In [1]:
```
# Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.graphics.tsaplots as sgt
import statsmodels.tsa.stattools as sts
from statsmodels.tsa.seasonal import seasonal_decompose
import seaborn as sns
import scipy.stats
import pylab
import os
from matplotlib import pyplot
from scipy.stats.distributions import chi2
from math import sqrt
from statsmodels.tsa.arima.model import ARIMA
sns.set()
```

In [32]:
```
from statsmodels.tsa.arima_process import ArmaProcess
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.ar_model import AutoReg
```

In [2]:
```
os.chdir("C:\\Users\\tural\\Downloads\\Crypto_Datasets")
```

Univariate Time Series Analysis of Bitcoin

In [239…
```
#Importing the data, checking teh number of observations and variables and types
btcdata = pd.read_csv("BTC-USD.csv")
btcdata.Date = pd.to_datetime(btcdata.Date, yearfirst= True)
```

In [240…
```
# Dropping columns adj close as it is the same as close price
btcdata.drop(['Adj Close'], axis=1, inplace= True)
```

In [241…
```
btcdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3028 entries, 0 to 3027
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   Date    3028 non-null   datetime64[ns]
 1   Open    3028 non-null   float64       
 2   High    3028 non-null   float64       
 3   Low     3028 non-null   float64       
 4   Close   3028 non-null   float64       
 5   Volume  3028 non-null   int64         
dtypes: datetime64[ns](1), float64(4), int64(1)
memory usage: 142.1 KB
```

In [244…
```
btcdata.head()
```

Out[244…

Date	Open	High	Low	Close	Volume
2014-09-17	465.864014	468.174011	452.421997	457.334015	21056800
2014-09-18	456.859985	456.859985	413.104004	424.440002	34483200

Date	Open	High	Low	Close	Volume
2014-09-19	424.102997	427.834991	384.532013	394.795990	37919700
2014-09-20	394.673004	423.295990	389.882996	408.903992	36863600
2014-09-21	408.084991	412.425995	393.181000	398.821014	26580100

In [243…
```
# Setting dates as index so we dont have it as a seperate data column
btcdata.set_index('Date', inplace = True)
```

In [8]:
```
# Positions and Variance of the observations
btcdata.describe()
```

Out[8]:

	Open	High	Low	Close	Volume
count	3028.000000	3028.000000	3028.000000	3028.000000	3.028000e+03
mean	12858.370058	13182.567711	12495.418710	12862.178425	1.634383e+10
std	16190.316636	16604.605750	15710.663914	16185.351172	1.997501e+10
min	176.897003	211.731003	171.509995	178.102997	5.914570e+06
25%	672.789017	684.548492	663.220993	675.556260	1.006505e+08
50%	7097.275635	7287.041260	6894.411133	7106.541992	7.367460e+09
75%	16804.387207	17089.460937	16530.455566	16803.772949	2.792719e+10
max	67549.734375	68789.625000	66382.062500	67566.828125	3.509679e+11

In [9]:
```
btcdata = btcdata.asfreq('d')
```

In [10]:
```
btcdata.head()
```

Out[10]:

Date	Open	High	Low	Close	Volume
2014-09-17	465.864014	468.174011	452.421997	457.334015	21056800
2014-09-18	456.859985	456.859985	413.104004	424.440002	34483200
2014-09-19	424.102997	427.834991	384.532013	394.795990	37919700
2014-09-20	394.673004	423.295990	389.882996	408.903992	36863600
2014-09-21	408.084991	412.425995	393.181000	398.821014	26580100

In [11]:
```
close_prices = btcdata['Close']
volumes = btcdata['Volume']
```

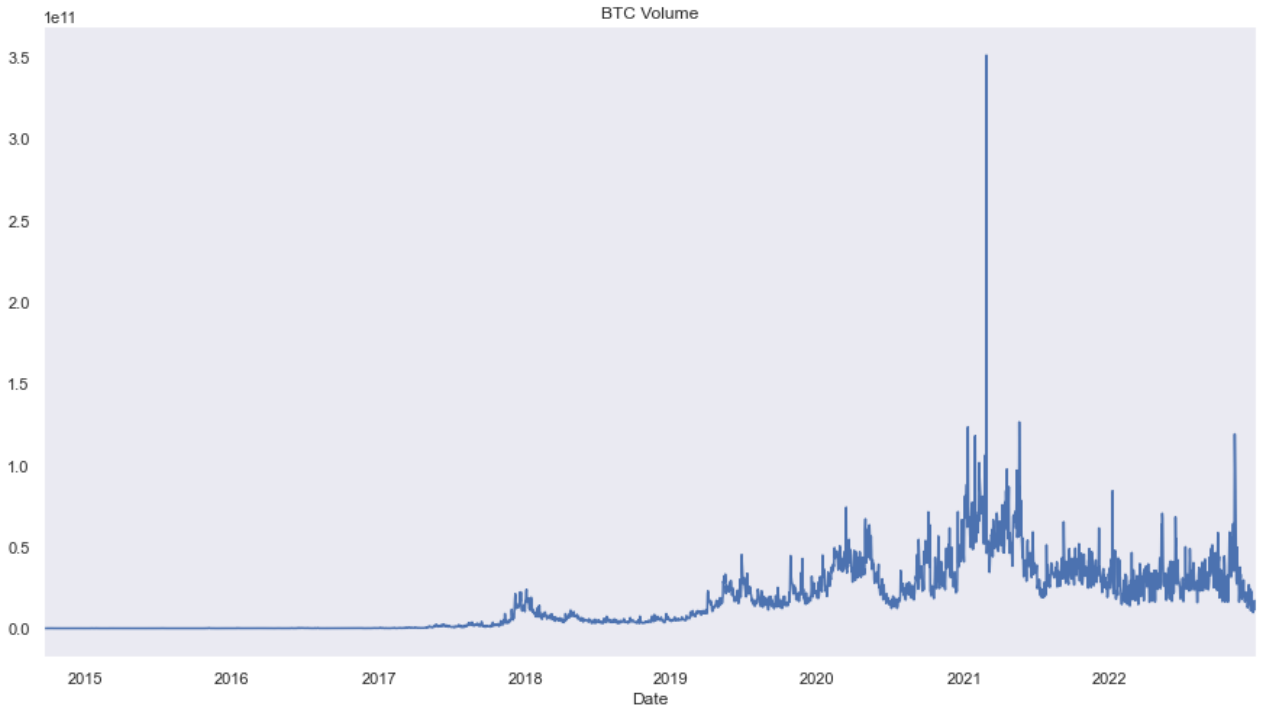
In [12]:
```
# Univariate Analysis

btcdata.Close.plot(title = "BTC Price", figsize = (15,5), grid  = False)
plt.title("BTC Prices", size = 24)
```

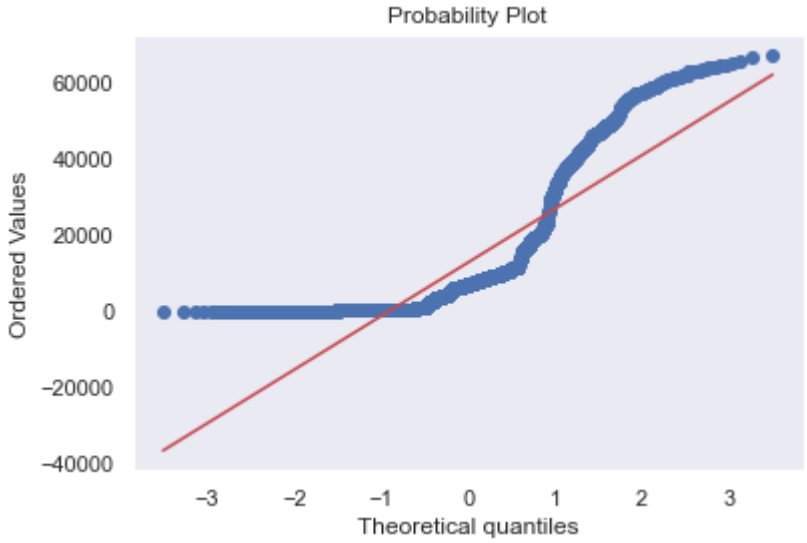
Out[12]: Text(0.5, 1.0, 'BTC Prices')

In [13]:
```
# Univariate Analysis
btcdata.Volume.plot(title = "BTC Volume", figsize = (15,8), grid  = False)
plt.title("BTC Volume")
```

Out[13]: Text(0.5, 1.0, 'BTC Volume')



In [15]:
```
# In the next step we are building a QQ(Quantile-Quantile plot) in order to understand
scipy.stats.probplot(btcdata.Close, plot = pylab)
pylab.grid(visible = None)
pylab.show()
```



In [367…
```
# Finding the 80 percent of the data to set as train data.
size = int(len(btcdata)*0.8)
```

In [368…
```
size = int(len(ethdata)*0.8)
```

In [369…
```
# Assigning first 80 percent of the data into training data set
btc_train = btcdata.iloc[:size]
```

In [243…
```
eth_train = ethdata.iloc[:size]
```

In [256…
```
btc_train.tail()
```

Out[256…

Date	Open	High	Low	Close	Volume	returns	norm	
2021-04-30	53568.664063	57900.718750	53129.601563	57750.175781	52395931985	7.833177	13606.204766	-1
2021-05-01	57714.664063	58448.339844	57052.273438	57828.050781	42836427360	0.134848	13624.552471	
2021-05-02	57825.863281	57902.593750	56141.906250	56631.078125	38177405335	-2.069882	13342.540255	
2021-05-03	56620.273438	58973.308594	56590.871094	57200.292969	51713139031	1.005128	13476.649868	
2021-05-04	57214.179688	57214.179688	53191.425781	53333.539063	68564706967	-6.760025	12565.625015	

In [255…
```
ethdata.head()
```

Out[255…

Date	Open	High	Low	Close	Adj Close	Volume
2017-11-09	308.644989	329.451996	307.056000	320.884003	320.884003	893249984

Date	Open	High	Low	Close	Adj Close	Volume
2017-11-10	320.670990	324.717987	294.541992	299.252991	299.252991	885985984
2017-11-11	298.585999	319.453003	298.191986	314.681000	314.681000	842300992
2017-11-12	314.690002	319.153015	298.513000	307.907990	307.907990	1613479936
2017-11-13	307.024994	328.415009	307.024994	316.716003	316.716003	1041889984

In [261…
```
ethdata.tail()
```

Out[261…

Date	Open	High	Low	Close	Adj Close	Volume
2022-12-27	1226.987061	1230.418091	1205.895630	1212.791626	1212.791626	4091530737
2022-12-28	1212.736572	1213.128906	1185.702148	1189.986084	1189.986084	4991669631
2022-12-29	1190.010132	1204.141602	1188.360229	1201.595337	1201.595337	4132233940
2022-12-30	1201.569580	1202.034668	1187.462524	1199.232788	1199.232788	4055668253
2022-12-31	1199.360107	1205.088623	1194.203735	1196.771240	1196.771240	3018513333

In [ ]:
```
```

In [370…
```
# Assigning last 20 percent of the data into testing data set
btc_test = btcdata.iloc[size:]
```

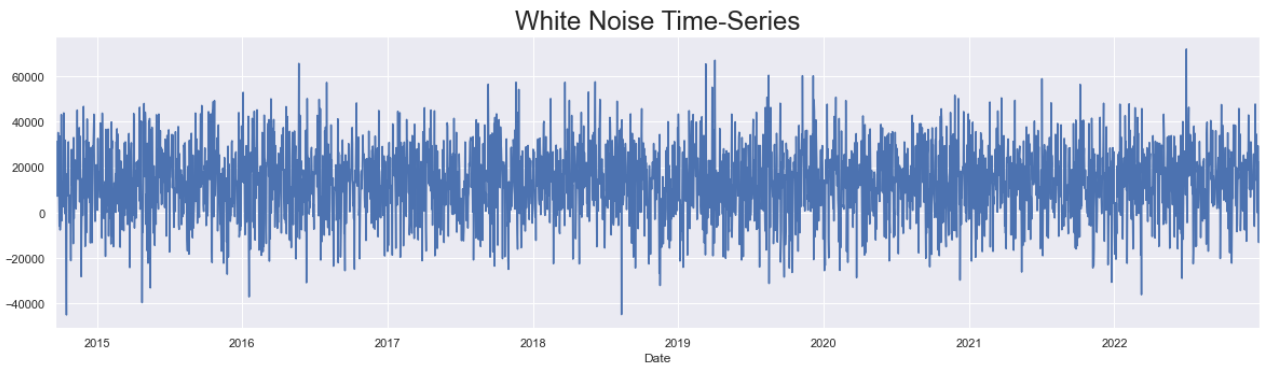
In [244…
```
eth_test = ethdata.iloc[size:]
```

In [555…
```
wn = np.random.normal(loc = btcdata.Close.mean(), scale = btcdata.Close.std(), size = l
```

In [556…
```
btcdata['wn'] = wn
```

In [557…
```
# Creating White-Noise Time Series
btcdata.wn.plot(figsize = (20,5))
plt.title("White Noise Time-Series", size = 24)
```

Out[557…
```
Text(0.5, 1.0, 'White Noise Time-Series')
```



In [558…
```
print(wn.mean())
```

```
13384.990595423738
```

Stationarity

In [560…
```
#Dicky-Fuller test to test stationarity
##as our T statistic is greated than all of the confidence levels, we dont have enough
#some autocorrelations go back 28 times, 2900 is the number of observations that test w
sts.adfuller(ethdata.Close)
```

Out[560…
```
(-1.4089708365742828,
 0.5779679105330212,
 17,
 1861,
 {'1%': -3.4338687226315336,
  '5%': -2.863094318475046,
  '10%': -2.5675974634086765},
 21446.440104463112)
```

In [559…
```
#Dicky-Fuller test to test White Noise
##as our T statistic is greated than all of the confidence levels, we dont have enough
#some autocorrelations go back 28 times, 2900 is the number of observations that test w
sts.adfuller(ethdata.wn)
```

Out[559…
```
(-15.611424890797505,
 1.7778347312680522e-28,
 6,
 1872,
 {'1%': -3.4338480179204556,
  '5%': -2.863085177979608,
  '10%': -2.567592596439203},
 31560.49364000751)
```

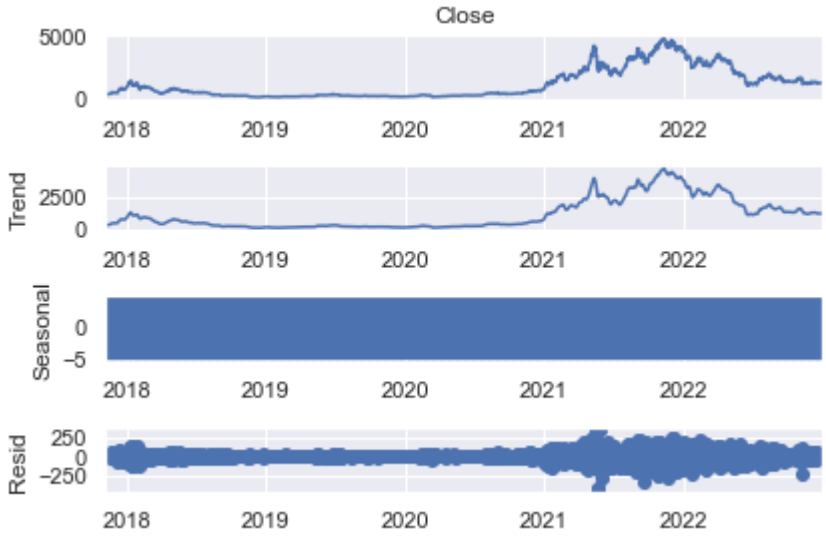
add random walk test as well

In [25]:
```
#Dicky-Fuller test to test stationarity for Volume
##as our T statistic is greated than all of the confidence levels, we dont have enough
#some autocorrelations go back 28 times, 2900 is the number of observations that test w
sts.adfuller(btcdata.Volume)
```

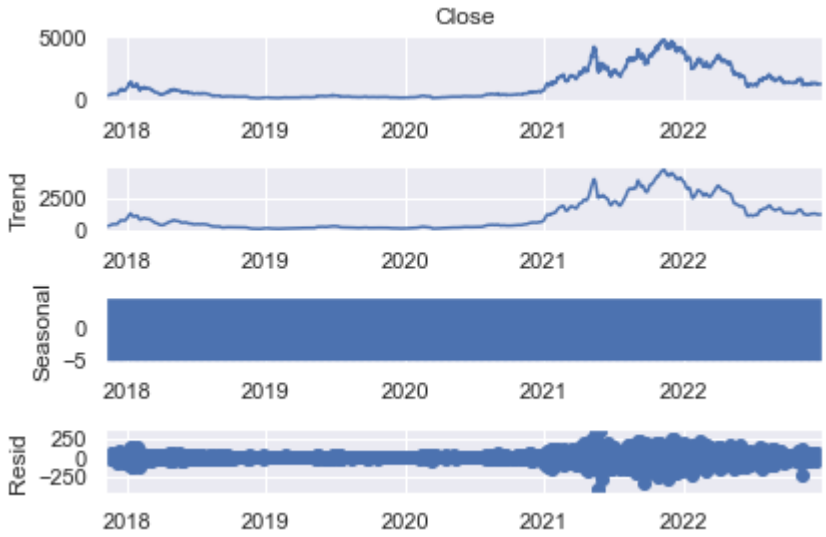
Out[25]:
```
(-2.089381063668316,
 0.24880591187769746,
 29,
 2998,
 {'1%': -3.4325330913621452,
  '5%': -2.862504548608965,
  '10%': -2.5672834546224057},
 145334.51937965507)
```

