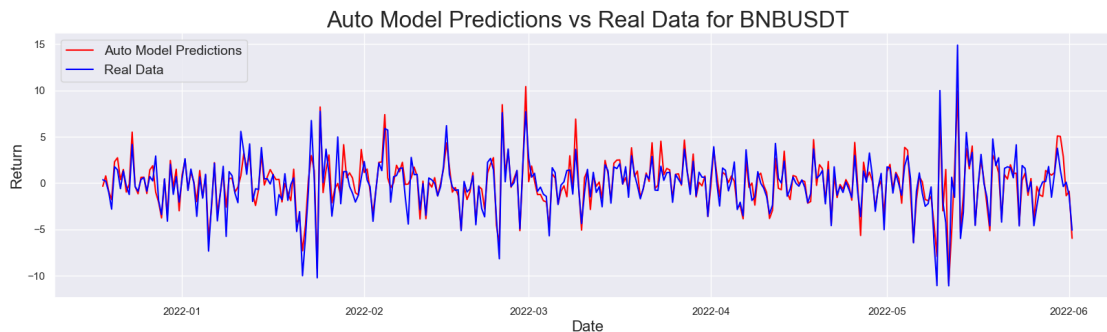
ADF p-value: 0.0

=====





```
[ ]: fit_pred(model_arima_bnb_exog, 'BNBUSDT', exogenous= ['Ret_BTCUSDT',  
↳ 'Ret_ADAUSDT', 'Ret_ETHUSDT'] )
```



```
[ ]: model_arima_btc = fit_auto(df, 'BTCUSDT', exog=False)
```

SARIMAX Results

```
=====
Dep. Variable:                y      No. Observations:          2592
Model:                SARIMAX(0, 0, 2)  Log Likelihood          -6205.835
Date:                Thu, 29 Jun 2023    AIC              12421.670
Time:                12:13:55           BIC              12450.971
Sample:                06-01-2018       HQIC              12432.288
                - 12-18-2021
```

Covariance Type: opg

```
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
intercept      0.0041      0.112      0.037      0.971      -0.215      0.224
drift          7.923e-05    6.94e-05     1.141      0.254     -5.69e-05     0.000
ma.L1          0.0512      0.011     4.688      0.000      0.030      0.073
ma.L2         -0.0915      0.015    -6.130      0.000     -0.121     -0.062
sigma2         7.0487      0.092    76.808      0.000      6.869      7.229
=====
```

```
===
Ljung-Box (L1) (Q):                0.00   Jarque-Bera (JB):
6835.33
Prob(Q):                            0.97   Prob(JB):
0.00
Heteroskedasticity (H):              1.58   Skew:
-0.40
Prob(H) (two-sided):                0.00   Kurtosis:
10.92
=====
===
```

Warnings:

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

```
[ ]: analyze_residuals(model_arima_btc, 'BTCUSDT')
```

```
count  2592.000000
mean    0.000883
std     2.652434
min     -23.776324
25%     -1.039865
50%      0.034647
75%      1.028928
max      17.298787
```

Residual Analysis:

=====

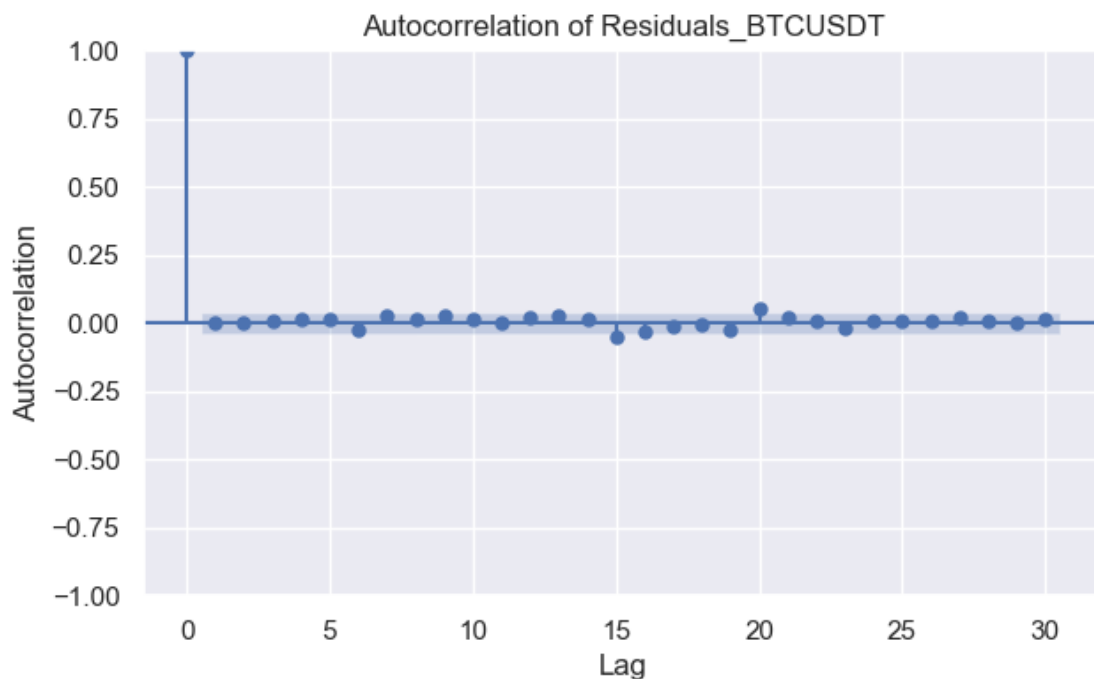
Ljung-Box (lag 1) p-value: 0.9728255007924227

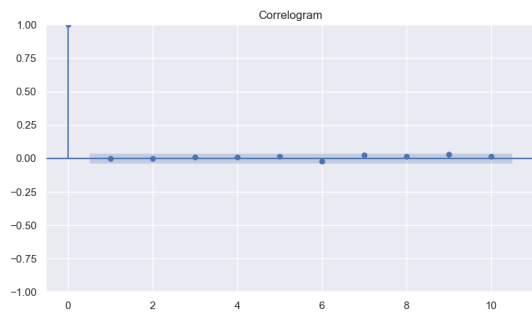
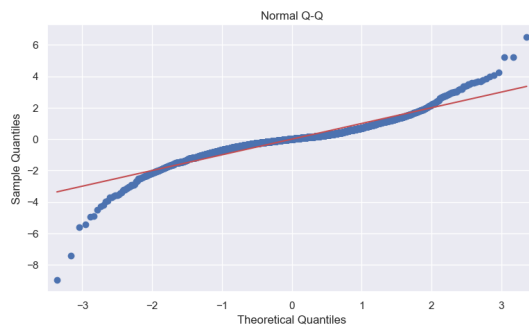
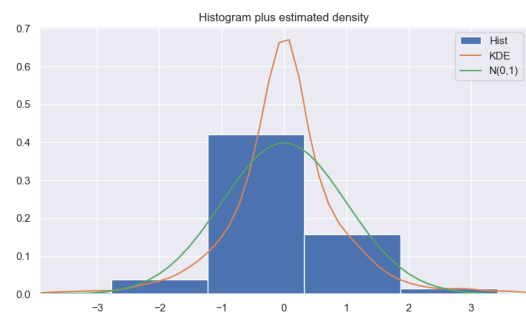
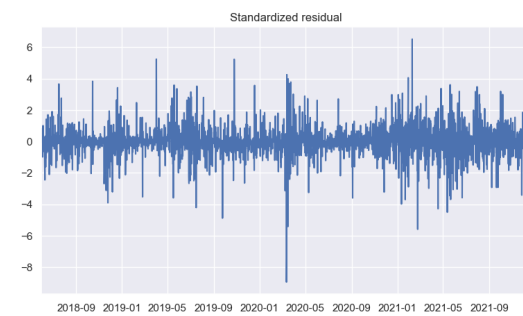
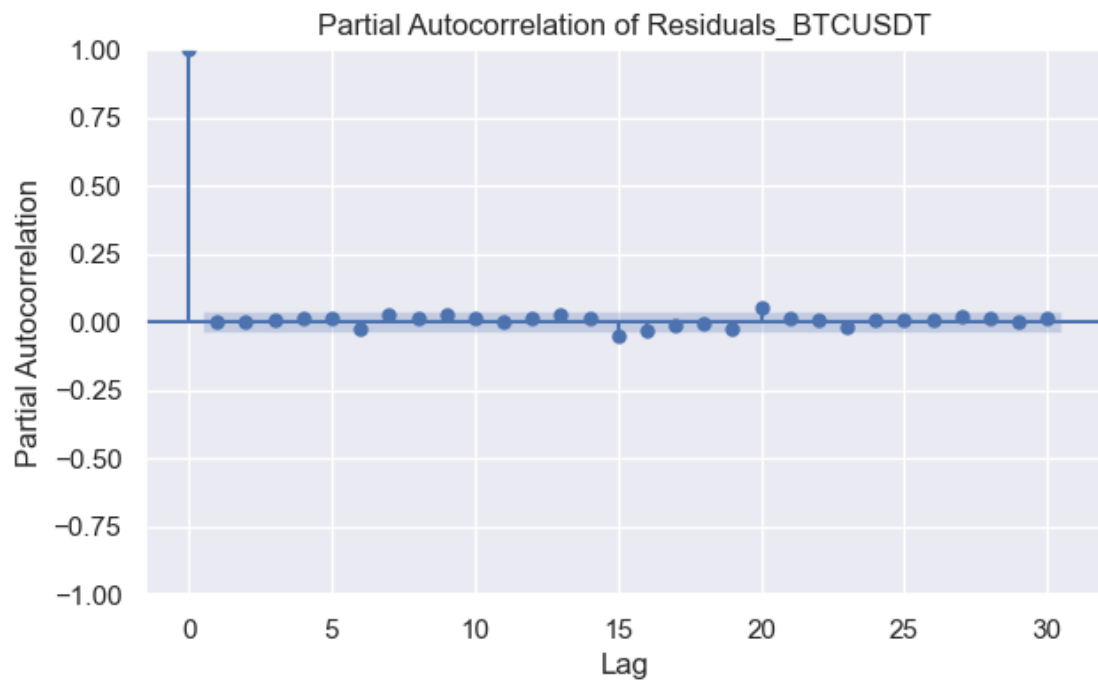
Jarque-Bera p-value: 0.0

Heteroskedasticity p-value: 1.6507209100506593e-53

ADF p-value: 0.0

=====





```
[ ]: model_arima_btc_exog = fit_auto(df, 'BTCUSDT', exogenous= ['Ret_BNBUSDT', 'Ret_ETHUSDT', 'Ret_ADAUSDT'])
```

SARIMAX Results

```
=====
Dep. Variable:          y      No. Observations:          2592
Model:                SARIMAX(0, 0, 1)  Log Likelihood      -4743.635
Date:                Thu, 29 Jun 2023    AIC                9501.270
Time:                12:14:37           BIC                9542.291
Sample:              06-01-2018         HQIC               9516.135
                  - 12-18-2021
```

Covariance Type: opg

```
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
intercept      0.0886      0.074      1.195      0.232      -0.057      0.234
drift      -6.454e-05      4.58e-05     -1.408      0.159      -0.000      2.53e-05
Ret_BNBUSDT      0.1228      0.008     15.339      0.000      0.107      0.138
Ret_ETHUSDT      0.4664      0.008     56.594      0.000      0.450      0.483
Ret_ADAUSDT      0.0742      0.008      8.945      0.000      0.058      0.090
ma.L1          0.0789      0.013      6.103      0.000      0.054      0.104
sigma2         2.2779      0.029     77.820      0.000      2.221      2.335
=====
```

===

```
Ljung-Box (L1) (Q):          0.00      Jarque-Bera (JB):
9053.21
Prob(Q):                    1.00      Prob(JB):
0.00
Heteroskedasticity (H):      1.67      Skew:
0.28
Prob(H) (two-sided):        0.00      Kurtosis:
12.14
```

```
=====
===
```

Warnings:

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

```
[ ]: analyze_residuals(model_arima_btc_exog, 'BTCUSDT_exog')
```

```
0
count  2592.000000
mean    0.002224
std     1.508866
min     -13.718343
25%     -0.662893
```

50% -0.039687
75% 0.615105
max 14.090610

Residual Analysis:

=====

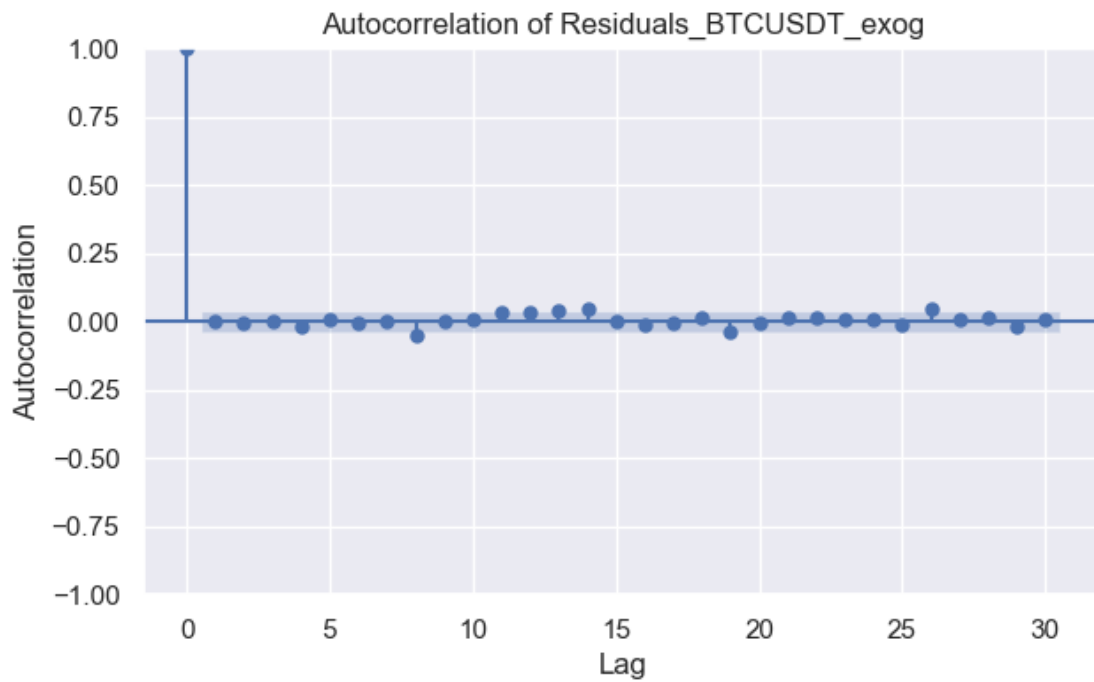
Ljung-Box (lag 1) p-value: 0.9988487367818375

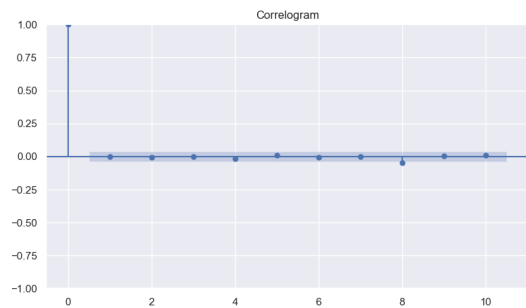
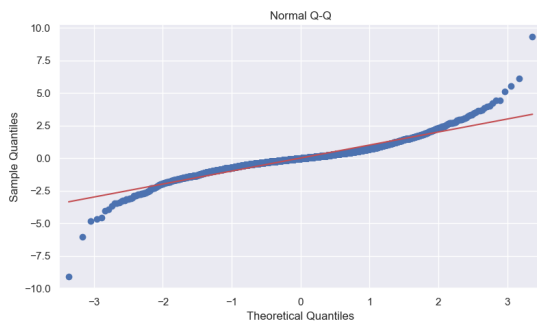
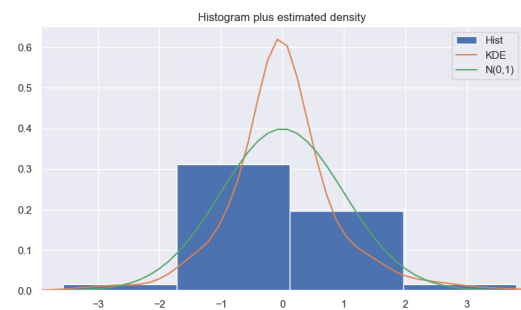
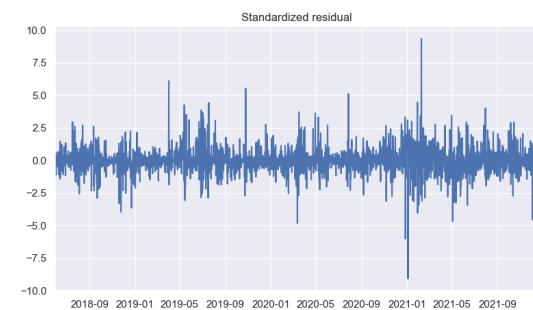
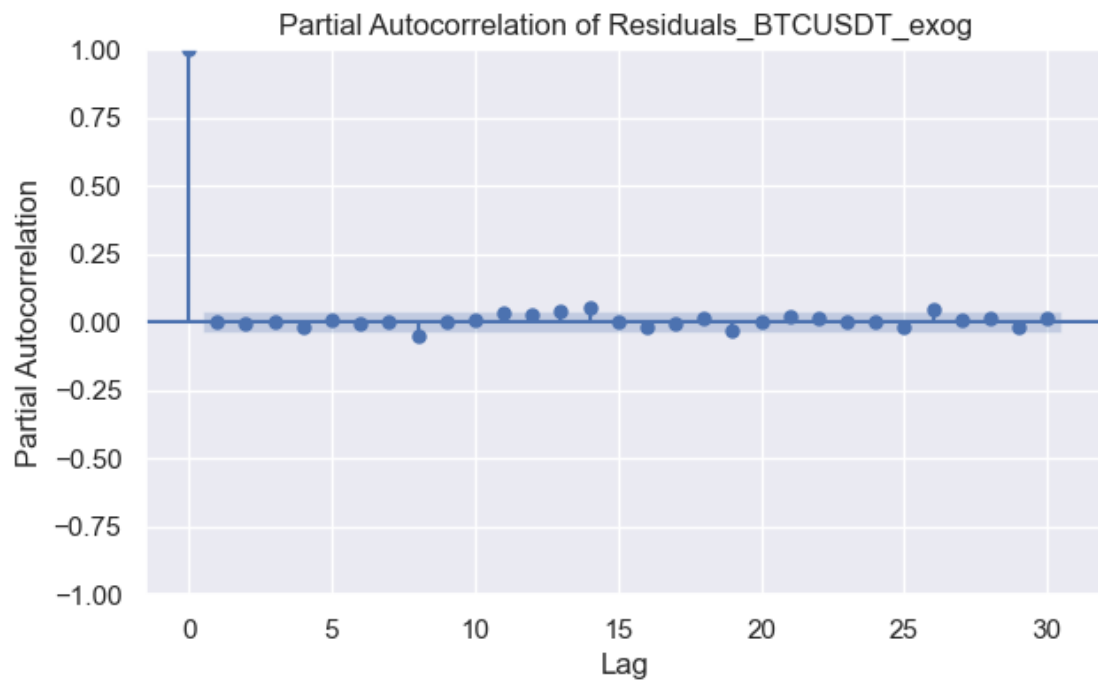
Jarque-Bera p-value: 0.0