Seasonality

In [561…
```
# Seasonality, from seasonal graph you can see that there is no clear seasonal cycle.
# Resid shows, that in 2018, mid2019 2020, 2021,2022 there were a lot of differences be
s_dec_additive = seasonal_decompose(ethdata.Close, model = "additive")
s_dec_additive.plot()
plt.show()
```



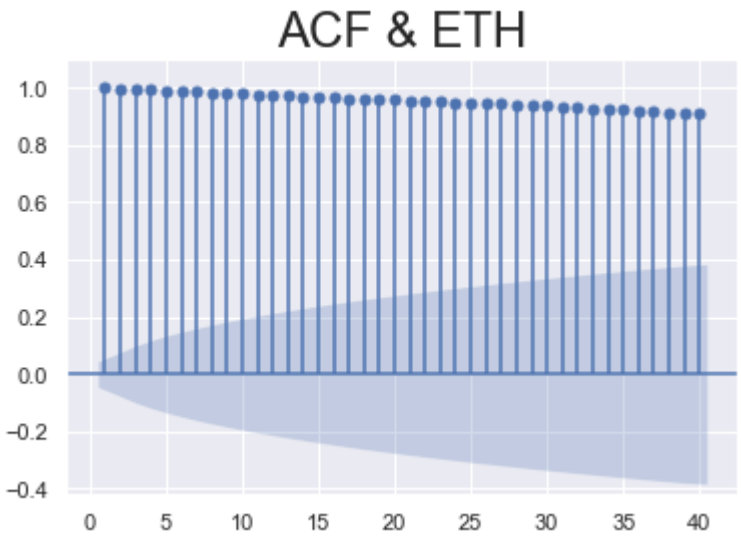
In [562…
```
#Very similar results in multiplicative test as well which shows that there is no seaso
s_dec_multiplicitive = seasonal_decompose(ethdata.Close, model = "Multiplicative")
s_dec_multiplicitive.plot()
plt.show()
```



ACF

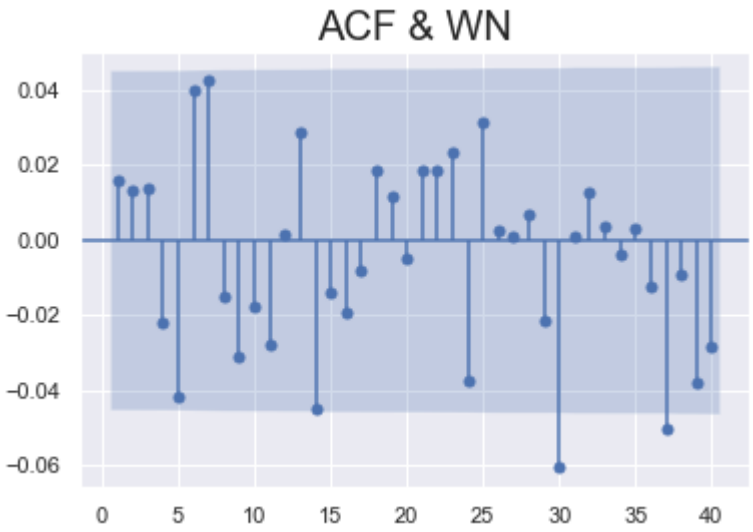
In [569…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(ethdata.Close, lags = 40, zero = False)
plt.title("ACF & ETH", size = 24)
plt.ylim(bottom = -0.42, top = 1.1)
```

Out[569… (-0.42, 1.1)



In [573…
```
sgt.plot_acf(ethdata.wn, lags = 40, zero = False)
plt.title("ACF & WN", size = 20)
plt.ylim(top = 0.05, bottom = -0.066)
```

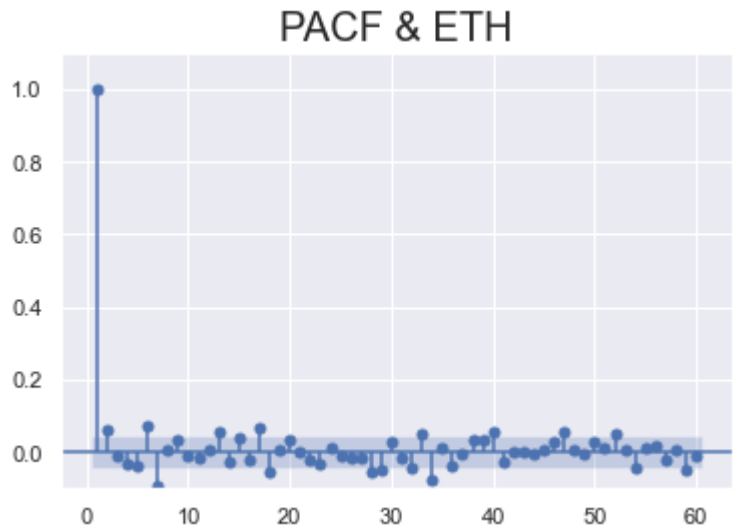
Out[573…
```
(-0.066, 0.05)
```



PACF

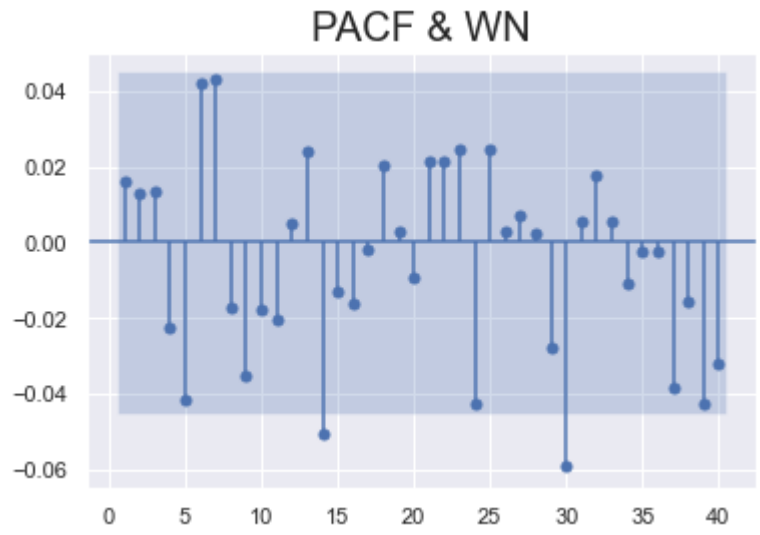
In [579…
```
sgt.plot_pacf(ethdata.Close, lags = 60, zero = False, method = ('ols'))
plt.title("PACF & ETH", size = 20)
plt.ylim(top = 1.10, bottom = -0.1)
```

Out[579…
```
(-0.1, 1.1)
```



In [583…
```
sgt.plot_pacf(ethdata.wn, lags = 40, zero = False, method = ('ols'))
plt.title("PACF & WN", size = 20)
plt.ylim(top = 0.05, bottom = -0.065)
```

Out[583…
```
(-0.065, 0.05)
```



LLR test function for models

In [40]:
```
def LLR_test(mod_1, mod_2, DF = 1):
    L1 = mod_1.fit().llf
    L2 = mod_2.fit().llf
    LR = (2*(L2-L1))
    p = chi2.sf(LR, DF).round(3)
    return p
```

In [29]:
```
import warnings
warnings.filterwarnings("ignore")
```

AR Model

In [30]:
```
from statsmodels.tsa.arima_process import ArmaProcess
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

```
from statsmodels.tsa.ar_model import AutoReg
```

In [31]:

In [588…
```
#ORDER first - AR, Second - Differences, Third - MA
ar_model = ARIMA(eth_train.Close[1:], order = (2,0,0))
results_ar_model = ar_model.fit()
results_ar_model.summary()
```

Out[588…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	1502
Model:	ARIMA(2, 0, 0)	**Log Likelihood**	-8630.028
Date:	Sat, 04 Feb 2023	**AIC**	17268.055
Time:	19:34:08	**BIC**	17289.313
Sample:	11-10-2017	**HQIC**	17275.974
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	901.5974	4705.739	0.192	0.848	-8321.481	1.01e+04
ar.L1	0.8998	0.013	71.623	0.000	0.875	0.924
ar.L2	0.0993	0.013	7.898	0.000	0.075	0.124
sigma2	5706.7400	61.354	93.014	0.000	5586.489	5826.991

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	35366.99
Prob(Q):	0.99	**Prob(JB):**	0.00
Heteroskedasticity (H):	13.85	**Skew:**	-0.96
Prob(H) (two-sided):	0.00	**Kurtosis:**	26.69

Warnings:

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

In [591…
```
#ORDER first - AR, Second - Differences, Third - MA
ar_model_1 = ARIMA(eth_train.Close[1:], order = (3,0,0))
results_ar_model_1 = ar_model_1.fit()
results_ar_model_1.summary()
```

Out[591…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	1502
Model:	ARIMA(3, 0, 0)	**Log Likelihood**	-8630.011
Date:	Sat, 04 Feb 2023	**AIC**	17270.021
Time:	19:36:07	**BIC**	17296.594

Sample:	11-10-2017	**HQIC**	17279.920
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	901.5950	4672.272	0.193	0.847	-8255.890	1.01e+04
ar.L1	0.9003	0.013	70.713	0.000	0.875	0.925
ar.L2	0.1035	0.015	6.842	0.000	0.074	0.133
ar.L3	-0.0047	0.010	-0.453	0.651	-0.025	0.016
sigma2	5706.3579	63.493	89.874	0.000	5581.914	5830.802

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	35121.95
Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	13.85	**Skew:**	-0.95
Prob(H) (two-sided):	0.00	**Kurtosis:**	26.61

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

In [590…
```
LLR_test(ar_model, ar_model_1)
```

Out[590…
```
0.851
```

As our data is coming from Non-Stationary data we wulll transform our data using returns. Returns are percentage change between the values for 2 consecutive periods

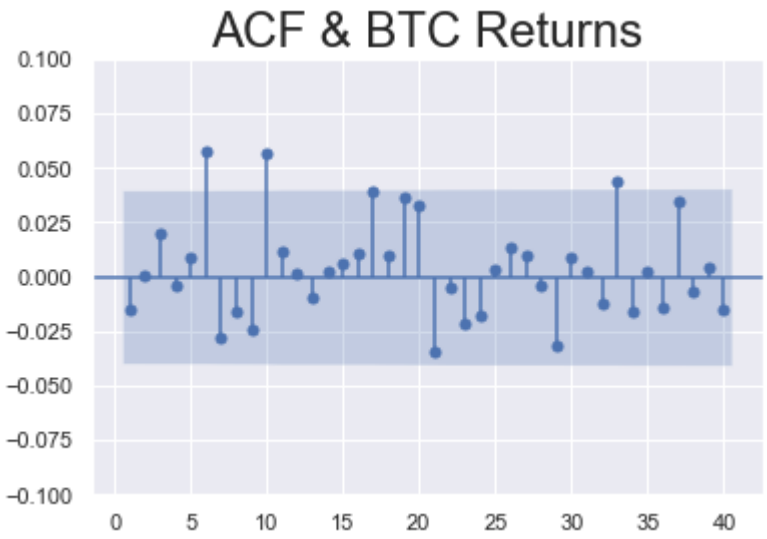
In [592…
```
eth_train['returns'] = eth_train.Close.pct_change(1).mul(100)
eth_test['returns'] = eth_test.Close.pct_change(1).mul(100)
eth_train = eth_train.iloc[1:]
```

In [593…
```
sts.adfuller(eth_train.returns)
```

Out[593…
```
(-11.412390521316967,
 7.197384243781305e-21,
 9,
 1492,
 {'1%': -3.434740473427213,
  '5%': -2.863479112458789,
  '10%': -2.5678023610641922},
 9054.48520801563)
```

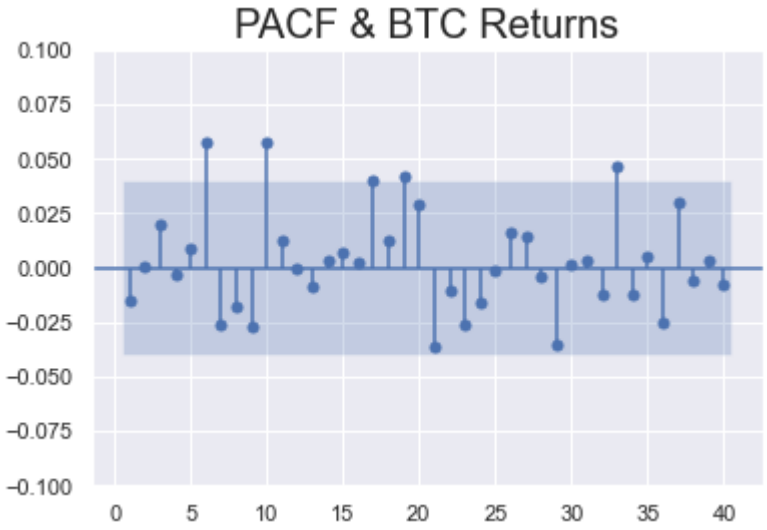
In [41]:
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.returns, lags = 40, zero = False)
plt.title("ACF & BTC Returns", size = 24)
plt.ylim(top = 0.1, bottom = -0.1)
```

Out[41]:
```
(-0.1, 0.1)
```



In [42]:
```
sgt.plot_pacf(btc_train.returns, lags = 40, zero = False, method = ('ols'))
plt.title("PACF & BTC Returns", size = 20)
plt.ylim(top = 0.1, bottom = -0.1)
```

Out[42]: (-0.1, 0.1)



In [43]:
```
#ORDER first - AR, Second - Differences, Third - MA
ar_model_ret_1 = ARIMA(btc_train.returns[1:], order = (1,0,0))
results_ar_model_ret_1 = ar_model_ret_1.fit()
results_ar_model_ret_1.summary()
```

Out[43]:

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	2420
Model:	ARIMA(1, 0, 0)	**Log Likelihood**	-6714.387
Date:	Wed, 01 Feb 2023	**AIC**	13434.773
Time:	05:56:37	**BIC**	13452.148
Sample:	09-19-2014	**HQIC**	13441.092
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.2760	0.078	3.537	0.000	0.123	0.429
ar.L1	-0.0163	0.013	-1.301	0.193	-0.041	0.008
sigma2	15.0485	0.195	77.297	0.000	14.667	15.430

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	6586.75
Prob(Q):	1.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.50	**Skew:**	-0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	11.08

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

In [44]:
```
#ORDER first - AR, Second - Differences, Third - MA
ar_model_ret_2 = ARIMA(btc_train.returns[1:], order = (7,0,0))
results_ar_model_ret_2 = ar_model_ret_2.fit()
results_ar_model_ret_2.summary()
```

Out[44]:

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	2420
Model:	ARIMA(7, 0, 0)	**Log Likelihood**	-6708.987
Date:	Wed, 01 Feb 2023	**AIC**	13435.973
Time:	05:56:38	**BIC**	13488.097
Sample:	09-19-2014	**HQIC**	13454.928
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.2759	0.083	3.307	0.001	0.112	0.439
ar.L1	-0.0153	0.013	-1.186	0.236	-0.040	0.010
ar.L2	0.0015	0.016	0.091	0.927	-0.030	0.033
ar.L3	0.0184	0.017	1.106	0.269	-0.014	0.051
ar.L4	-0.0024	0.016	-0.154	0.878	-0.033	0.028
ar.L5	0.0117	0.016	0.715	0.475	-0.020	0.044
ar.L6	0.0567	0.016	3.497	0.000	0.025	0.088
ar.L7	-0.0267	0.015	-1.818	0.069	-0.055	0.002
sigma2	14.9802	0.207	72.305	0.000	14.574	15.386

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	6290.78

Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.51	**Skew:**	-0.17
Prob(H) (two-sided):	0.00	**Kurtosis:**	10.89

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

In [45]:
```
LLR_test(ar_model_ret_1, ar_model_ret_2)
```

Out[45]:
```
0.001
```

Normalizing Values

In [46]:
```
benchmark = btc_train.Close.iloc[0]
```

In [47]:
```
btc_train['norm'] = btc_train.Close.div(benchmark).mul(100)
btc_test['norm'] = btc_test.Close.div(benchmark).mul(100)
```

In [48]:
```
sts.adfuller(btc_train.norm)
```

Out[48]:
```
(2.72497449552523,
 0.9990879280496889,
 27,
 2393,
 {'1%': -3.4330856157531304,
  '5%': -2.862748556756925,
  '10%': -2.567413365583104},
 29941.910212222094)
```

Normalizing Returns

In [49]:
```
benchmark_ret = btc_train.returns.iloc[0]
btc_train['norm_ret'] = btc_train.returns.div(benchmark_ret).mul(100)
```

In [50]:
```
sts.adfuller(btc_train.norm_ret)
```

Out[50]:
```
(-14.841520071127832,
 1.8340338853570292e-27,
 9,
 2411,
 {'1%': -3.433065170436013,
  '5%': -2.8627395284138935,
  '10%': -2.567408558721166},
 25877.467540695696)
```

In [51]:
```
### ORDER first - AR, Second - Differences, Third - MA
ar_model_norm_ret_1 = ARIMA(btc_train.norm_ret, order = (6,0,0))
results_ar_model_norm_ret_1 = ar_model_norm_ret_1.fit()
results_ar_model_norm_ret_1.summary()
```

Out[51]:

SARIMAX Results

Dep. Variable:	norm_ret	**No. Observations:**	2421
Model:	ARIMA(6, 0, 0)	**Log Likelihood**	-13086.337

Date:	Wed, 01 Feb 2023	**AIC**	26188.674
Time:	05:56:40	**BIC**	26235.009
Sample:	09-18-2014	**HQIC**	26205.523
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.7937	1.189	-3.191	0.001	-6.124	-1.464
ar.L1	-0.0153	0.013	-1.205	0.228	-0.040	0.010
ar.L2	0.0006	0.016	0.036	0.972	-0.031	0.032
ar.L3	0.0190	0.017	1.142	0.253	-0.014	0.052
ar.L4	-0.0030	0.016	-0.194	0.846	-0.034	0.028
ar.L5	0.0100	0.016	0.615	0.539	-0.022	0.042
ar.L6	0.0577	0.016	3.608	0.000	0.026	0.089
sigma2	2892.8194	39.042	74.096	0.000	2816.299	2969.340

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	6385.47
Prob(Q):	0.94	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.50	**Skew:**	0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	10.95

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

In [52]:
```
#ORDER first - AR, Second - Differences, Third - MA
ar_model_norm_ret_2 = ARIMA(btc_train.norm_ret, order = (10,0,0))
results_ar_model_norm_ret_2 = ar_model_norm_ret_2.fit()
results_ar_model_norm_ret_2.summary()
```

Out[52]:

SARIMAX Results

Dep. Variable:	norm_ret	**No. Observations:**	2421
Model:	ARIMA(10, 0, 0)	**Log Likelihood**	-13080.232
Date:	Wed, 01 Feb 2023	**AIC**	26184.464
Time:	05:56:42	**BIC**	26253.968
Sample:	09-18-2014	**HQIC**	26209.738
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]

const	-3.7937	1.188	-3.193	0.001	-6.122	-1.465
ar.L1	-0.0131	0.013	-1.023	0.307	-0.038	0.012
ar.L2	0.0021	0.016	0.130	0.897	-0.030	0.034
ar.L3	0.0223	0.017	1.330	0.183	-0.011	0.055
ar.L4	-0.0056	0.016	-0.355	0.723	-0.036	0.025
ar.L5	0.0097	0.017	0.581	0.561	-0.023	0.042
ar.L6	0.0579	0.016	3.580	0.000	0.026	0.090
ar.L7	-0.0274	0.015	-1.868	0.062	-0.056	0.001
ar.L8	-0.0185	0.018	-1.015	0.310	-0.054	0.017
ar.L9	-0.0262	0.018	-1.465	0.143	-0.061	0.009
ar.L10	0.0578	0.017	3.344	0.001	0.024	0.092
sigma2	2881.0588	39.771	72.441	0.000	2803.109	2959.009

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	6307.27
Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.50	**Skew:**	0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	10.90

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

In [53]:
```
LLR_test(ar_model_norm_ret_1, ar_model_norm_ret_2)
```

Out[53]:
```
0.0
```

AR model for normalized returns

AR RESIDUALS Price

In [54]:
```
btc_train['res_price'] = results_ar_model_1.resid
btc_test['res_price'] = results_ar_model_1.resid
```

In [55]:
```
btc_train.res_price.mean()
```

Out[55]:
```
16.63053668795223
```

In [56]:
```
btc_train.res_price.var()
```

Out[56]:
```
318652.7922050983
```

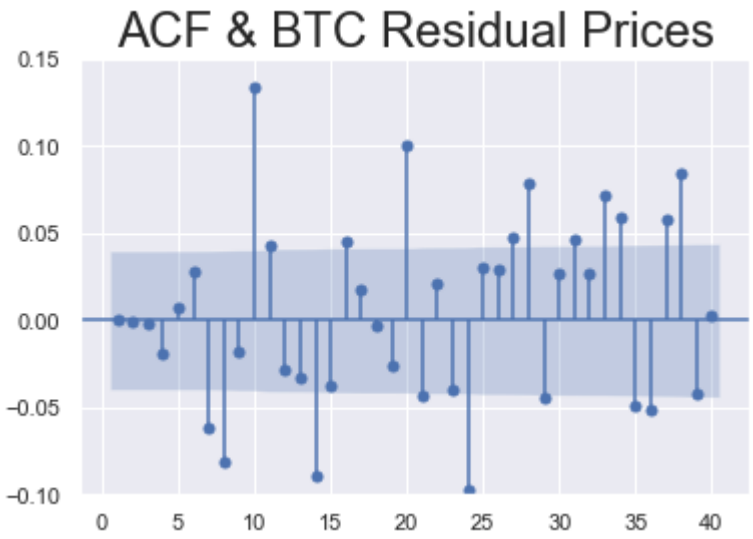
In [57]:
```
sts.adfuller(btc_train.res_price[1:])
```

Out[57]:
```
(-8.266610060180582,
 4.918949933531937e-13,
```

```
27,
2392,
{'1%': -3.4330867606360274,
 '5%': -2.862749062318083,
 '10%': -2.5674136347538057},
36841.16941469677)
```

In [58]:
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.res_price[1:], lags = 40, zero = False)
plt.title("ACF & BTC Residual Prices", size = 24)
plt.ylim(top = 0.15, bottom = -0.1)
```

Out[58]: (-0.1, 0.15)



In [59]:
```
btc_train.res_price[1:].plot(figsize = (22,8))
plt.title("Residuals of Prices", size=22)
```

Out[59]: Text(0.5, 1.0, 'Residuals of Prices')

In [60]:
```
btc_train['res_price_ret'] = results_ar_model_ret_2.resid
btc_test['res_price_ret'] = results_ar_model_ret_2.resid
```

In [61]:
```
btc_train.res_price_ret.mean()
```

Out[61]: -3.7530309792591246e-05

In [62]:
```
btc_train.res_price_ret.var()
```

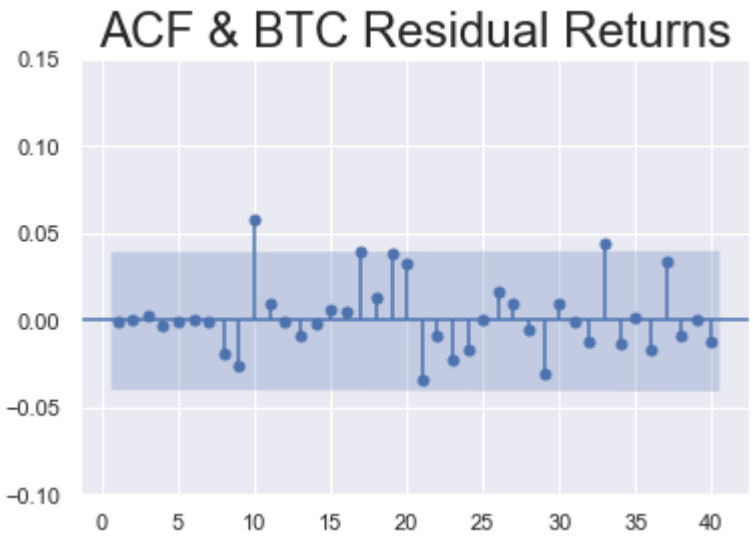
Out[62]:
```
14.986649487280625
```

In [63]:
```
sts.adfuller(btc_train.res_price_ret[1:])
```

Out[63]:
```
(-49.191126014559075,
 0.0,
 0,
 2419,
 {'1%': -3.4330561813996505,
  '5%': -2.8627355589717505,
  '10%': -2.567406445317754},
 13267.654510963248)
```

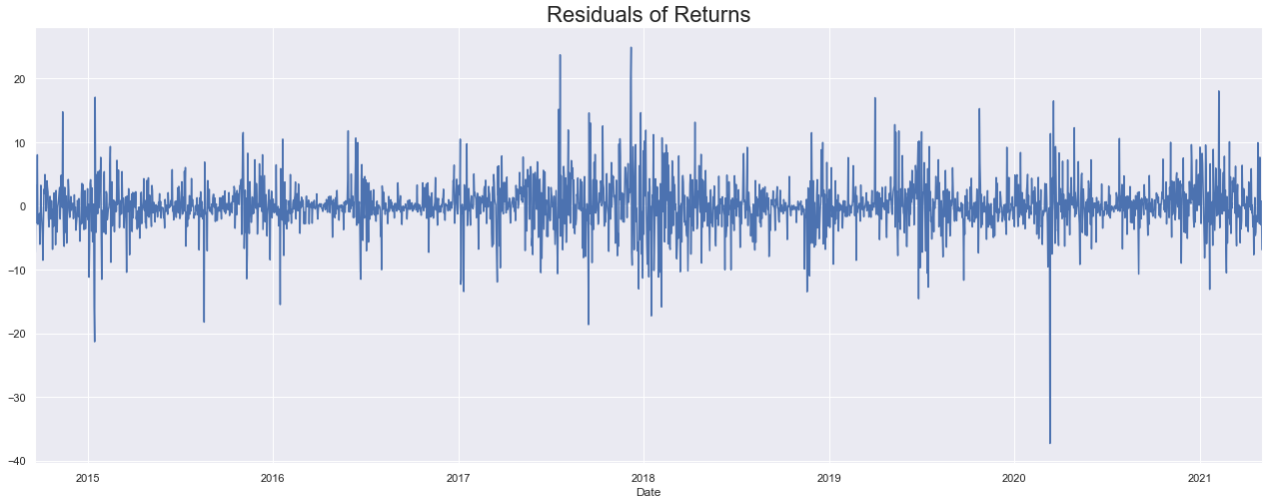
In [64]:
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.res_price_ret[1:], lags = 40, zero = False)
plt.title("ACF & BTC Residual Returns", size = 24)
plt.ylim(top = 0.15, bottom = -0.1)
```

Out[64]:
```
(-0.1, 0.15)
```



In [65]:
```
btc_train.res_price_ret[1:].plot(figsize = (22,8))
plt.title("Residuals of Returns", size=22)
```

Out[65]:
```
Text(0.5, 1.0, 'Residuals of Returns')
```



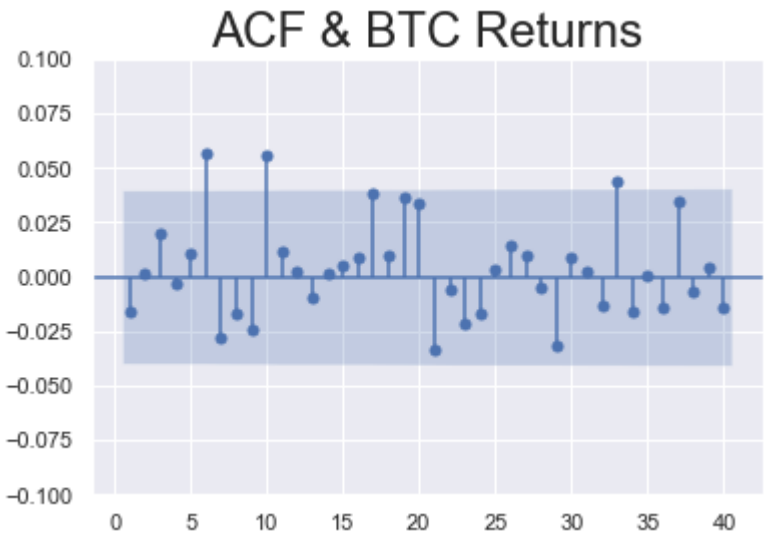
In [ ]:

MA model

In [161…
```
from statsmodels.tsa.arima_process import ArmaProcess
```

In [206…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.returns, lags = 40, zero = False)
plt.title("ACF & BTC Returns", size = 24)
plt.ylim(top = 0.1, bottom = -0.1)
```

Out[206… (-0.1, 0.1)



In [91]:
```
#ORDER first - AR, Second - Differences, Third - MA
model_ret_ma_1 = ARIMA(btc_train.returns[1:], order = (0,0,7))
results_ret_ma_1 = model_ret_ma_1.fit()
```

In [92]:
```
results_ret_ma_1.summary()
```

Out[92]:

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	2420

Model:	ARIMA(0, 0, 7)	**Log Likelihood**	-6708.664
Date:	Wed, 01 Feb 2023	**AIC**	13435.328
Time:	06:28:17	**BIC**	13487.452
Sample:	09-19-2014	**HQIC**	13454.283
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.2759	0.083	3.329	0.001	0.113	0.438
ma.L1	-0.0161	0.013	-1.252	0.211	-0.041	0.009
ma.L2	0.0024	0.016	0.149	0.882	-0.030	0.034
ma.L3	0.0248	0.017	1.489	0.136	-0.008	0.058
ma.L4	-0.0089	0.016	-0.574	0.566	-0.039	0.022
ma.L5	0.0103	0.016	0.625	0.532	-0.022	0.042
ma.L6	0.0581	0.016	3.583	0.000	0.026	0.090
ma.L7	-0.0308	0.015	-2.118	0.034	-0.059	-0.002
sigma2	14.9762	0.207	72.339	0.000	14.570	15.382

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	6291.73
Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.51	**Skew:**	-0.17
Prob(H) (two-sided):	0.00	**Kurtosis:**	10.89

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

In [228…
```
from statsmodels.tsa.arima.model import ARIMA
```

In [87]:
```
#ORDER first - AR, Second - Differences, Third - MA
model_ret_ma_2 = ARIMA(btc_train.returns[1:], order = (0,0,10))
results_ret_ma_2 = model_ret_ma_2.fit()
results_ret_ma_2.summary()
print("\nLLR Test P-value = " + str(LLR_test(model_ret_ma_1,model_ret_ma_2, DF = 3)))
```

Out[87]:

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	2420
Model:	ARIMA(0, 0, 10)	**Log Likelihood**	-6703.508
Date:	Wed, 01 Feb 2023	**AIC**	13431.016
Time:	06:21:35	**BIC**	13500.515

Sample:	09-19-2014	**HQIC**	13456.289
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.2758	0.085	3.257	0.001	0.110	0.442
ma.L1	-0.0161	0.013	-1.233	0.218	-0.042	0.009
ma.L2	0.0035	0.016	0.214	0.830	-0.029	0.036
ma.L3	0.0236	0.017	1.402	0.161	-0.009	0.057
ma.L4	-0.0052	0.016	-0.328	0.743	-0.036	0.026
ma.L5	0.0089	0.017	0.535	0.593	-0.024	0.041
ma.L6	0.0563	0.016	3.468	0.001	0.024	0.088
ma.L7	-0.0332	0.015	-2.271	0.023	-0.062	-0.005
ma.L8	-0.0166	0.018	-0.910	0.363	-0.052	0.019
ma.L9	-0.0273	0.018	-1.529	0.126	-0.062	0.008
ma.L10	0.0583	0.018	3.317	0.001	0.024	0.093
sigma2	14.9123	0.207	71.930	0.000	14.506	15.319

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	6322.16
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.51	**Skew:**	-0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	10.91