Heteroskedasticity p-value: 2.9452294353070785e-13

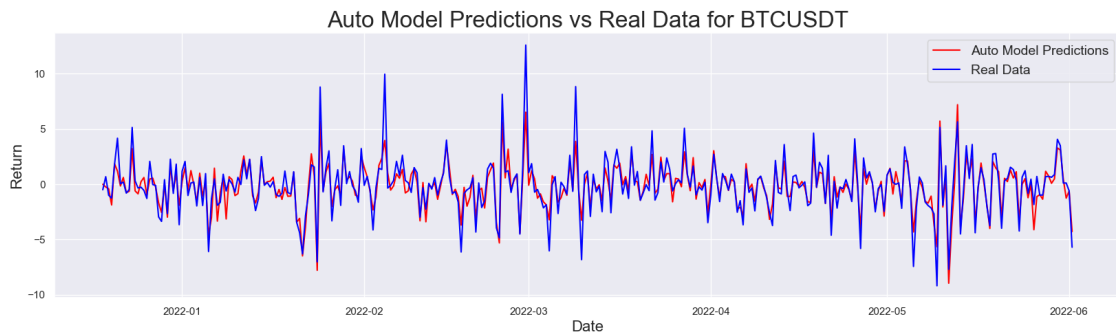
ADF p-value: 0.0

=====





```
[ ]: fit_pred(model_arima_btc_exog, 'BTCUSDT', exogenous= ['Ret_BNBUSDT',  
↳ "Ret_ETHUSDT", 'Ret_ADAUSDT'] )
```



```
[ ]: model_arima_ada = fit_auto(df, 'ADAUSDT', exog=False)
```

SARIMAX Results

```
=====
Dep. Variable:                y      No. Observations:          2592
Model:                SARIMAX(0, 1, 5)  Log Likelihood          -7322.288
Date:                Thu, 29 Jun 2023    AIC              14660.575
Time:                12:15:58            BIC              14707.454
Sample:                06-01-2018        HQIC             14677.563
                - 12-18-2021
```

Covariance Type: opg

```
=====
              coef    std err          z      P>|z|      [0.025      0.975]
-----
intercept      0.0098      0.004      2.219      0.027      0.001      0.019
drift      -6.839e-06    2.88e-06     -2.378      0.017     -1.25e-05     -1.2e-06
ma.L1       -0.8832      0.016    -56.904      0.000     -0.914     -0.853
ma.L2       -0.1333      0.021     -6.354      0.000     -0.174     -0.092
ma.L3        0.0952      0.023      4.091      0.000      0.050      0.141
ma.L4       -0.0274      0.023     -1.185      0.236     -0.073      0.018
ma.L5       -0.0325      0.018     -1.770      0.077     -0.068      0.003
sigma2       17.6643      0.304     58.188      0.000     17.069     18.259
=====
```

```
===
Ljung-Box (L1) (Q):                10.50    Jarque-Bera (JB):
2043.73
Prob(Q):                0.00    Prob(JB):
0.00
Heteroskedasticity (H):            1.67    Skew:
0.12
Prob(H) (two-sided):            0.00    Kurtosis:
7.34
```

```
=====
===
```

Warnings:

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

```
[ ]: analyze_residuals(model_arima_ada, 'ADAUSDT')
```

```

              0
count  2592.000000
mean    -0.056444
std      4.079430
min     -29.703929
25%     -2.033429
50%     -0.190451
75%      1.838509
max      22.889457
```

Residual Analysis:

```
=====
```

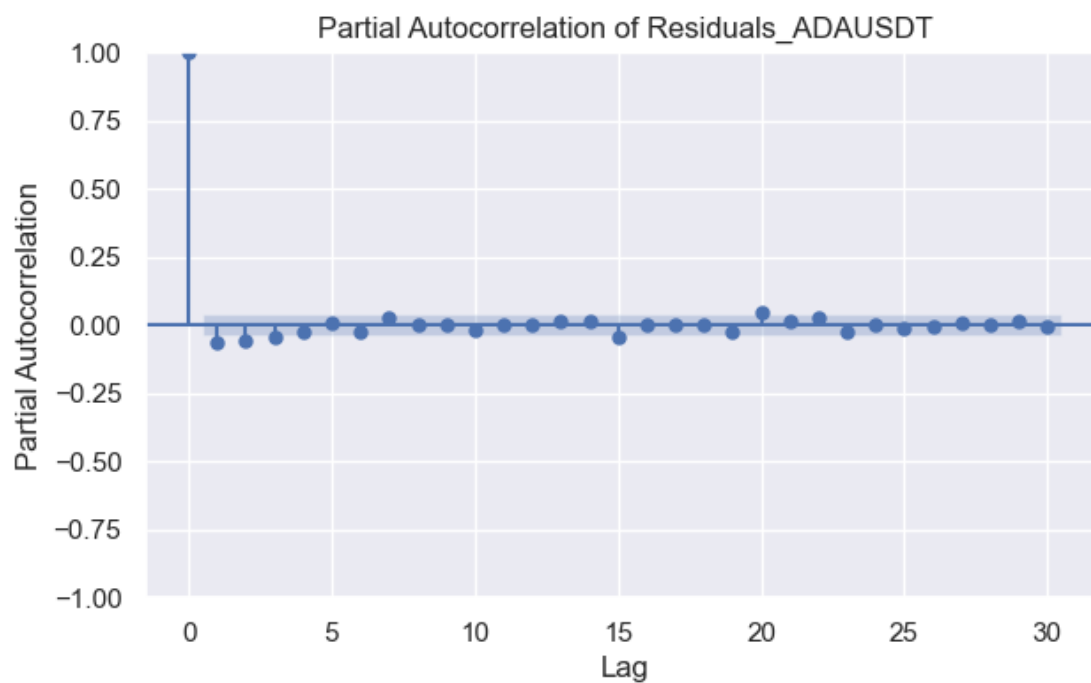
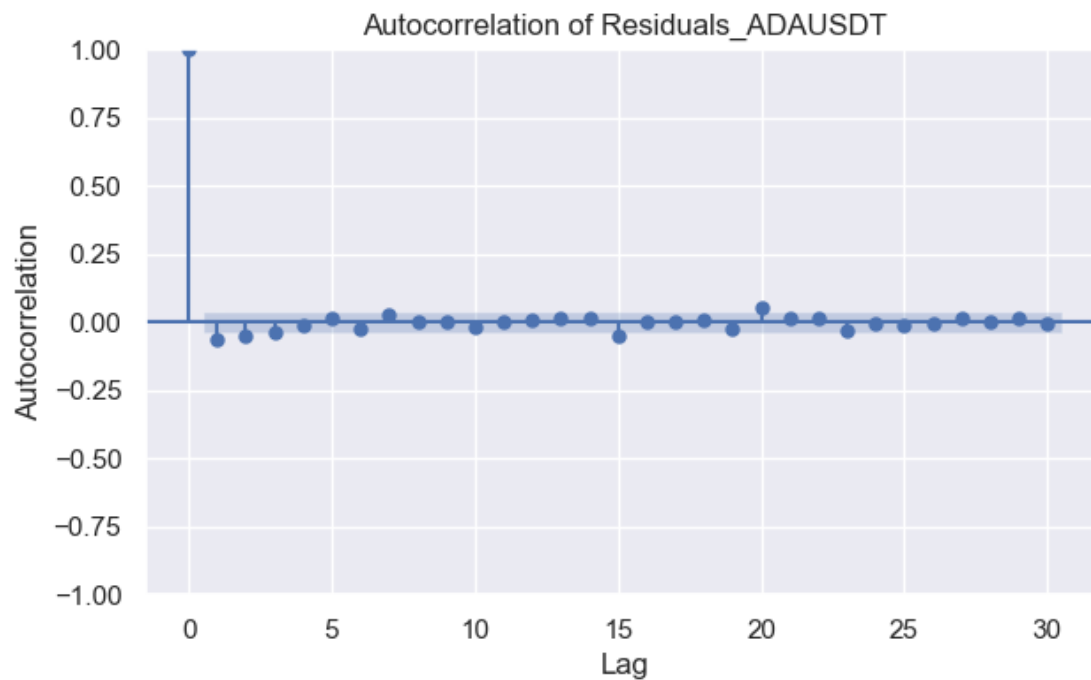
Ljung-Box (lag 1) p-value: 0.0012026015535582908