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

In [93]:
```
LLR_test(model_ret_ma_1,model_ret_ma_2, DF = 3)
```

Out[93]:
```
0.016
```

Residual Analysis

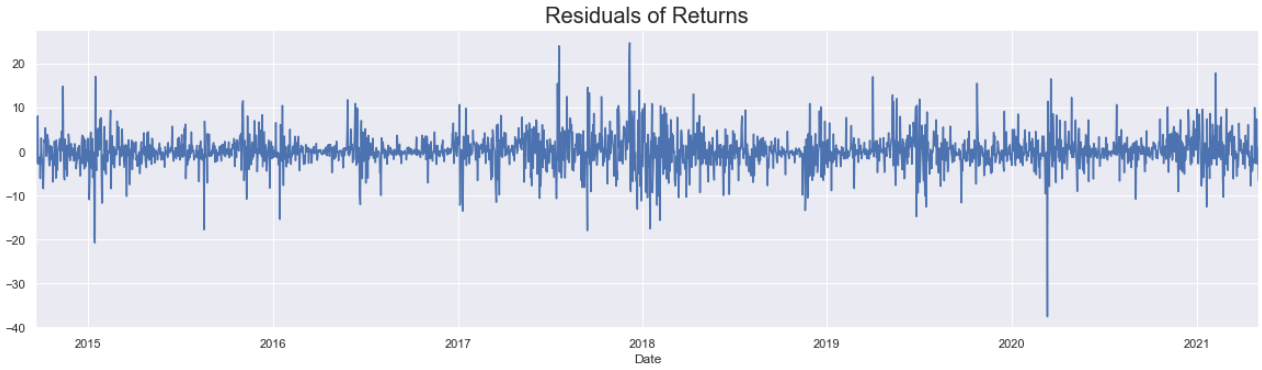
In [94]:
```
btc_train['res_ret_ma_2'] = results_ret_ma_2.resid[1:]
```

In [100…
```
print("mean is " + str(round(btc_train.res_ret_ma_2.mean(),3)))
print("variance is " + str(round(btc_train.res_ret_ma_2.var(),3)))
print("Standard deviation is " + str(round(sqrt(btc_train.res_ret_ma_2.var()), 3)))
```

```
mean is 0.003
variance is 14.903
Standard deviation is 3.86
```

In [101…
```
btc_train.res_ret_ma_2[1:].plot(figsize = (20,5))
plt.title("Residuals of Returns", size = 20)
plt.ylim(bottom = -40)
```

Out[101…
```
(-40.0, 27.740790855171557)
```

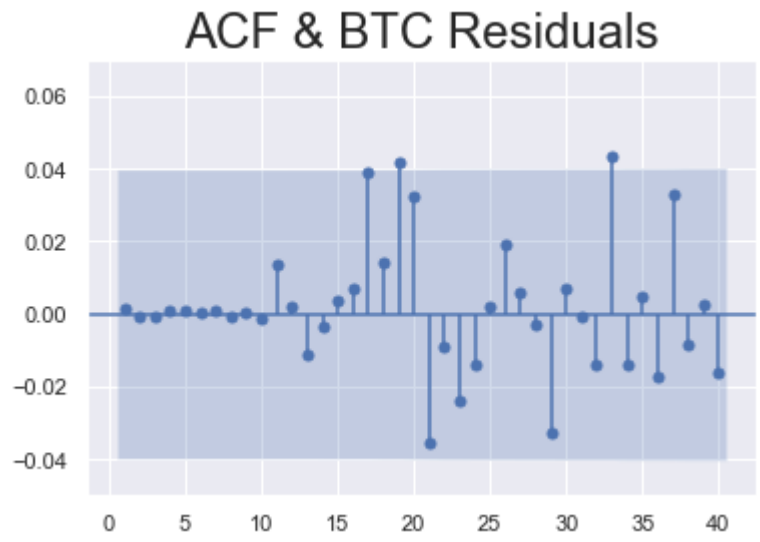


In [102…
```
sts.adfuller(btc_train.res_ret_ma_2[2:])
```

Out[102…
```
(-49.055967881609746,
 0.0,
 0,
 2418,
 {'1%': -3.4330573017728736,
  '5%': -2.8627360537147197,
  '10%': -2.5674067087278276},
 13251.566606107284)
```

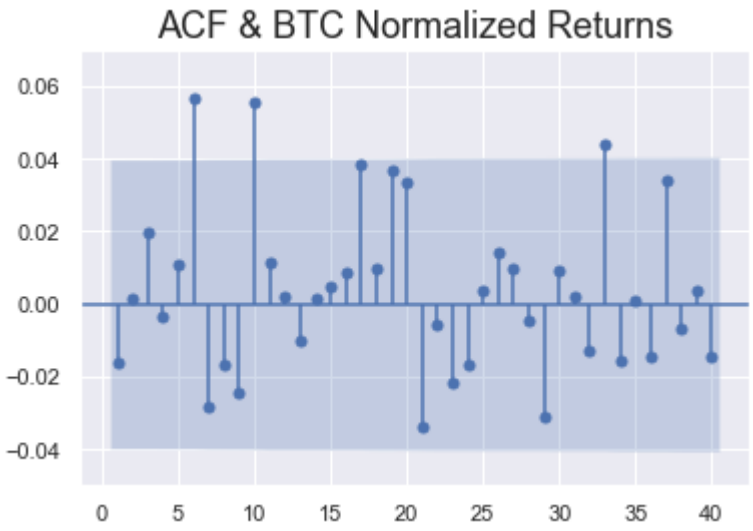
In [103…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.res_ret_ma_2[2:], lags = 40, zero = False)
plt.title("ACF & BTC Residuals", size = 24)
plt.ylim(top = 0.07, bottom = -0.05)
```

Out[103…
```
(-0.05, 0.07)
```



In [105…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.norm_ret[1:], lags = 40, zero = False)
plt.title("ACF & BTC Normalized Returns", size = 18)
plt.ylim(top = 0.07, bottom = -0.05)
```

Out[105… (-0.05, 0.07)



In [120…
```
#ORDER first - AR, Second - Differences, Third - MA
model_norm_ret_ma_1 = ARIMA(btc_train.norm_ret[1:], order = (0,0,10))
results_norm_ret_ma_1 = model_norm_ret_ma_1.fit()
results_norm_ret_ma_1.summary()
```

Out[120…

SARIMAX Results

Dep. Variable:	norm_ret	**No. Observations:**	2420
Model:	ARIMA(0, 0, 10)	**Log Likelihood**	-13073.251
Date:	Wed, 01 Feb 2023	**AIC**	26170.503
Time:	08:03:41	**BIC**	26240.001
Sample:	09-19-2014	**HQIC**	26195.775
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.8366	1.181	-3.249	0.001	-6.151	-1.522
ma.L1	-0.0161	0.013	-1.230	0.219	-0.042	0.010
ma.L2	0.0035	0.017	0.213	0.831	-0.029	0.036
ma.L3	0.0236	0.017	1.399	0.162	-0.009	0.057
ma.L4	-0.0052	0.016	-0.328	0.743	-0.036	0.026
ma.L5	0.0089	0.017	0.534	0.593	-0.024	0.041
ma.L6	0.0563	0.016	3.458	0.001	0.024	0.088
ma.L7	-0.0332	0.015	-2.264	0.024	-0.062	-0.004
ma.L8	-0.0166	0.018	-0.907	0.364	-0.053	0.019
ma.L9	-0.0273	0.018	-1.525	0.127	-0.062	0.008
ma.L10	0.0583	0.018	3.308	0.001	0.024	0.093

sigma2	2890.5087	40.296	71.732	0.000	2811.530	2969.488

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	6322.17
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.51	**Skew:**	0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	10.91

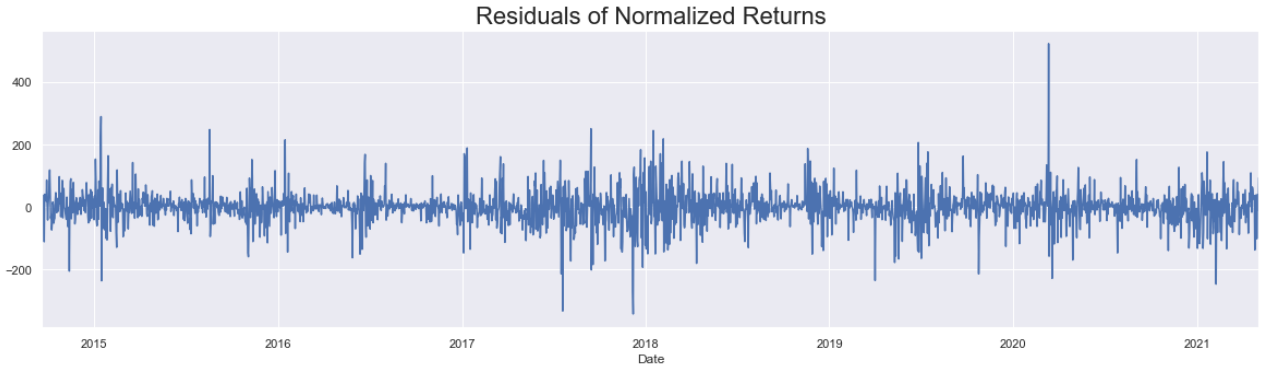
Warnings:

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

In [121…
```
btc_train['res_norm_ret_ma_1'] = results_norm_ret_ma_1.resid[1:]
```

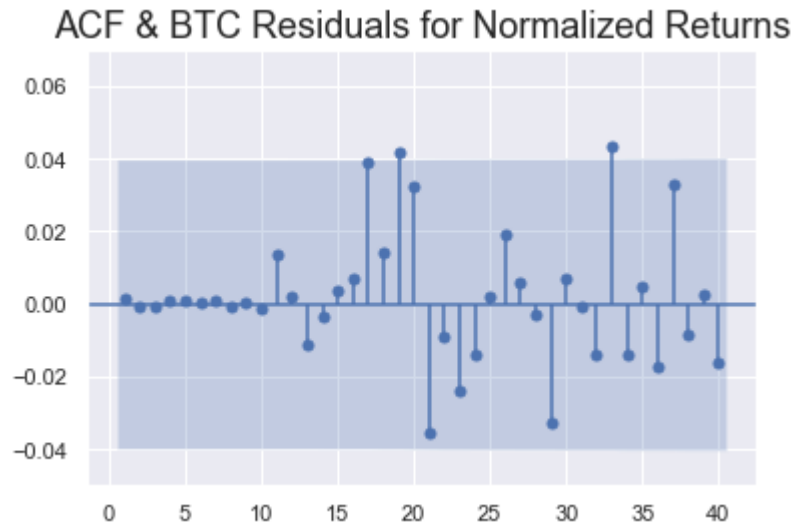
In [122…
```
btc_train.res_norm_ret_ma_1[1:].plot(figsize = (20,5))
plt.title("Residuals of Normalized Returns", size = 22)
```

Out[122…
```
Text(0.5, 1.0, 'Residuals of Normalized Returns')
```



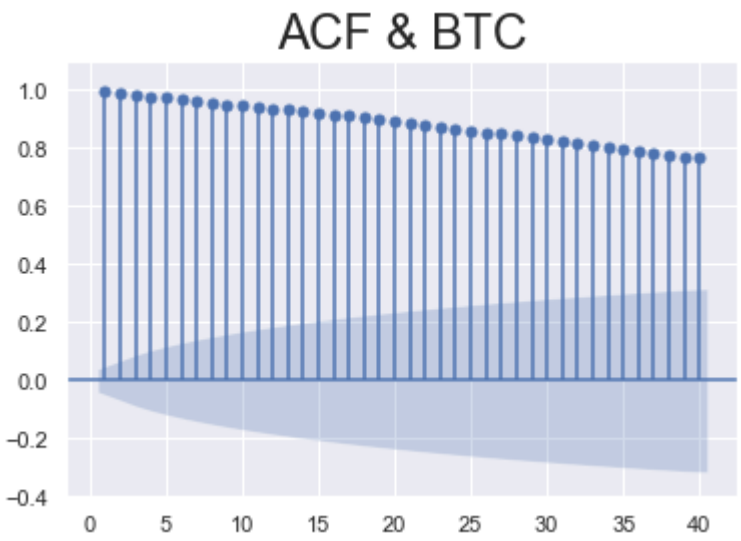
In [124…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.res_norm_ret_ma_1[2:], lags = 40, zero = False)
plt.title("ACF & BTC Residuals for Normalized Returns", size = 18)
plt.ylim(top = 0.07, bottom = -0.05)
```

Out[124…
```
(-0.05, 0.07)
```



In [128…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.Close, lags = 40, zero = False)
plt.title("ACF & BTC", size = 24)
plt.ylim(top = 1.10, bottom = -0.4)
```

Out[128… (-0.4, 1.1)



ARMA

In [140…
```
#ORDER first - AR, Second - Differences, Third - MA
model_ret_arma_1 = ARIMA(btc_train.returns[1:], order = (3,0,3))
results_ret_arma_1 = model_ret_arma_1.fit()
results_ret_arma_1.summary()
```

Out[140…

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	2420
Model:	ARIMA(3, 0, 3)	**Log Likelihood**	-6710.047
Date:	Wed, 01 Feb 2023	**AIC**	13436.095
Time:	15:39:32	**BIC**	13482.427
Sample:	09-19-2014	**HQIC**	13452.944
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.2680	0.089	3.020	0.003	0.094	0.442
ar.L1	0.3481	0.124	2.798	0.005	0.104	0.592
ar.L2	-0.4338	0.074	-5.878	0.000	-0.578	-0.289
ar.L3	0.9335	0.123	7.562	0.000	0.692	1.175
ma.L1	-0.3507	0.129	-2.719	0.007	-0.604	-0.098
ma.L2	0.4414	0.076	5.840	0.000	0.293	0.590

ma.L3	-0.9265	0.128	-7.233	0.000	-1.178	-0.675
sigma2	14.9923	0.201	74.551	0.000	14.598	15.386

Ljung-Box (L1) (Q):	0.43	**Jarque-Bera (JB):**	6416.59
Prob(Q):	0.51	**Prob(JB):**	0.00
Heteroskedasticity (H):	1.51	**Skew:**	-0.16
Prob(H) (two-sided):	0.00	**Kurtosis:**	10.97

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

In [162…
```
#ORDER first - AR, Second - Differences, Third - MA
model_ret_arma_2 = ARIMA(btc_train.returns[1:], order = (4,0,4))
results_ret_arma_2 = model_ret_arma_2.fit()
results_ret_arma_2.summary()
```

Out[162…

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	2420
Model:	ARIMA(4, 0, 4)	**Log Likelihood**	-6710.810
Date:	Wed, 01 Feb 2023	**AIC**	13441.620
Time:	15:59:08	**BIC**	13499.535
Sample:	09-19-2014	**HQIC**	13462.681
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.2791	0.079	3.555	0.000	0.125	0.433
ar.L1	-0.5407	1.559	-0.347	0.729	-3.596	2.514
ar.L2	-0.5609	1.381	-0.406	0.685	-3.267	2.145
ar.L3	-0.5386	1.432	-0.376	0.707	-3.345	2.267
ar.L4	0.3307	1.319	0.251	0.802	-2.255	2.916
ma.L1	0.5229	1.554	0.336	0.737	-2.523	3.569
ma.L2	0.5716	1.357	0.421	0.674	-2.088	3.232
ma.L3	0.5357	1.450	0.370	0.712	-2.305	3.377
ma.L4	-0.3441	1.327	-0.259	0.795	-2.945	2.257
sigma2	15.0146	0.199	75.288	0.000	14.624	15.405

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	6719.47
Prob(Q):	0.92	**Prob(JB):**	0.00

Heteroskedasticity (H):	1.50	**Skew:**	-0.15
Prob(H) (two-sided):	0.00	**Kurtosis:**	11.16

Warnings:

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

In [163…
```
LLR_test(model_ret_arma_1, model_ret_arma_2, DF = 2)
```

Out[163…
```
1.0
```

In [172…
```
print("ARMA(4,4) AIC Value is " + str(results_ret_arma_2.aic))
print("ARMA(3,3) AIC Value is " + str(results_ret_arma_1.aic))
```

```
ARMA(4,4) AIC Value is 13441.619914324332
ARMA(3,3) AIC Value is 13436.094945896086
```