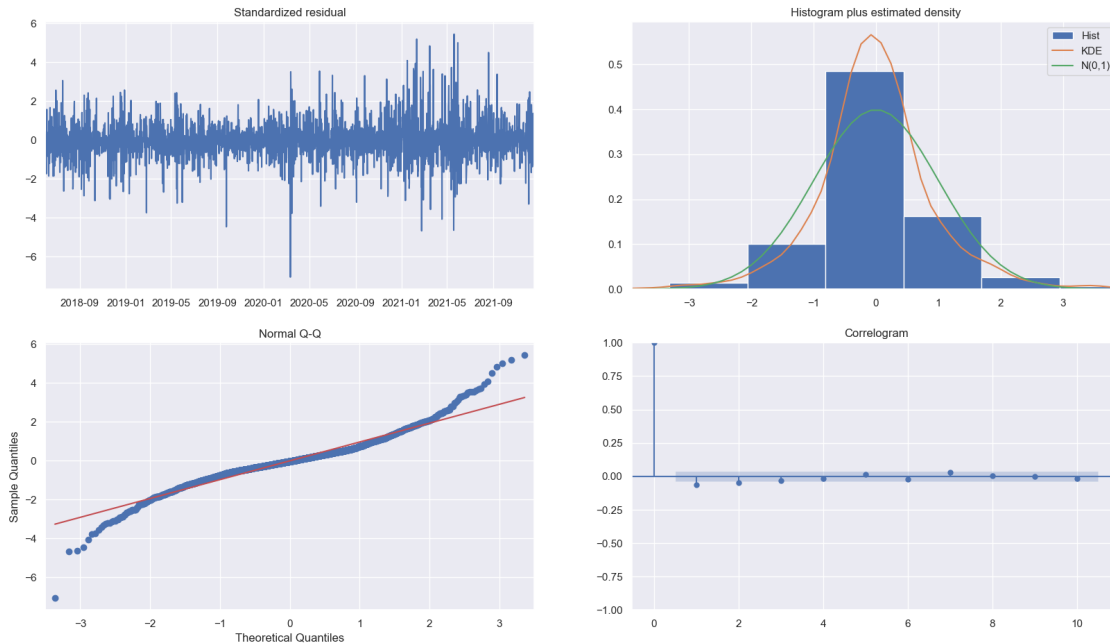
Jarque-Bera p-value: 0.0

Heteroskedasticity p-value: 1.4490184559264293e-28

ADF p-value: 0.0

```
=====
```





```
[ ]: model_arima_ada_exog = fit_auto(df, 'ADAUSD', exogenous=['Ret_BTCUSD', 'Ret_ETHUSD', 'Ret_BNBUSD'])
```

SARIMAX Results

```
=====
Dep. Variable:          y      No. Observations:      2592
Model:                SARIMAX(0, 0, 1)  Log Likelihood      -6127.400
Date:                Thu, 29 Jun 2023    AIC                12268.800
Time:                12:16:38            BIC                12309.821
Sample:              06-01-2018          HQIC              12283.665
                  - 12-18-2021

Covariance Type:      opg
=====
```

	coef	std err	z	P> z	[0.025	0.975]
intercept	-0.0585	0.139	-0.421	0.674	-0.331	0.214
drift	4.727e-05	8.16e-05	0.579	0.562	-0.000	0.000
Ret_BTCUSD	0.2146	0.024	9.073	0.000	0.168	0.261
Ret_ETHUSD	0.6010	0.020	29.876	0.000	0.562	0.640
Ret_BNBUSD	0.2040	0.011	19.225	0.000	0.183	0.225
ma.L1	0.0449	0.012	3.810	0.000	0.022	0.068
sigma2	6.6294	0.092	71.866	0.000	6.449	6.810

```
=====
Ljung-Box (L1) (Q):      0.00    Jarque-Bera (JB):
9294.04
```

```

Prob(Q):                1.00    Prob(JB):
0.00
Heteroskedasticity (H):    3.19    Skew:
1.51
Prob(H) (two-sided):      0.00    Kurtosis:
11.77

```

```

=====
===

```

Warnings:

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

```
[ ]: analyze_residuals(model_arima_ada_exog, 'ADAUSDT_exog')
```

```

              0
count  2592.000000
mean    0.000349
std     2.573378
min    -12.967833
25%    -1.324885
50%    -0.191984
75%     0.889473
max     19.448952

```

Residual Analysis:

```
=====
```

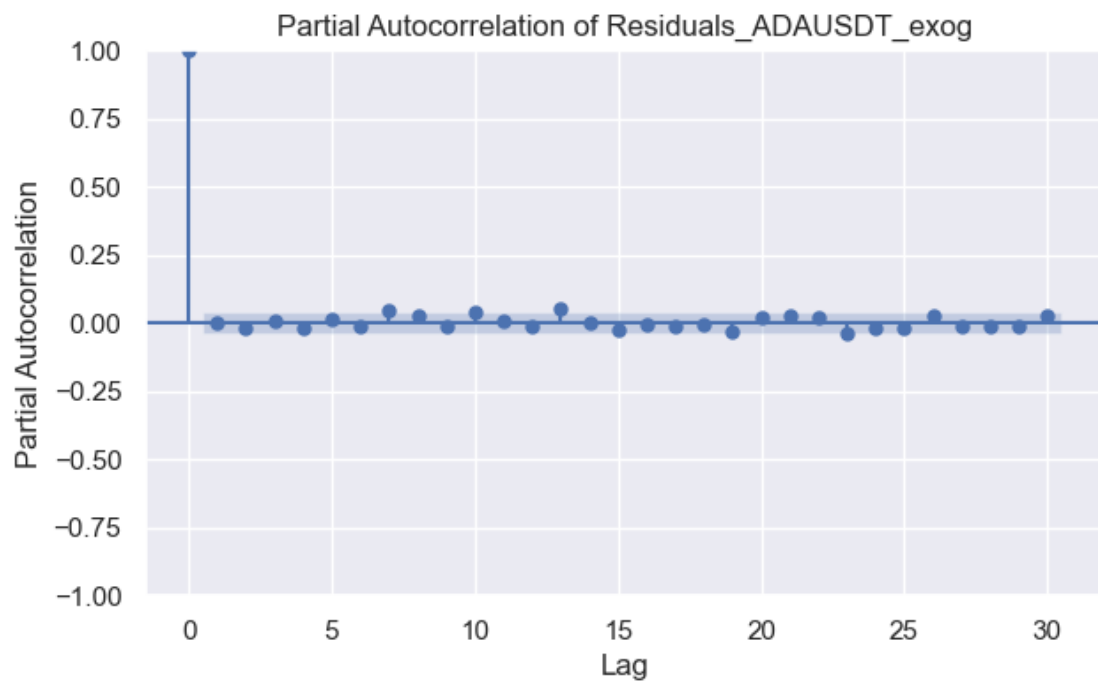
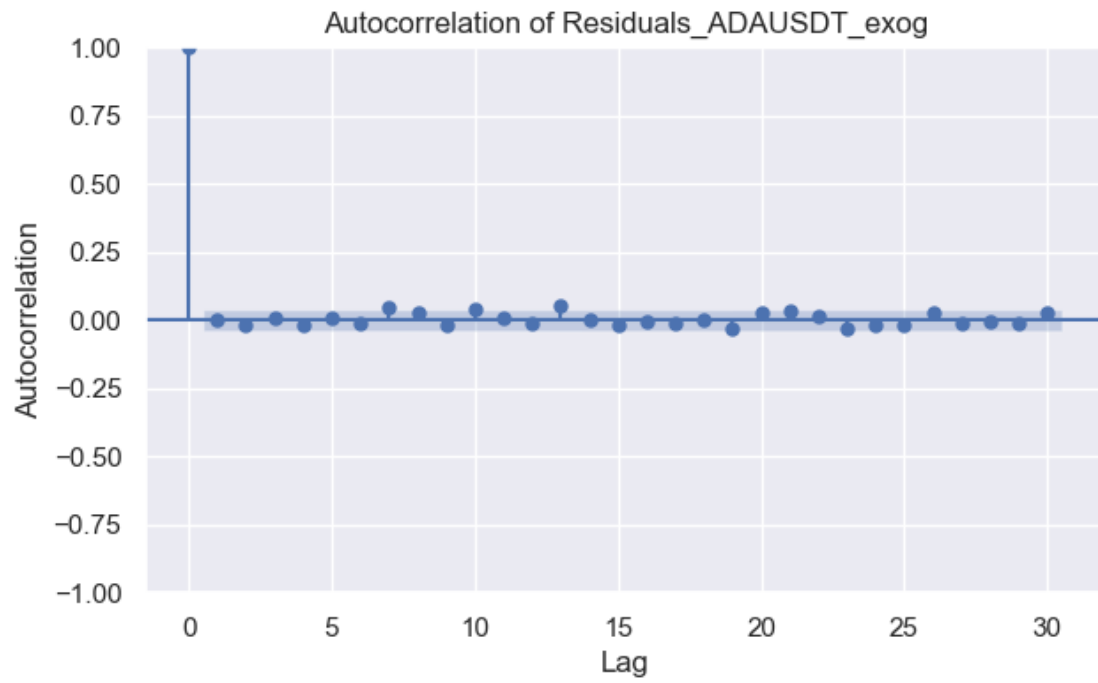
Ljung-Box (lag 1) p-value: 0.9956717218523251