In [165…
```
results_ret_arma_2.llf
```

Out[165…
```
-6710.809957162166
```

In [166…
```
results_ret_arma_1.aic
```

Out[166…
```
13436.094945896086
```

In [167…
```
results_ret_arma_1.llf
```

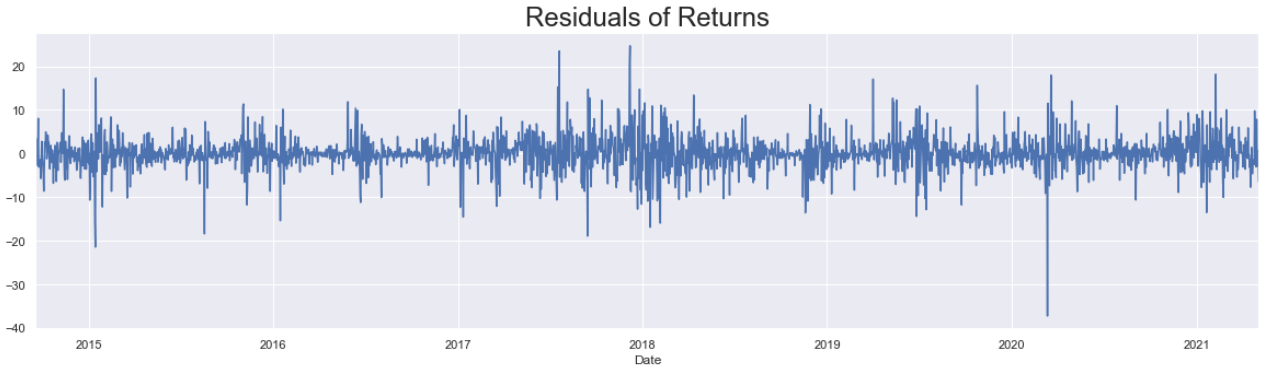
Out[167…
```
-6710.047472948043
```

Residuals for Returns

In [173…
```
btc_train['res_ret_arma_1'] = results_ret_arma_1.resid[1:]
```

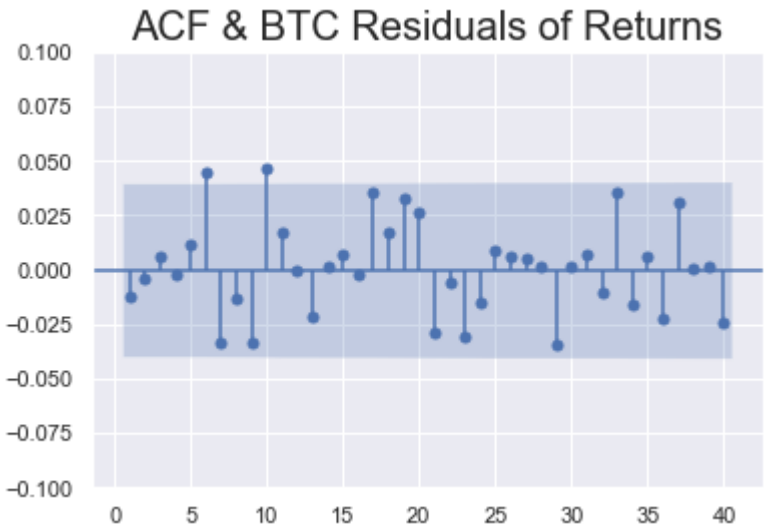
In [174…
```
btc_train.res_ret_arma_1.plot(figsize = (20,5))
plt.title('Residuals of Returns', size = 24)
```

Out[174…
```
Text(0.5, 1.0, 'Residuals of Returns')
```



In [177…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.res_ret_arma_1[2:], lags = 40, zero = False)
plt.title("ACF & BTC Residuals of Returns", size = 20)
plt.ylim(top = 0.1, bottom = -0.1)
```

Out[177… (-0.1, 0.1)

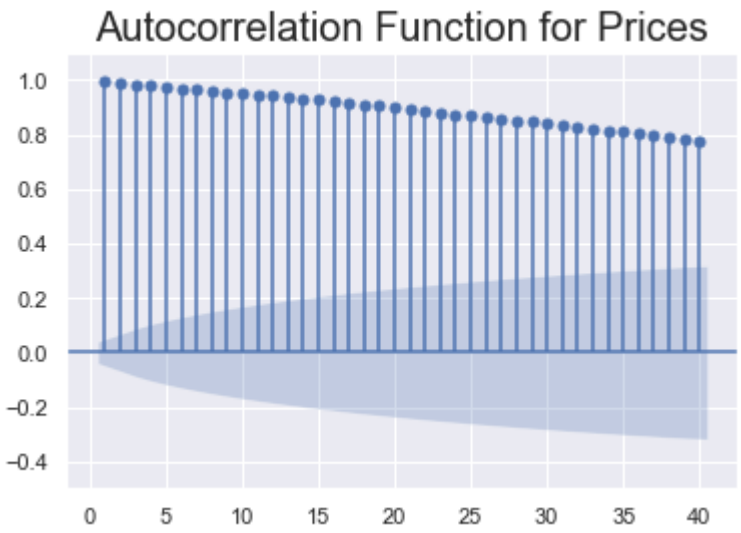


if the model is not sufficient try with different lags in ARMA and follow above steps again Reevaluating Model

ARMA MODEL FOR Close Prices

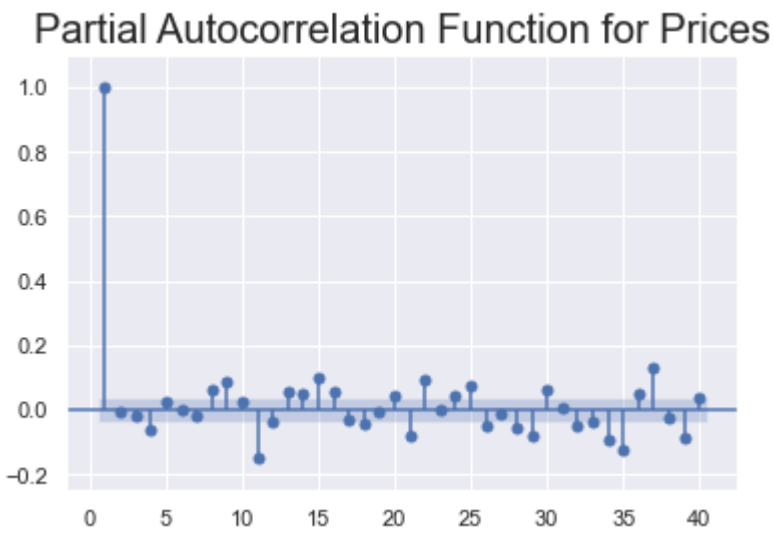
In [179…
```
sgt.plot_acf(btc_train.Close, unbiased=True, zero = False, lags = 40)
plt.title("Autocorrelation Function for Prices",size=20)
plt.ylim(top = 1.1, bottom = -0.5)
```

Out[179… (-0.5, 1.1)



In [181…
```
sgt.plot_pacf(btc_train.Close, lags = 40, alpha = 0.05, zero = False , method = ('ols'))
plt.title("Partial Autocorrelation Function for Prices",size=20)
plt.ylim(top = 1.1, bottom = -0.25)
```

Out[181… (-0.25, 1.1)



In [189…

```
#ORDER first - AR, Second - Differences, Third - MA
model_arma_1 = ARIMA(btc_train.Close, order = (3,0,3))
results_arma_1 = model_arma_1.fit()
results_arma_1.summary()
```

Out[189…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	2421
Model:	ARIMA(3, 0, 3)	**Log Likelihood**	-18695.052
Date:	Wed, 01 Feb 2023	**AIC**	37406.104
Time:	16:23:31	**BIC**	37452.439
Sample:	09-18-2014	**HQIC**	37422.953
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	7170.5465	1.09e-10	6.6e+13	0.000	7170.547	7170.547
ar.L1	-0.5941	0.006	-102.277	0.000	-0.605	-0.583
ar.L2	0.6075	0.004	155.027	0.000	0.600	0.615
ar.L3	0.9865	0.006	164.015	0.000	0.975	0.998
ma.L1	1.6159	0.010	161.532	0.000	1.596	1.636
ma.L2	1.0141	0.016	63.318	0.000	0.983	1.046
ma.L3	0.0293	0.009	3.353	0.001	0.012	0.046
sigma2	3.008e+05	7.32e-09	4.11e+13	0.000	3.01e+05	3.01e+05

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	146715.78
Prob(Q):	0.99	**Prob(JB):**	0.00
Heteroskedasticity (H):	4776.46	**Skew:**	1.35
Prob(H) (two-sided):	0.00	**Kurtosis:**	41.04

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 2.1e+30. Standard errors may be unstable.

In [188…
```
#ORDER first - AR, Second - Differences, Third - MA
model_arma_2 = ARIMA(btc_train.Close, order = (3,0,6))
results_arma_2 = model_arma_2.fit()
results_arma_2.summary()
```

Out[188…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	2421
Model:	ARIMA(3, 0, 6)	**Log Likelihood**	-18673.037
Date:	Wed, 01 Feb 2023	**AIC**	37368.073
Time:	16:23:28	**BIC**	37431.785
Sample:	09-18-2014	**HQIC**	37391.241
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	7170.5465	6.440	1113.375	0.000	7157.924	7183.169
ar.L1	0.5315	0.005	116.439	0.000	0.523	0.540
ar.L2	-0.4875	0.005	-95.766	0.000	-0.497	-0.477
ar.L3	0.9557	0.004	238.521	0.000	0.948	0.964
ma.L1	0.4855	0.010	48.887	0.000	0.466	0.505
ma.L2	1.0109	0.009	108.931	0.000	0.993	1.029
ma.L3	0.0918	0.011	7.988	0.000	0.069	0.114
ma.L4	0.0392	0.011	3.644	0.000	0.018	0.060
ma.L5	0.0471	0.009	5.431	0.000	0.030	0.064
ma.L6	0.0278	0.008	3.474	0.001	0.012	0.043
sigma2	2.967e+05	2183.294	135.912	0.000	2.92e+05	3.01e+05

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	128907.71
Prob(Q):	0.94	**Prob(JB):**	0.00
Heteroskedasticity (H):	3812.82	**Skew:**	1.30
Prob(H) (two-sided):	0.00	**Kurtosis:**	38.65

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.14e+18. Standard errors may be unstable.

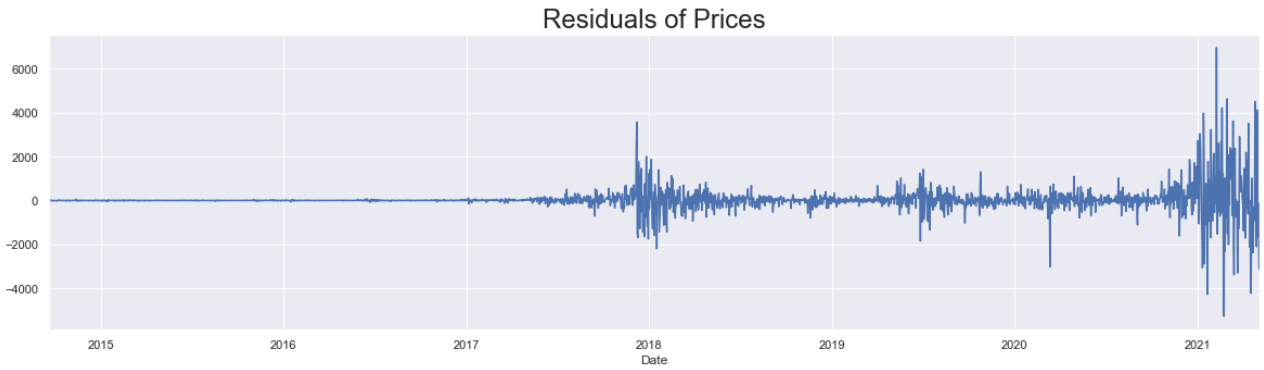
In [190…
```
LLR_test(model_arma_1, model_arma_2, DF = 3)
```

Out[190…
```
0.0
```

In [191…
```
btc_train['res_arma'] = results_arma_2.resid[1:]
```

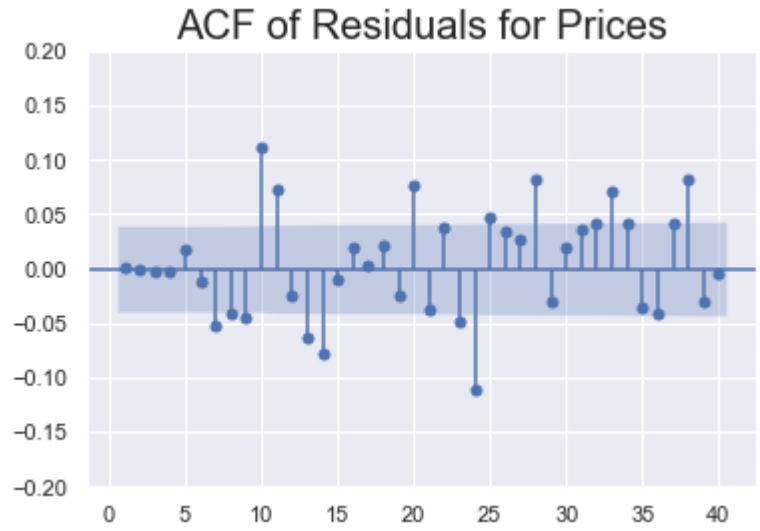
In [199…
```
btc_train.res_arma.plot(figsize = (20,5))
plt.title('Residuals of Prices', size = 24)
```

Out[199…
```
Text(0.5, 1.0, 'Residuals of Prices')
```



In [203…
```
sgt.plot_acf(btc_train.res_arma[1:], zero = False, lags = 40)
plt.title("ACF of Residuals for Prices",size=20)
plt.ylim(top = 0.2, bottom = -0.2)
```

Out[203…
```
(-0.2, 0.2)
```



Choose the best lags model and do AIC and LLF comparison as we have done in simple lag

ARIMA Model

In [217…
```
model_arima_1 = ARIMA(btc_train.Close[1:], order = (1,1,1))
results_model_arima1 = model_arima1.fit()
results_model_arima1.summary()
```

Out[217…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	2420
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-18690.930
Date:	Thu, 02 Feb 2023	**AIC**	37387.860
Time:	06:52:34	**BIC**	37405.234
Sample:	09-19-2014	**HQIC**	37394.178
	- 05-04-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.6822	0.084	8.149	0.000	0.518	0.846
ma.L1	-0.6547	0.088	-7.446	0.000	-0.827	-0.482
sigma2	3.007e+05	1939.343	155.030	0.000	2.97e+05	3.04e+05

Ljung-Box (L1) (Q):	0.71	**Jarque-Bera (JB):**	145640.80
Prob(Q):	0.40	**Prob(JB):**	0.00
Heteroskedasticity (H):	4958.80	**Skew:**	1.32
Prob(H) (two-sided):	0.00	**Kurtosis:**	40.92

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

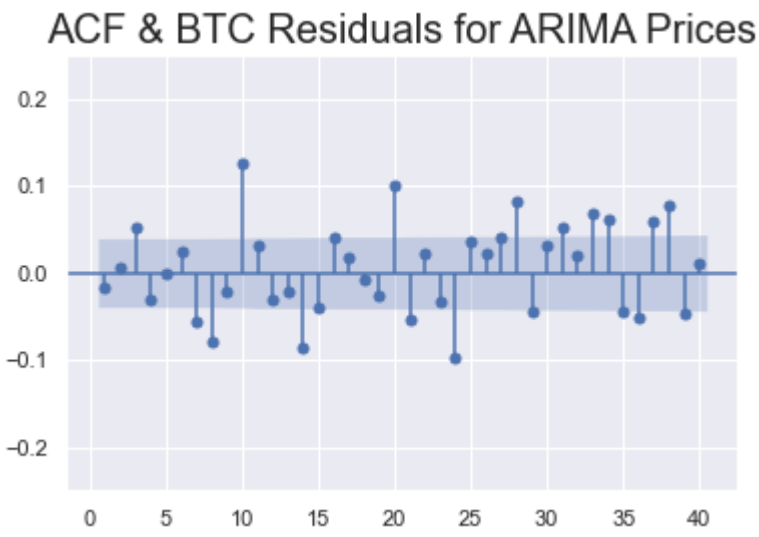
Residuals of simple ARIMA Model

In [208…
```
btc_train['results_model_arima1'] = results_model_arima1.resid[1:]
btc_train.results_model_arima1.plot(figsize = (20,5))
plt.title('Residuals of Close Prices', size = 24)
```

Out[208…
```
Text(0.5, 1.0, 'Residuals of Close Prices')
```

In [209…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.results_model_arima1[2:], lags = 40, zero = False)
plt.title("ACF & BTC Residuals for ARIMA Prices", size = 20)
plt.ylim(top = 0.25, bottom = -0.25)
```

Out[209…
```
(-0.25, 0.25)
```



In [222…
```
# 1
model_arima_1 = ARIMA(btc_train.Close[1:], order = (1,1,1))
results_model_arima1 = model_arima1.fit()
results_model_arima1.summary()
# 2
model_arima_2 = ARIMA(btc_train.Close[1:], order = (1,1,2))
results_model_arima_2 = model_arima_2.fit()
results_model_arima_2.summary()
# 3
model_arima_3 = ARIMA(btc_train.Close[1:], order = (1,1,3))
results_model_arima_3 = model_arima_3.fit()
results_model_arima_3.summary()
# 4
model_arima_4 = ARIMA(btc_train.Close[1:], order = (2,1,1))
results_model_arima_4 = model_arima_4.fit()
# 5
model_arima_5 = ARIMA(btc_train.Close[1:], order = (3,1,1))
results_model_arima_5 = model_arima_5.fit()

# 6
model_arima_6 = ARIMA(btc_train.Close[1:], order = (3,1,2))
results_model_arima_6 = model_arima_6.fit()
```

FIND LL AND AIC

In [224…
```
print("ARIMA(1,1,1):  \t LL = ", results_model_arima_1.llf, "\t AIC = ", results_model_
print("ARIMA(1,1,2):  \t LL = ", results_model_arima_2.llf, "\t AIC = ", results_model_
print("ARIMA(1,1,3):  \t LL = ", results_model_arima_3.llf, "\t AIC = ", results_model_
print("ARIMA(2,1,1):  \t LL = ", results_model_arima_4.llf, "\t AIC = ", results_model_
print("ARIMA(3,1,1):  \t LL = ", results_model_arima_5.llf, "\t AIC = ", results_model_
print("ARIMA(3,1,2):  \t LL = ", results_model_arima_6.llf, "\t AIC = ", results_model_
```

```
ARIMA(1,1,1):    LL =  -18690.930198249596       AIC =  37387.86039649919
ARIMA(1,1,2):    LL =  -18689.748010232594       AIC =  37387.49602046519
ARIMA(1,1,3):    LL =  -18686.541816054236       AIC =  37383.08363210847
```

```
ARIMA(2,1,1):    LL =  -18689.894258525324       AIC =  37387.78851705065
ARIMA(3,1,1):    LL =  -18686.098516218015       AIC =  37382.19703243603
ARIMA(3,1,2):    LL =  -18655.94414688421        AIC =  37323.88829376842
```

Do the LLR test with and without DF

In [228…
```
print("\nLLR test p-value = " + str(LLR_test(model_arima_5, model_arima_6)))
print("\nLLR test p-value = " + str(LLR_test(model_arima_4, model_arima_6, DF = 2)))
print("\nLLR test p-value = " + str(LLR_test(model_arima_2, model_arima_6, DF = 2)))
print("\nLLR test p-value = " + str(LLR_test(model_arima_1, model_arima_6, DF = 3)))
```

```
LLR test p-value = 0.0

LLR test p-value = 0.0

LLR test p-value = 0.0

LLR test p-value = 0.0
```

In [229…
```
btc_train['delta_prices']=btc_train.Close.diff(1)
```

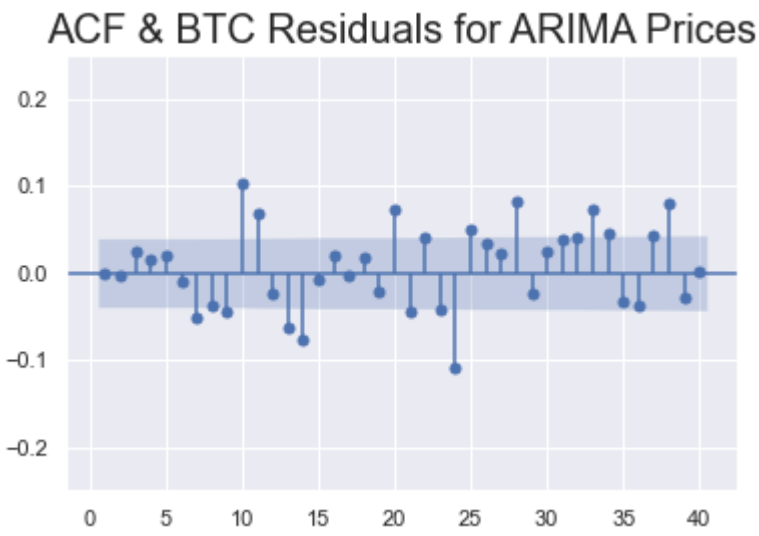
In [230…
```
sts.adfuller(btc_train.delta_prices[1:])
```

Out[230…
```
(-8.177641199656662,
 8.293887243613951e-13,
 27,
 2392,
 {'1%': -3.4330867606360274,
  '5%': -2.862749062318083,
  '10%': -2.5674136347538057},
 36839.442886560086)
```

In [231…
```
btc_train['res_arima'] = results_model_arima_6.resid[1:]
```

In [234…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btc_train.res_arima[2:], lags = 40, zero = False)
plt.title("ACF & BTC Residuals for ARIMA Prices", size = 20)
plt.ylim(top = 0.25, bottom = -0.25)
```

Out[234…
```
(-0.25, 0.25)
```



In [246…
```
btc_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2421 entries, 2014-09-18 to 2021-05-04
Freq: D
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Open                  2421 non-null   float64
 1   High                  2421 non-null   float64
 2   Low                   2421 non-null   float64
 3   Close                 2421 non-null   float64
 4   Volume                2421 non-null   int64  
 5   returns               2421 non-null   float64
 6   norm                  2421 non-null   float64
 7   norm_ret              2421 non-null   float64
 8   res_price             2421 non-null   float64
 9   res_price_ret         2420 non-null   float64
 10  res_ret_ma_2          2419 non-null   float64
 11  res_norm_ret_ma_1     2419 non-null   float64
 12  res_ret_arma_1        2419 non-null   float64
 13  res_arma              2420 non-null   float64
 14  results_model_arima1  2419 non-null   float64
 15  delta_prices          2420 non-null   float64
 16  res_arima             2419 non-null   float64
dtypes: float64(16), int64(1)
memory usage: 340.5 KB
```

In [247…

```
eth_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1503 entries, 0 to 1502
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   Date       1503 non-null   datetime64[ns]
 1   Open       1503 non-null   float64       
 2   High       1503 non-null   float64       
 3   Low        1503 non-null   float64       
 4   Close      1503 non-null   float64       
 5   Adj Close  1503 non-null   float64       
 6   Volume     1503 non-null   int64         
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 82.3 KB
```

ARIMAX

In [266…

```
btc_eth = pd.DataFrame(btc_train['2017-11-09':])
btc_eth.head()
```

Out[266…

Date	Open	High	Low	Close	Volume	returns	norm	norm
2017-11-09	7446.830078	7446.830078	7101.520020	7143.580078	3226249984	-4.237574	1683.060042	58.916
2017-11-10	7173.729980	7312.000000	6436.870117	6618.140137	5208249856	-7.355415	1559.263996	102.264
2017-11-11	6618.609863	6873.149902	6204.220215	6357.600098	4908680192	-3.936756	1497.879575	54.733
2017-11-12	6295.450195	6625.049805	5519.009766	5950.069824	8957349888	-6.410128	1401.863584	89.121

Date	Open	High	Low	Close	Volume	returns	norm	norm
2017-11-13	5938.250000	6811.189941	5844.290039	6559.490234	6263249920	10.242240	1545.445812	-142.400

In [268…
```
eth_train.head()
```

Out[268…

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-11-09	308.644989	329.451996	307.056000	320.884003	320.884003	893249984
1	2017-11-10	320.670990	324.717987	294.541992	299.252991	299.252991	885985984
2	2017-11-11	298.585999	319.453003	298.191986	314.681000	314.681000	842300992
3	2017-11-12	314.690002	319.153015	298.513000	307.907990	307.907990	1613479936
4	2017-11-13	307.024994	328.415009	307.024994	316.716003	316.716003	1041889984

In [372…
```
btc_train.tail()
```

Out[372…

Date	Open	High	Low	Close	Volume
2018-10-24	6478.890137	6521.990234	6468.859863	6495.839844	3424670000
2018-10-25	6484.649902	6504.649902	6447.029785	6476.290039	3230550000
2018-10-26	6468.439941	6498.290039	6449.609863	6474.750000	3306050000
2018-10-27	6480.839844	6507.410156	6453.529785	6480.379883	3393250000
2018-10-28	6482.660156	6502.279785	6447.910156	6486.390137	3445190000

In [374…
```
ethdata.head()
```

Out[374…

Date	Open	High	Low	Close	Volume	wn
2017-11-09	308.644989	329.451996	307.056000	320.884003	893249984	972.675449
2017-11-10	320.670990	324.717987	294.541992	299.252991	885985984	481.118096
2017-11-11	298.585999	319.453003	298.191986	314.681000	842300992	76.783736
2017-11-12	314.690002	319.153015	298.513000	307.907990	1613479936	4475.842063
2017-11-13	307.024994	328.415009	307.024994	316.716003	1041889984	542.517858

In [376…
```
#will be done for other Cryptos

model_arimax = ARIMA(btc_train['2017-11-09':].Close, exog = ethdata[:"2018-10-28"].Clos
results_arimax = model_arimax.fit()
results_arimax.summary()
```

Out[376...

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	354
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-2651.841
Date:	Thu, 09 Feb 2023	**AIC**	5311.681
Time:	07:27:56	**BIC**	5327.147
Sample:	11-09-2017	**HQIC**	5317.835
	- 10-28-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
Close	8.1114	0.417	19.457	0.000	7.294	8.929
ar.L1	-0.3027	0.082	-3.676	0.000	-0.464	-0.141
ma.L1	0.5463	0.084	6.535	0.000	0.382	0.710
sigma2	1.984e+05	6336.192	31.307	0.000	1.86e+05	2.11e+05

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	2346.59
Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.02	**Skew:**	0.98
Prob(H) (two-sided):	0.00	**Kurtosis:**	15.48

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

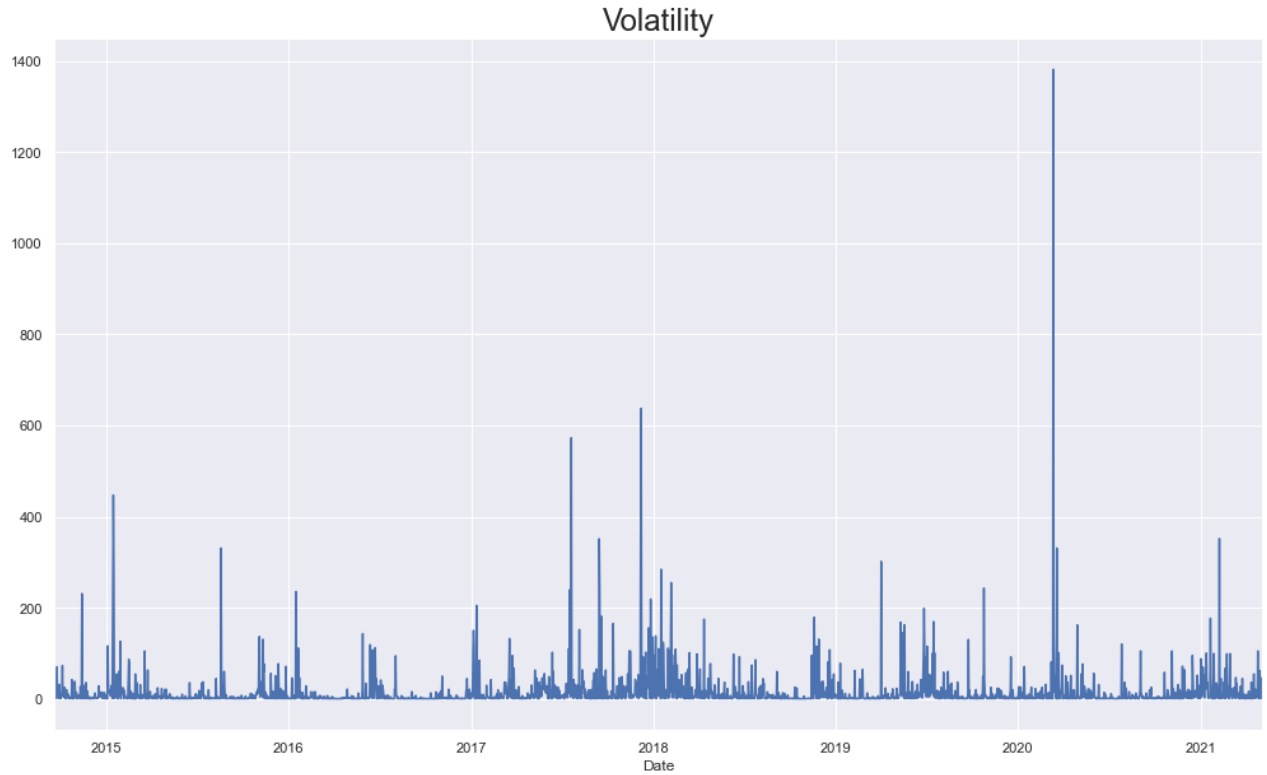
ARCH Model

In [294...
```
from arch import arch_model
```

In [377...
```
btc_train["sq_returns"] = btc_train.returns.mul(btc_train.returns)
btc_test["sq_returns"] = btc_test.returns.mul(btc_test.returns)
```

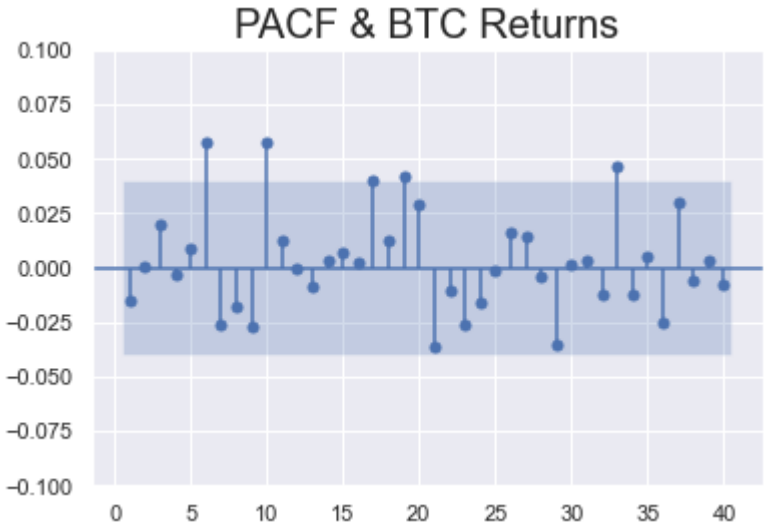
In [285...
```
btc_train.sq_returns.plot(figsize = (17,10))
plt.title('Volatility', size = 24)
```

Out[285...
```
Text(0.5, 1.0, 'Volatility')
```



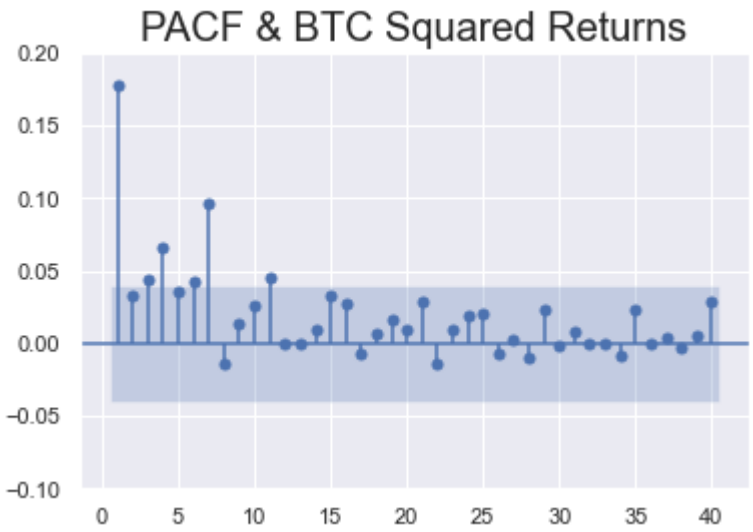
In [288…
```
sgt.plot_pacf(btc_train.returns, lags = 40, zero = False, method = ('ols'))
plt.title("PACF & BTC Returns", size = 20)
plt.ylim(top = 0.1, bottom = -0.1)
```

Out[288…
```
(-0.1, 0.1)
```



In [289…
```
sgt.plot_pacf(btc_train.sq_returns, lags = 40, zero = False, method = ('ols'))
plt.title("PACF & BTC Squared Returns", size = 20)
plt.ylim(top = 0.2, bottom = -0.1)
```

Out[289…
```
(-0.1, 0.2)
```



Simple ARCH MODEL

In [295...
```
model_arch_1 = arch_model(btc_train.returns[1:], mean = "Constant", vol = "ARCH", p = 1
results_model_arch_1 = model_arch_1.fit(update_freq= 5)
results_model_arch_1.summary()
```

```
Iteration:      5,   Func. Count:     27,   Neg. LLF: 6648.060739180086
Optimization terminated successfully    (Exit mode 0)
            Current function value: 6648.060739180086
            Iterations: 6
            Function evaluations: 30
            Gradient evaluations: 6
```

Out[295...

Constant Mean - ARCH Model Results

Dep. Variable:	returns	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	ARCH	**Log-Likelihood:**	-6648.06
Distribution:	Normal	**AIC:**	13302.1
Method:	Maximum Likelihood	**BIC:**	13319.5
		No. Observations:	2420
Date:	Thu, Feb 02 2023	**Df Residuals:**	2419
Time:	15:41:25	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	0.2964	7.489e-02	3.958	7.568e-05	[0.150, 0.443]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	12.4160	1.126	11.026	2.871e-28	[10.209, 14.623]
alpha[1]	0.1698	4.147e-02	4.094	4.236e-05	[8.852e-02, 0.251]

Covariance estimator: robust

Change mean from constant to "ZERO" or "AR" and see the results and interpret. If you you use mean AR then mention lags

In [300…
```
model_arch_2 = arch_model(btc_train.returns[1:], mean = "Constant", vol = "ARCH", p = 2
results_model_arch_2 = model_arch_2.fit(update_freq= 5)
results_model_arch_2.summary()
```