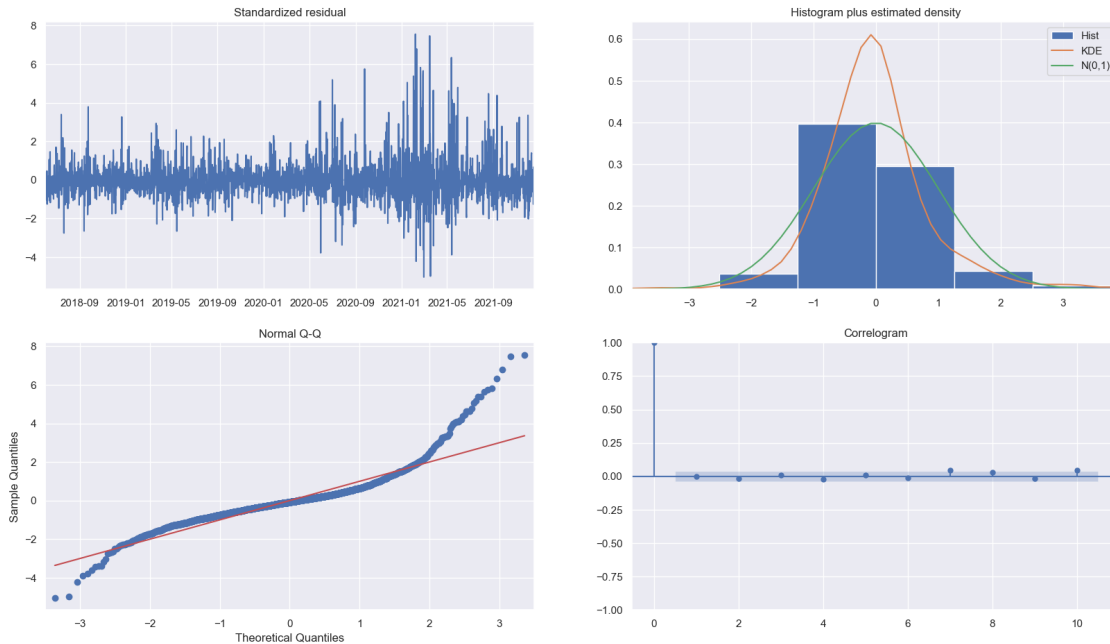
Jarque-Bera p-value: 0.0

Heteroskedasticity p-value: 1.3310543282414287e-51

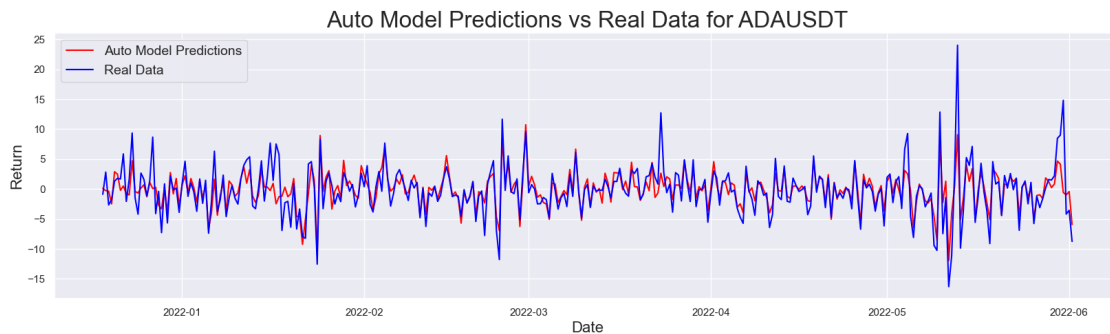
ADF p-value: 0.0

```
=====
```





```
[ ]: fit_pred(model_arima_ada_exog, 'ADAUSDT', exogenous=['Ret_BTCUSDT',
↳ 'Ret_ETHUSDT', 'Ret_BNBUSDT'])
```



```
[ ]: model_arima_eth = fit_auto(df, 'ETHUSDT', exog=False)
```

SARIMAX Results

```
=====
Dep. Variable:          y      No. Observations:      2592
Model:                 SARIMAX(0, 1, 4)  Log Likelihood      -6900.697
Date:                  Thu, 29 Jun 2023  AIC              13815.394
Time:                  12:18:37          BIC              13856.413
Sample:                06-01-2018        HQIC             13830.259
                  - 12-18-2021
Covariance Type:       opg
```

	coef	std err	z	P> z	[0.025	0.975]
intercept	-3.961e-05	0.008	-0.005	0.996	-0.016	0.016
drift	-3.552e-07	5.07e-06	-0.070	0.944	-1.03e-05	9.58e-06
ma.L1	-0.8650	0.013	-64.518	0.000	-0.891	-0.839
ma.L2	-0.1396	0.019	-7.275	0.000	-0.177	-0.102
ma.L3	0.0916	0.022	4.197	0.000	0.049	0.134
ma.L4	-0.0358	0.018	-2.015	0.044	-0.071	-0.001
sigma2	12.8468	0.200	64.207	0.000	12.455	13.239

===

```

Ljung-Box (L1) (Q):                12.05   Jarque-Bera (JB):
3557.47
Prob(Q):                          0.00   Prob(JB):
0.00
Heteroskedasticity (H):            1.34   Skew:
-0.40
Prob(H) (two-sided):              0.00   Kurtosis:
8.69

```

=====

===

Warnings:

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

```
[ ]: analyze_residuals(model_arima_eth, 'ETHUSDT')
```

```

0
count  2592.000000
mean    0.005349
std     3.466415
min     -29.192620
25%     -1.487099
50%     -0.042985
75%      1.556170
max      21.088147
Residual Analysis:

```

=====

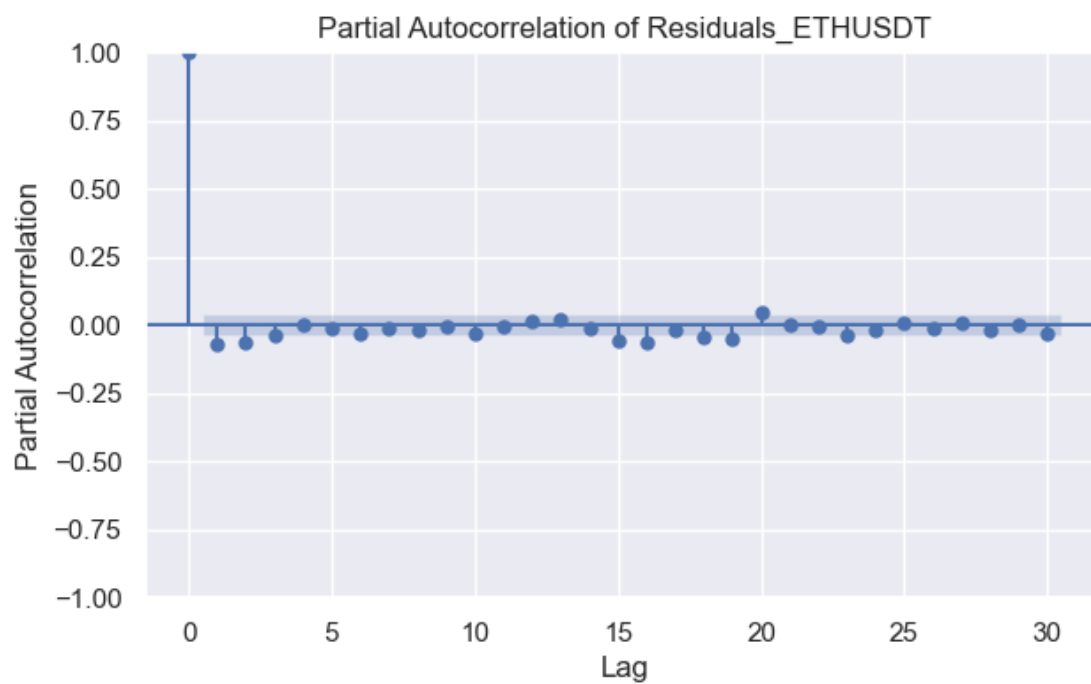
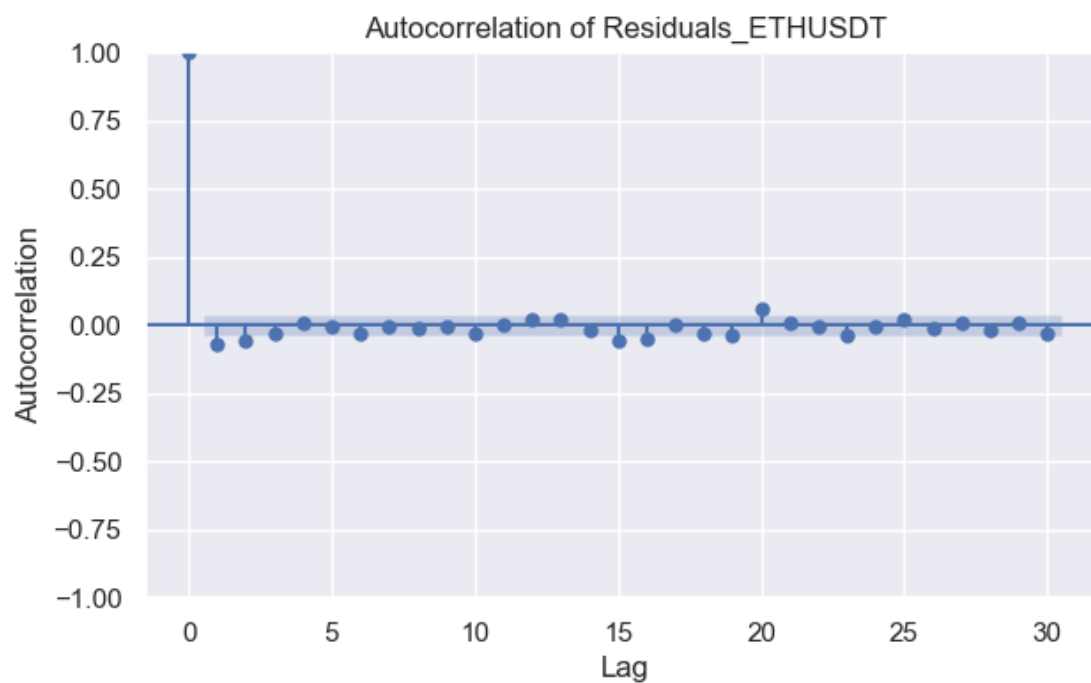
Ljung-Box (lag 1) p-value: 0.0005161000851650652

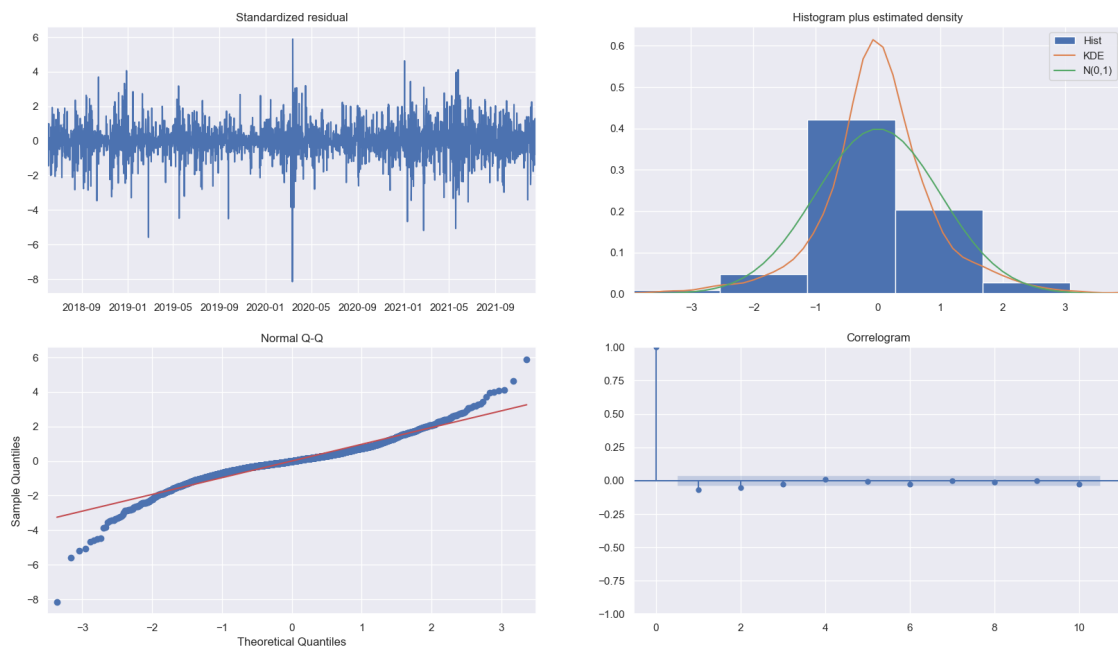
Jarque-Bera p-value: 0.0

Heteroskedasticity p-value: 1.3676721077124925e-51

ADF p-value: 0.0

=====





```
[ ]: model_arima_eth_exog = fit_auto(df, 'ETHUSDT', exogenous=['Ret_BTCUSDT', 'Ret_BNBUSDT', 'Ret_ADAUSDT'])
```

SARIMAX Results

```
=====
Dep. Variable:          y      No. Observations:      2592
Model:                SARIMAX(3, 0, 0)  Log Likelihood      -5094.692
Date:                 Thu, 29 Jun 2023  AIC              10207.385
Time:                 12:19:21   BIC              10260.127
Sample:              06-01-2018  HQIC             10226.498
                        - 12-18-2021
```

Covariance Type: opg

```
=====
              coef    std err          z      P>|z|      [0.025      0.975]
-----
intercept    -0.1082     0.076     -1.422    0.155     -0.257     0.041
drift         7.985e-05  4.65e-05     1.717    0.086    -1.13e-05    0.000
Ret_BTCUSDT    0.6099     0.011    55.495    0.000     0.588     0.631
Ret_BNBUSDT    0.1593     0.007    22.014    0.000     0.145     0.173
Ret_ADAUSDT    0.2716     0.007    39.785    0.000     0.258     0.285
ar.L1         0.0266     0.015     1.812    0.070     -0.002     0.055
ar.L2         0.0357     0.015     2.323    0.020     0.006     0.066
=====
```

ar.L3	0.0345	0.017	2.058	0.040	0.002	0.067
sigma2	2.9867	0.040	74.909	0.000	2.909	3.065

=====

===

Ljung-Box (L1) (Q): 0.00 Jarque-Bera (JB):

8060.55

Prob(Q): 0.96 Prob(JB):

0.00

Heteroskedasticity (H): 1.84 Skew:

1.02

Prob(H) (two-sided): 0.00 Kurtosis:

11.39

=====

===

Warnings:

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

```
[ ]: analyze_residuals(model_arima_eth_exog, 'ETHUSDT_exog')
```

0

count 2592.000000

mean -0.000263

std 1.727712

min -8.707345

25% -0.854268

50% -0.072341

75% 0.634941

max 17.296564

Residual Analysis:

=====

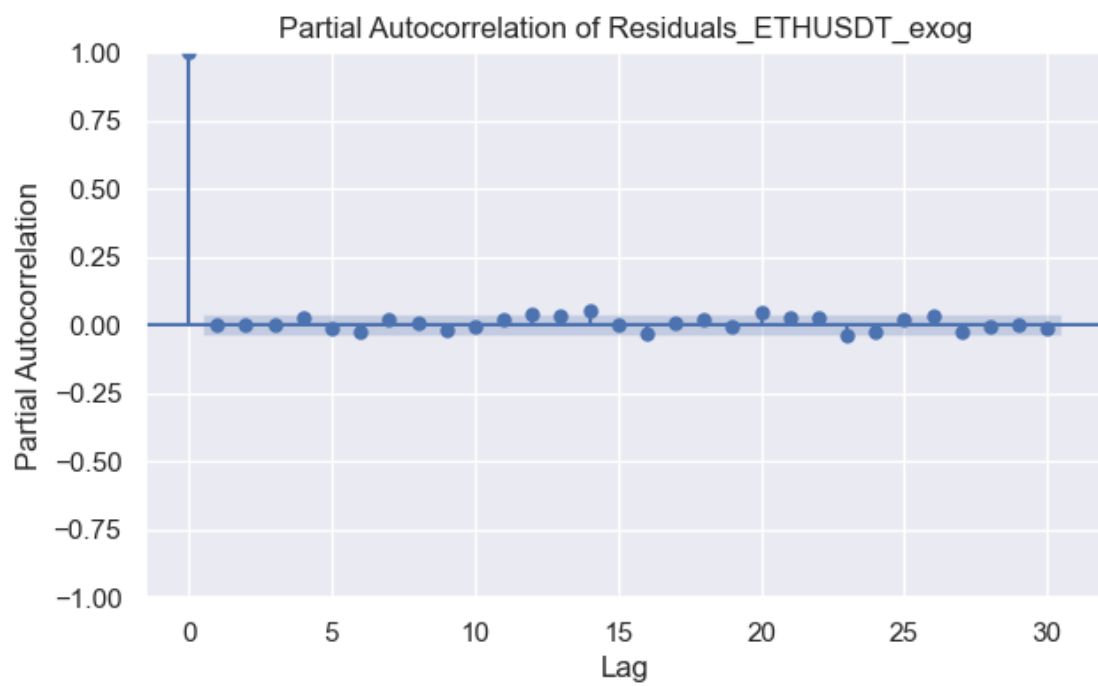
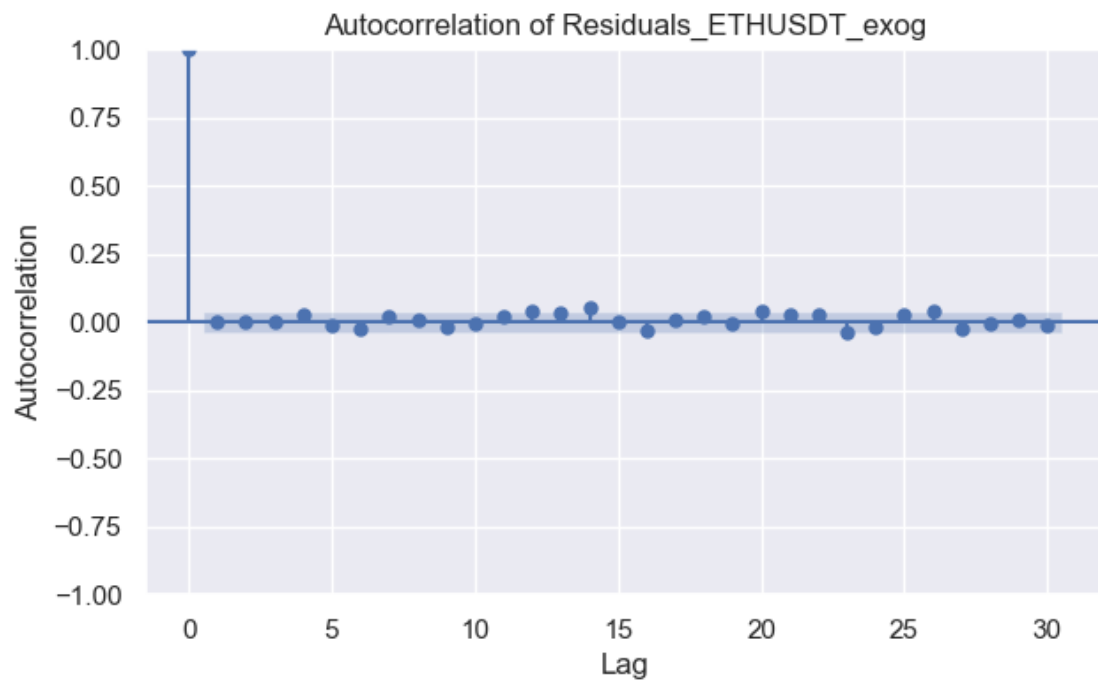
Ljung-Box (lag 1) p-value: 0.9572779336464502