```
Iteration:      5,   Func. Count:     34,   Neg. LLF: 6627.907409514692
Optimization terminated successfully    (Exit mode 0)
            Current function value: 6627.6901264643675
            Iterations: 7
            Function evaluations: 44
            Gradient evaluations: 7
```

Out[300…

Constant Mean - ARCH Model Results

Dep. Variable:	returns	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	ARCH	**Log-Likelihood:**	-6627.69
Distribution:	Normal	**AIC:**	13263.4
Method:	Maximum Likelihood	**BIC:**	13286.5
		No. Observations:	2420
Date:	Thu, Feb 02 2023	**Df Residuals:**	2419
Time:	15:50:06	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	0.3064	6.836e-02	4.482	7.393e-06	[0.172, 0.440]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	10.9034	1.407	7.749	9.296e-15	[8.145, 13.661]
alpha[1]	0.1542	4.116e-02	3.747	1.786e-04	[7.357e-02, 0.235]
alpha[2]	0.1355	5.558e-02	2.439	1.475e-02	[2.660e-02, 0.244]

Covariance estimator: robust

Garch Model

In [301…
```
model_arch_1 = arch_model(btc_train.returns[1:], mean = "Constant", vol = "GARCH", p =
results_model_arch_1 = model_arch_1.fit(update_freq= 5)
results_model_arch_1.summary()
```

```
Iteration:      5,   Func. Count:     39,   Neg. LLF: 6496.1780209668295
Iteration:     10,   Func. Count:     64,   Neg. LLF: 6486.725847041355
Optimization terminated successfully    (Exit mode 0)
```

```
Current function value: 6486.725847041356
Iterations: 10
Function evaluations: 64
Gradient evaluations: 10
```

Out[301…

Constant Mean - GARCH Model Results

Dep. Variable:	returns	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	GARCH	**Log-Likelihood:**	-6486.73
Distribution:	Normal	**AIC:**	12981.5
Method:	Maximum Likelihood	**BIC:**	13004.6
		No. Observations:	2420
Date:	Thu, Feb 02 2023	**Df Residuals:**	2419
Time:	16:11:03	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	0.2392	6.322e-02	3.783	1.547e-04	[0.115, 0.363]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	0.6753	0.258	2.616	8.891e-03	[0.169, 1.181]
alpha[1]	0.1294	3.184e-02	4.065	4.794e-05	[6.703e-02, 0.192]
beta[1]	0.8373	2.924e-02	28.633	2.575e-180	[0.780, 0.895]

Covariance estimator: robust

try higher lags if necessary

Forecasting

In [310…

```
import yfinance
```

In [311…

```
btc_train.tail()
```

Out[311…

Date	Open	High	Low	Close	Volume	returns	norm
2021-04-30	53568.664063	57900.718750	53129.601563	57750.175781	52395931985	7.833177	13606.204766
2021-05-01	57714.664063	58448.339844	57052.273438	57828.050781	42836427360	0.134848	13624.552471
2021-05-02	57825.863281	57902.593750	56141.906250	56631.078125	38177405335	-2.069882	13342.540255

Date	Open	High	Low	Close	Volume	returns	norm
2021-05-03	56620.273438	58973.308594	56590.871094	57200.292969	51713139031	1.005128	13476.649868
2021-05-04	57214.179688	57214.179688	53191.425781	53333.539063	68564706967	-6.760025	12565.625015

In [450…
```
start_date = "2021-05-05"
end_date = "2022-12-31"
```

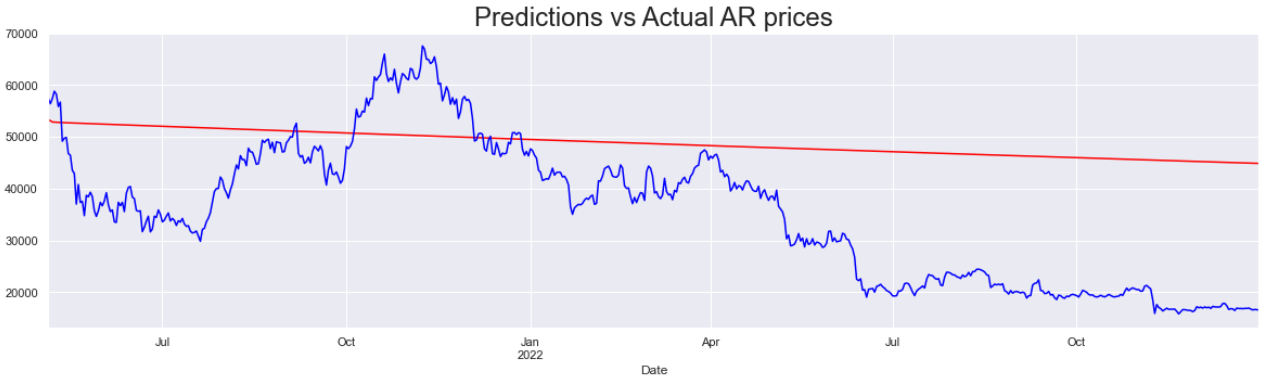
AR Forecasting

In [331…
```
pred_ar = results_ar_model_1.predict(start = start_date, end = end_date)
```

In [333…
```
pred_ar
```

Out[333…
```
2021-05-05    53211.110243
2021-05-06    53133.375864
2021-05-07    52863.160361
2021-05-08    52836.926378
2021-05-09    52811.511753
                  ...
2022-12-27    44913.110490
2022-12-28    44901.128701
2022-12-29    44889.150717
2022-12-30    44877.176535
2022-12-31    44865.206154
Freq: D, Name: predicted_mean, Length: 606, dtype: float64
```

In [334…
```
pred_ar[start_date:end_date].plot(figsize = (20,5), color = "red")
btc_test.Close[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual AR prices", size = 24)
plt.show()
```

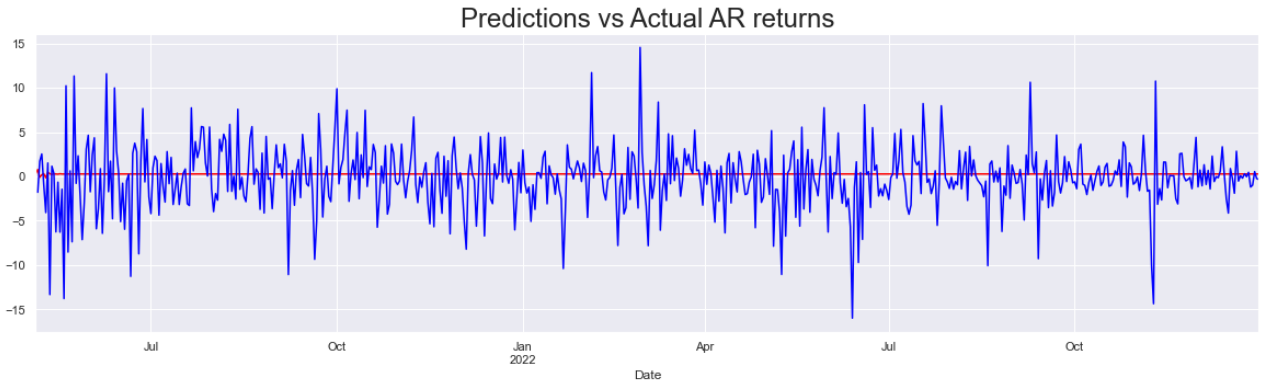


Using Returns for Forecasting

In [328…
```
pred_ret_ar = results_ar_model_ret_2.predict(start = start_date, end = end_date)
```

In [329…
```
pred_ret_ar[start_date:end_date].plot(figsize = (20,5), color = "red")
btc_test.returns[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual AR returns", size = 24)
```

Out[329…
```
Text(0.5, 1.0, 'Predictions vs Actual AR returns')
```



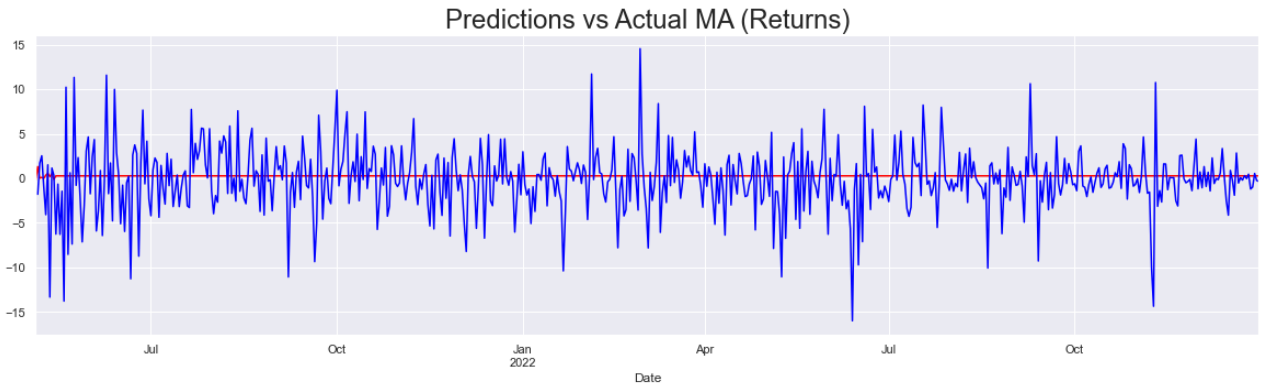
In [ ]:
```
```

MA Forecasting

In [335…
```
pred_ma_ret = results_ret_ma_2.predict(start = start_date, end = end_date)
```

In [349…
```
pred_ma_ret[start_date:end_date].plot(figsize = (20,5), color = "red")
btc_test.returns[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual MA (Returns)", size = 24)
```

Out[349…
```
Text(0.5, 1.0, 'Predictions vs Actual MA (Returns)')
```

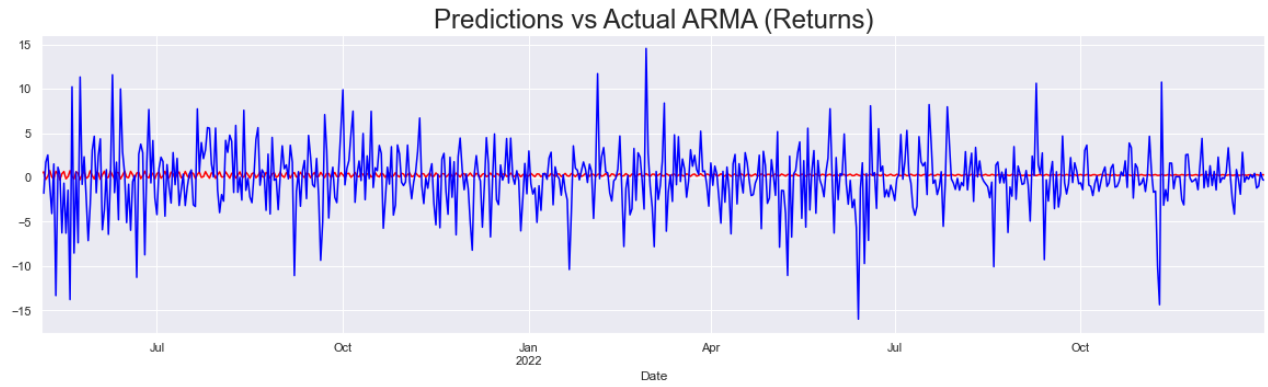


ARMA Forecasting

In [352…
```
pred_arma_ret = results_ret_arma_1.predict(start = start_date, end = end_date)
```

In [354…
```
pred_arma_ret[start_date:end_date].plot(figsize = (20,5), color = "red")
btc_test.returns[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual ARMA (Returns)", size = 24)
```

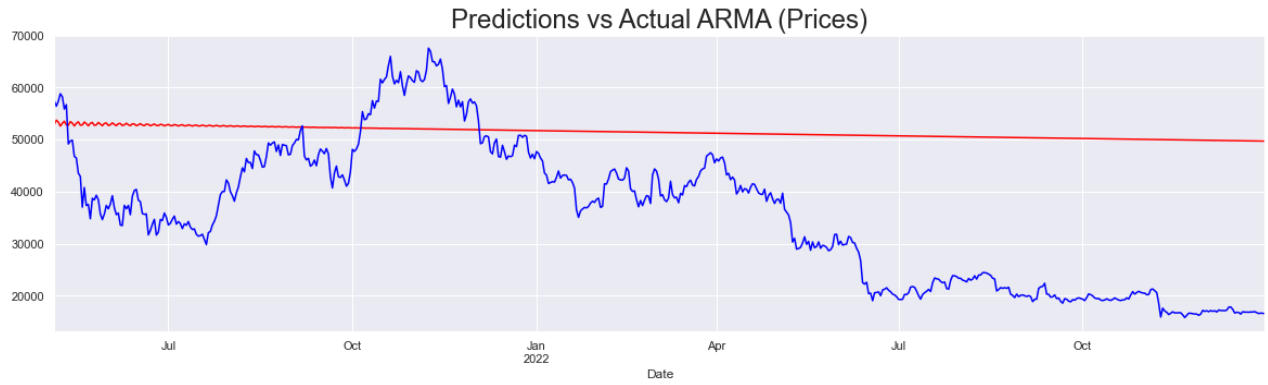
Out[354…
```
Text(0.5, 1.0, 'Predictions vs Actual ARMA (Returns)')
```



In [357…
```
pred_arma_prices = results_arma_2.predict(start = start_date, end = end_date)
```

In [379…
```
pred_arma_prices[start_date:end_date].plot(figsize = (20,5), color = "red")
btc_test.Close[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual ARMA (Prices)", size = 24)
```

Out[379…
```
Text(0.5, 1.0, 'Predictions vs Actual ARMA (Prices)')
```

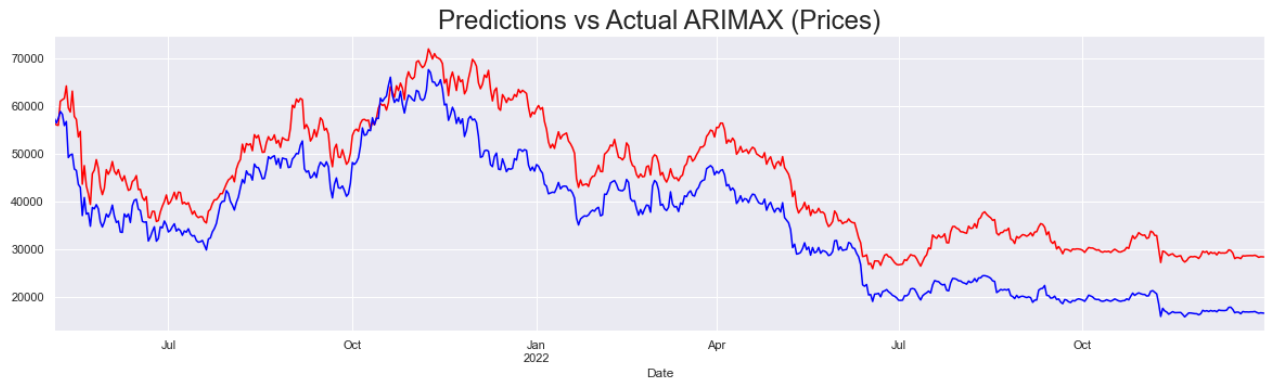


ARMAX Forecasting

In [440…
```
pred_armax = results_arimax.predict(start = start_date, end = end_date,
                                    exog = ethdata.Close[start_date:end_date])

pred_armax[start_date:end_date].plot(figsize = (20,5), color = "red")
btc_test.Close[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual ARIMAX (Prices)", size = 24)
```

Out[440…
```
Text(0.5, 1.0, 'Predictions vs Actual ARIMAX (Prices)')
```



In [475…
```
pred_armax = results_arimax.predict(start = start_date, end = end_date
                                    ,exog = ethdata.Close[start_date:end_date])
```

In [476…
```
pred_armax.tail()
```

Out[476…
```
2022-12-27    28526.433273
2022-12-28    28251.553771
2022-12-29    28391.482288
2022-12-30    28363.006039
2022-12-31    28333.336537
Freq: D, Name: predicted_mean, dtype: float64
```

Forecasting Volatility

In [516…
```
mod_garch = arch_model(btc_train.returns[1:], vol = 'GARCH', p = 1, q = 1, mean = 'cons
res_garch = mod_garch.fit(last_obs=start_date, update_freq = 10)
```