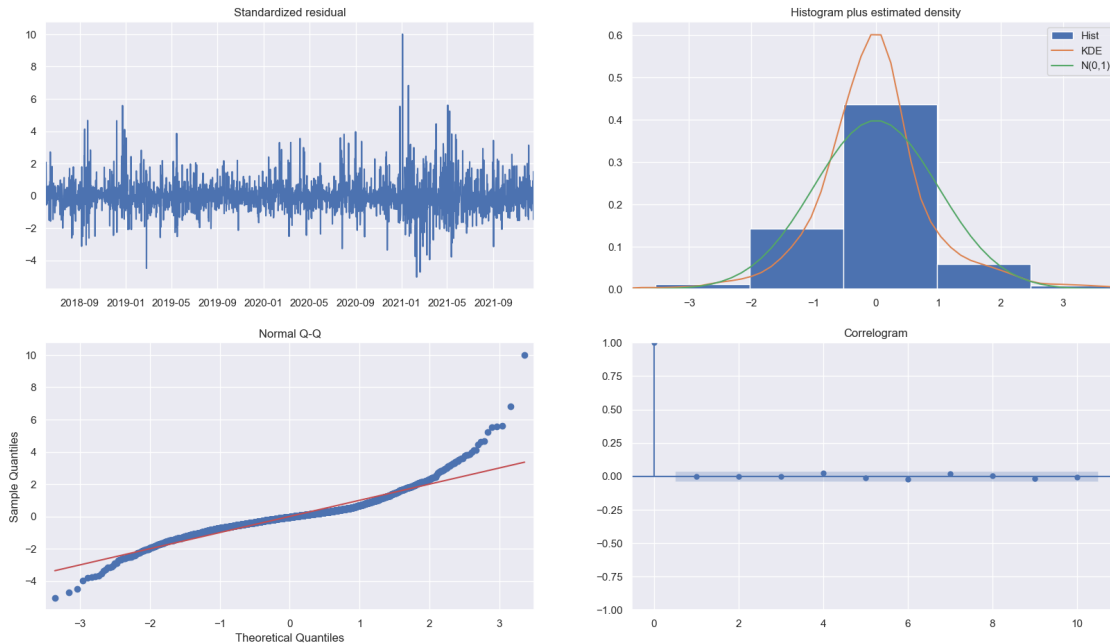
Jarque-Bera p-value: 0.0

Heteroskedasticity p-value: 9.67260472973527e-08

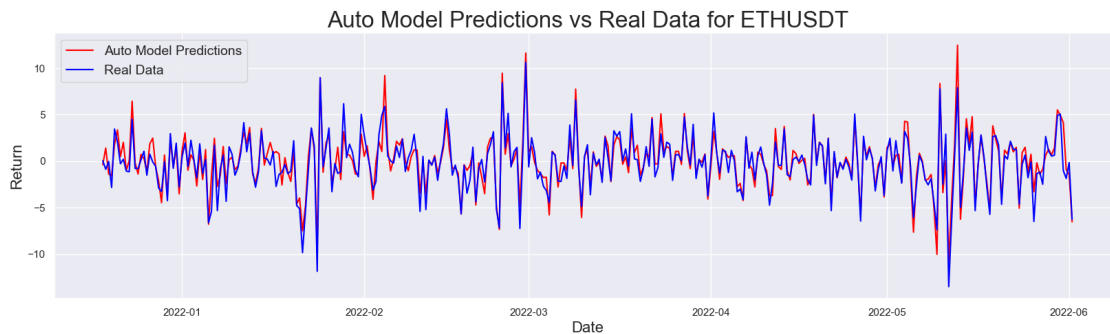
ADF p-value: 0.0

=====





```
[ ]: fit_pred(model_arima_eth_exog, 'ETHUSDT', exogenous=['Ret_BTCUSDT',
↳ 'Ret_BNBUSDT', 'Ret_ADAUSDT'] )
```



Forecasting Volitlty

```
[ ]:
```

```
[ ]: df_test.head()[start_date:]
```

```
[ ]:
      Date      BTCUSDT  ETHUSDT  ADAUSDT  BNBUSDT  Ret_BTCUSDT
2021-12-18 12:00:00  46834.48  3959.93    1.242    533.9   -0.501291 \
2021-12-19 00:00:00  47152.72  3925.36    1.277    535.0    0.679499
2021-12-19 12:00:00  46681.23  3924.14    1.243    529.8   -0.999921
```

2021-12-20 00:00:00	46092.60	3813.02	1.221	514.9	-1.260956
2021-12-20 12:00:00	46914.16	3944.73	1.237	524.0	1.782412

	Ret_cum_BTCUSDT	Norm_BTCUSDT	Ret_ETHUSDT
Date			
2021-12-18 12:00:00	278.249627	636.217888	0.045476 \
2021-12-19 00:00:00	278.929126	640.540985	-0.872995
2021-12-19 12:00:00	277.929205	634.136080	-0.031080
2021-12-20 00:00:00	276.668249	626.139900	-2.831703
2021-12-20 12:00:00	278.450661	637.300292	3.454217

	Ret_cum_ETHUSDT	Norm_ETHUSDT	Ret_ADAUSDT
Date			
2021-12-18 12:00:00	349.537959	693.957556	-0.877893 \
2021-12-19 00:00:00	348.664963	687.899339	2.818035
2021-12-19 12:00:00	348.633884	687.685541	-2.662490
2021-12-20 00:00:00	345.802180	668.212327	-1.769912
2021-12-20 12:00:00	349.256398	691.293833	1.310401

	Ret_cum_ADAUSDT	Norm_ADAUSDT	Ret_BNBUSDT
Date			
2021-12-18 12:00:00	389.675144	575.532901	0.357143 \
2021-12-19 00:00:00	392.493179	591.751622	0.206031
2021-12-19 12:00:00	389.830689	575.996293	-0.971963
2021-12-20 00:00:00	388.060778	565.801668	-2.812382
2021-12-20 12:00:00	389.371179	573.215941	1.767333

	Ret_cum_BNBUSDT	Norm_BNBUSDT
Date		
2021-12-18 12:00:00	552.756524	3874.455733
2021-12-19 00:00:00	552.962555	3882.438316
2021-12-19 12:00:00	551.990592	3844.702467
2021-12-20 00:00:00	549.178210	3736.574746
2021-12-20 12:00:00	550.945544	3802.612482

```
[ ]: df_com = pd.concat([df, df_test])
```

```
[ ]: df_com.tail()
```

	BTCUSDT	ETHUSDT	ADAUSDT	BNBUSDT	Ret_BTCUSDT
Date					
2023-06-25 00:00:00	30716.03	1918.37	0.2981	240.2	0.617805 \
2023-06-25 12:00:00	30462.66	1898.80	0.2921	238.8	-0.824879
2023-06-26 00:00:00	30322.89	1880.87	0.2890	237.9	-0.458824
2023-06-26 12:00:00	30267.99	1858.97	0.2805	236.1	-0.181051
2023-06-27 00:00:00	30735.98	1881.51	0.2842	239.4	1.546155