```
Iteration:     10,   Func. Count:     64,   Neg. LLF: 6486.725847041355
Optimization terminated successfully    (Exit mode 0)
            Current function value: 6486.725847041356
            Iterations: 10
            Function evaluations: 64
            Gradient evaluations: 10
```

In [518…
```
pred_garch = res_garch.forecast()
```

In [523…
```
pred_garch.mean
```

Out[523…

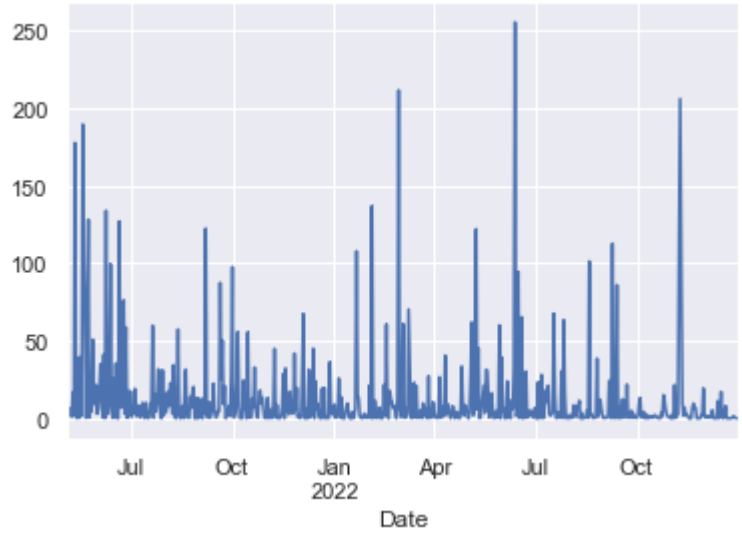
Date	h.1
2014-09-19	NaN
2014-09-20	NaN
2014-09-21	NaN
2014-09-22	NaN
2014-09-23	NaN
…	…
2021-04-30	NaN
2021-05-01	NaN
2021-05-02	NaN
2021-05-03	NaN
2021-05-04	0.239181

2420 rows × 1 columns

In [487…
```
btc_test.sq_returns.plot()
pred_garch
```

Out[487…

```
<AxesSubplot:xlabel='Date'>
```



In [474…

```
start_date = "2021-05-05"
end_date = "2022-12-31"
```

Univariate Time Series Analysis of ETH

In [3]:

```
ethdata = pd.read_csv("ETH-USD.csv")
ethdata.Date = pd.to_datetime(ethdata.Date, yearfirst= True)
```

In [4]:

```
# Dropping columns adj close as it is the same as close price
ethdata.drop(['Adj Close'], axis=1, inplace= True)
```

In [5]:

```
ethdata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1879 entries, 0 to 1878
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype         
---  ------  --------------  -----         
 0   Date    1879 non-null   datetime64[ns]
 1   Open    1879 non-null   float64       
 2   High    1879 non-null   float64       
 3   Low     1879 non-null   float64       
 4   Close   1879 non-null   float64       
 5   Volume  1879 non-null   int64         
dtypes: datetime64[ns](1), float64(4), int64(1)
memory usage: 88.2 KB
```

In [10]:

```
ethdata.head()
```

Out[10]:

Date	Open	High	Low	Close	Volume
2017-11-09	308.644989	329.451996	307.056000	320.884003	893249984
2017-11-10	320.670990	324.717987	294.541992	299.252991	885985984
2017-11-11	298.585999	319.453003	298.191986	314.681000	842300992

Date	Open	High	Low	Close	Volume
2017-11-12	314.690002	319.153015	298.513000	307.907990	1613479936
2017-11-13	307.024994	328.415009	307.024994	316.716003	1041889984

In [11]:
```
# Positions and Variance of the observations
ethdata.describe()
```

Out[11]:

	Open	High	Low	Close	Volume
count	1879.000000	1879.000000	1879.000000	1879.000000	1.879000e+03
mean	1129.660247	1165.978366	1088.324066	1129.901821	1.289976e+10
std	1188.560481	1224.871061	1146.506745	1187.932777	1.078927e+10
min	84.279694	85.342743	82.829887	84.308296	6.217330e+08
25%	211.185448	216.114403	206.339790	211.278061	4.478524e+09
50%	502.880005	523.544983	488.278992	502.894012	1.063103e+10
75%	1777.508423	1817.918396	1706.658325	1775.859924	1.819402e+10
max	4810.071289	4891.704590	4718.039063	4812.087402	8.448291e+10

In [12]:
```
ethdata= ethdata.asfreq('d')
```

In [13]:
```
ethdata.head()
```

Out[13]:

Date	Open	High	Low	Close	Volume
2017-11-09	308.644989	329.451996	307.056000	320.884003	893249984
2017-11-10	320.670990	324.717987	294.541992	299.252991	885985984
2017-11-11	298.585999	319.453003	298.191986	314.681000	842300992
2017-11-12	314.690002	319.153015	298.513000	307.907990	1613479936
2017-11-13	307.024994	328.415009	307.024994	316.716003	1041889984

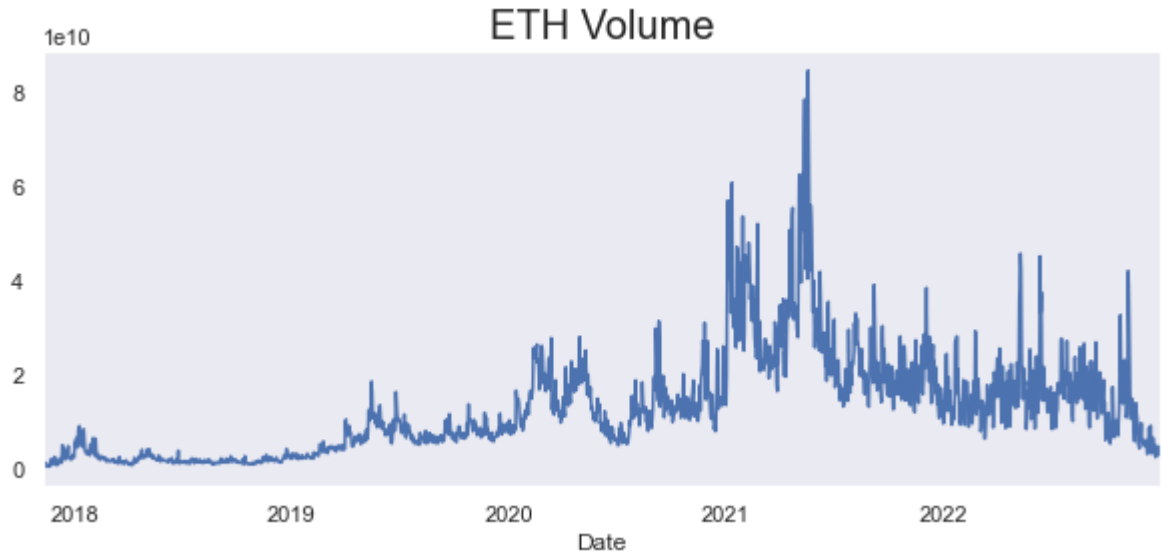
In [14]:
```
# Univariate Analysis

ethdata.Close.plot(title = "BTC Price", figsize = (15,5), grid  = False)
plt.title("ETH Prices", size = 24)
```

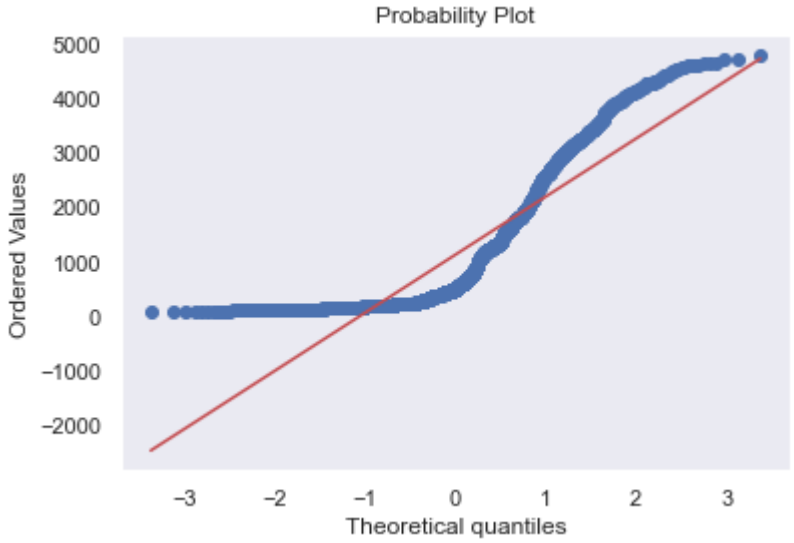
Out[14]:
```
Text(0.5, 1.0, 'ETH Prices')
```

In [15]:
```
# Univariate Analysis
ethdata.Volume.plot(title = "ETH Volume", figsize = (10,4), grid  = False)
plt.title("ETH Volume", size = 20)
```

Out[15]:
```
Text(0.5, 1.0, 'ETH Volume')
```



In [16]:
```
# In the next step we are building a QQ(Quantile-Quantile plot) in order to understand
scipy.stats.probplot(ethdata.Close, plot = pylab)
pylab.grid(visible = None)
pylab.show()
```



In [17]:
```
# Finding the 80 percent of the data to set as train data.
size = int(len(ethdata)*0.8)
```

In [18]:
```
eth_train = ethdata.iloc[:size]
```

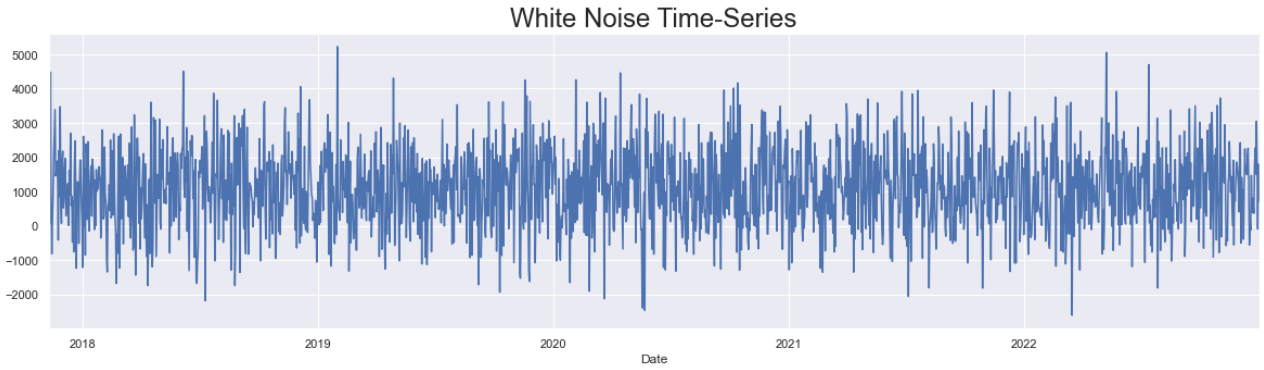
In [19]:
```
eth_test = ethdata.iloc[size:]
```

In [20]:
```
wn = np.random.normal(loc = ethdata.Close.mean(), scale = ethdata.Close.std(), size = l
```

In [21]:
```
ethdata['wn'] = wn
```

In [22]:
```
# Creating White-Noise Time Series
ethdata.wn.plot(figsize = (20,5))
plt.title("White Noise Time-Series", size = 24)
```

Out[22]:
```
Text(0.5, 1.0, 'White Noise Time-Series')
```



In [23]:
```
print(wn.mean())
```

```
1103.4580659925439
```

Random walk ?

Stationarity

In [25]:
```
#Dicky-Fuller test to test stationarity
##as our T statistic is greated than all of the confidence levels, we dont have enough
#some autocorrelations go back 28 times, 2900 is the number of observations that test w
sts.adfuller(ethdata.Close)
```

Out[25]:
```
(-1.4089708365742828,
 0.5779679105330212,
 17,
 1861,
 {'1%': -3.4338687226315336,
  '5%': -2.863094318475046,
  '10%': -2.5675974634086765},
 21446.440104463112)
```

In [26]:
```
#Dicky-Fuller test to test White Noise
##as our T statistic is greated than all of the confidence levels, we dont have enough
#some autocorrelations go back 28 times, 2900 is the number of observations that test w
sts.adfuller(ethdata.wn)
```

Out[26]:
```
(-43.36758246285912,
 0.0,
 0,
 1878,
 {'1%': -3.4338368268127306,
  '5%': -2.863080237422199,
  '10%': -2.567589965782827},
 31467.46942173355)
```

add random walk test as well

In [ ]:
```
#Dicky-Fuller test to test stationarity for Volume
##as our T statistic is greated than all of the confidence levels, we dont have enough
#some autocorrelations go back 28 times, 2900 is the number of observations that test w
sts.adfuller(btcdata.Volume)
```

Seasonality

In [ ]:
```
# Seasonality, from seasonal graph you can see that there is no clear seasonal cycle.
# Resid shows, that in 2018, mid2019 2020, 2021,2022 there were a lot of differences be
s_dec_additive = seasonal_decompose(btcdata.Close, model = "additive")
s_dec_additive.plot()
plt.show()
```

In [ ]:
```
#Very similar results in multiplicative test as well which shows that there is no seaso
s_dec_multiplicitive = seasonal_decompose(btcdata.Close, model = "Multiplicative")
s_dec_multiplicitive.plot()
plt.show()
```

ACF

In [ ]:
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(btcdata.Close, lags = 40, zero = False)
plt.title("ACF & BTC", size = 24)
plt.ylim(bottom = -0.35, top = 1.1)
```

In [ ]:
```
sgt.plot_acf(btcdata.wn, lags = 40, zero = False)
plt.title("ACF & WN", size = 20)
plt.ylim(top = 0.05, bottom = -0.05)
```

PACF

In [ ]:
```
sgt.plot_pacf(btcdata.Close, lags = 60, zero = False, method = ('ols'))
plt.title("PACF & BTC", size = 20)
plt.ylim(top = 1.10, bottom = -0.25)
```

In [ ]:
```
sgt.plot_pacf(btcdata.wn, lags = 40, zero = False, method = ('ols'))
plt.title("PACF & WN", size = 20)
plt.ylim(top = 0.05, bottom = -0.05)
```

LLR test function for models

In [ ]:
```
def LLR_test(mod_1, mod_2, DF = 1):
    L1 = mod_1.fit().llf
    L2 = mod_2.fit().llf
    LR = (2*(L2-L1))
    p = chi2.sf(LR, DF).round(3)
    return p
```

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")
```

AR Model

In [ ]:
```
from statsmodels.tsa.arima_process import ArmaProcess
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

In [ ]:
```
from statsmodels.tsa.ar_model import AutoReg
```

In [100…
```
#ORDER first - AR, Second - Differences, Third - MA
ar_model = ARIMA(eth_train.Close[1:], order = (2,0,0))
results_ar_model = ar_model.fit()
results_ar_model.summary()
```

Out[100…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	1500
Model:	ARIMA(2, 0, 0)	**Log Likelihood**	-8619.516
Date:	Sat, 04 Feb 2023	**AIC**	17247.033
Time:	20:26:31	**BIC**	17268.286
Sample:	11-12-2017	**HQIC**	17254.950
	- 12-20-2021		
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**

const	902.3916	4711.752	0.192	0.848	-8332.473	1.01e+04
ar.L1	0.8998	0.013	71.536	0.000	0.875	0.925
ar.L2	0.0992	0.013	7.886	0.000	0.075	0.124
sigma2	5713.8926	61.503	92.905	0.000	5593.350	5834.436

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	35217.81
Prob(Q):	0.99	**Prob(JB):**	0.00
Heteroskedasticity (H):	13.85	**Skew:**	-0.96
Prob(H) (two-sided):	0.00	**Kurtosis:**	26.66

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

In [101…
```
#ORDER first - AR, Second - Differences, Third - MA
ar_model_1 = ARIMA(eth_train.Close[1:], order = (3,0,0))
results_ar_model_1 = ar_model_1.fit()
results_ar_model_1.summary()
```

Out[101…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	1500
Model:	ARIMA(3, 0, 0)	**Log Likelihood**	-8619.500
Date:	Sat, 04 Feb 2023	**AIC**	17248.999
Time:	20:26:39	**BIC**	17275.565
Sample:	11-12-2017	**HQIC**	17258.896
	- 12-20-2021		
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	902.3877	4677.272	0.193	0.847	-8264.897	1.01e+04
ar.L1	0.9003	0.013	70.624	0.000	0.875	0.925
ar.L2	0.1035	0.015	6.832	0.000	0.074	0.133
ar.L3	-0.0047	0.010	-0.452	0.651	-0.025	0.016
sigma2	5713.8087	63.656	89.760	0.000	5589.045	5838.573

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	34973.33
Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	13.85	**Skew:**	-0.95
Prob(H) (two-