	Ret_cum_BTCUSDT	Norm_BTCUSDT	Ret_ETHUSDT	
Date				
2023-06-25 00:00:00	262.724610	417.258561	2.315249	\
2023-06-25 12:00:00	261.899732	413.816684	-1.020137	
2023-06-26 00:00:00	261.440908	411.917994	-0.944281	
2023-06-26 12:00:00	261.259856	411.172211	-1.164355	
2023-06-27 00:00:00	262.806011	417.529570	1.212499	

	Ret_cum_ETHUSDT	Norm_ETHUSDT	Ret_ADAUSDT	
Date				
2023-06-25 00:00:00	321.074468	336.184568	3.077455	\
2023-06-25 12:00:00	320.054331	332.755025	-2.012747	
2023-06-26 00:00:00	319.110050	329.612884	-1.061280	
2023-06-26 12:00:00	317.945696	325.775021	-2.941176	
2023-06-27 00:00:00	319.158195	329.725041	1.319073	

	Ret_cum_ADAUSDT	Norm_ADAUSDT	Ret_BNBUSDT	
Date				
2023-06-25 00:00:00	305.644200	138.137164	1.607445	\
2023-06-25 12:00:00	303.631452	135.356812	-0.582848	
2023-06-26 00:00:00	302.570172	133.920297	-0.376884	
2023-06-26 12:00:00	299.628996	129.981464	-0.756620	
2023-06-27 00:00:00	300.948069	131.696015	1.397713	

	Ret_cum_BNBUSDT	Norm_BNBUSDT
Date		
2023-06-25 00:00:00	506.181376	1743.105951
2023-06-25 12:00:00	505.598528	1732.946299
2023-06-26 00:00:00	505.221644	1726.415094
2023-06-26 12:00:00	504.465023	1713.352685
2023-06-27 00:00:00	505.862736	1737.300435

```
[ ]:
```

```
[ ]: def garch_volatility(column, df = df_com, df_test = df_test, start_date =
      ↪start_date):

    model = arch.arch_model(df[f'Ret_{column}'], vol='Garch', p=1, q=1)

    # Fit the model
    results = model.fit(last_obs=start_date, update_freq=10)

    #forecast
    pred_garch = results.forecast(horizon=1, align='target')

    # Plot the estimated volatility
```

```

fig, ax = plt.subplots(figsize=(20, 6))
pred_garch.residual_variance[start_date:].plot(figsize = (20,5), color = "red", zorder = 2, ax = ax)
df_test[f'Ret_{column}'].abs().plot(color = "blue", zorder = 1, ax = ax)
plt.title("Volatility Predictions", size = 24)
ax.set_xlabel('Time', fontsize = 15)
ax.set_ylabel('Volatility', fontsize = 17)
ax.set_title(f'GARCH Estimated Volatility {column}', fontsize = 20)
ax.legend()
plt.savefig(f'../reports/figures/Ret_{column}_garch_forecast.png')
plt.show()

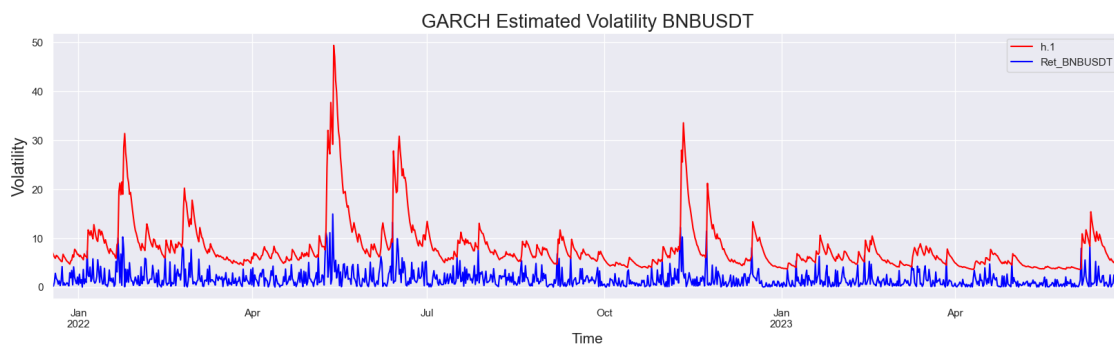
```

```
[ ]: garch_volatility('BNBUSDT')
```

```

Iteration:      10,   Func. Count:      63,   Neg. LLF: 6787.4369759566325
Optimization terminated successfully   (Exit mode 0)
    Current function value: 6787.4369759566325
    Iterations: 11
    Function evaluations: 67
    Gradient evaluations: 11

```

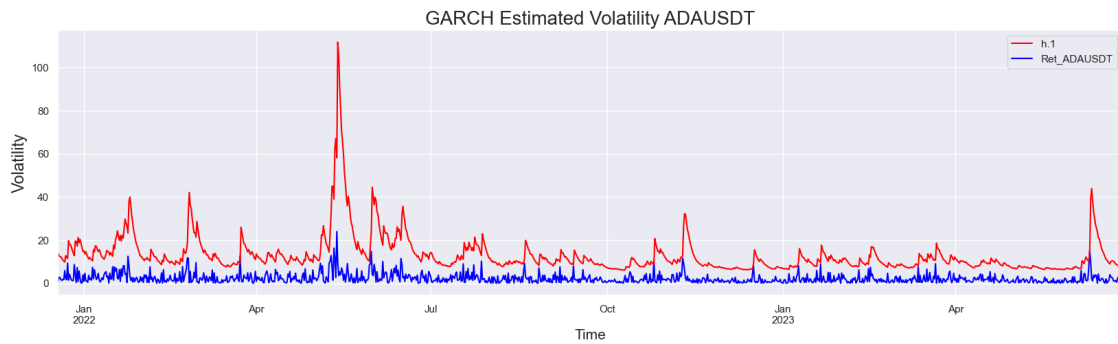


```
[ ]: garch_volatility('ADAUSDT')
```

```

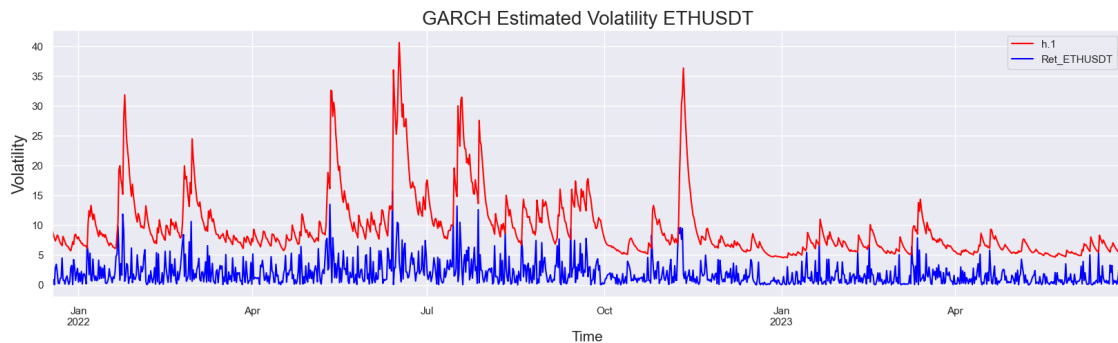
Iteration:      10,   Func. Count:      60,   Neg. LLF: 7151.37612664454
Optimization terminated successfully   (Exit mode 0)
    Current function value: 7151.376082066814
    Iterations: 13
    Function evaluations: 74
    Gradient evaluations: 13

```



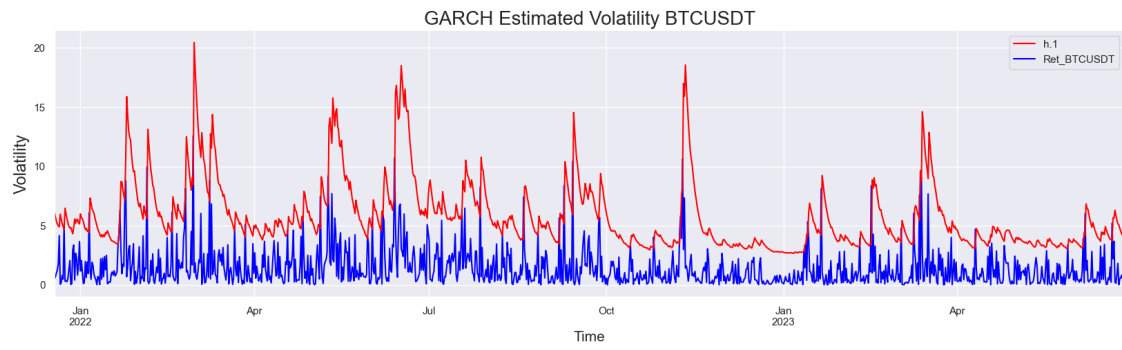
```
[ ]: garch_volatility('ETHUSDt')
```

```
Iteration:      10,  Func. Count:      61,  Neg. LLF: 6708.680754873141
Optimization terminated successfully  (Exit mode 0)
Current function value: 6708.680717520079
Iterations: 12
Function evaluations: 71
Gradient evaluations: 12
```



```
[ ]: garch_volatility('BTCUSDt')
```

```
Iteration:      10,  Func. Count:      66,  Neg. LLF: 6042.567397166942
Optimization terminated successfully  (Exit mode 0)
Current function value: 6042.567397166942
Iterations: 11
Function evaluations: 70
Gradient evaluations: 11
```



[]:

[]:

[]:

[]: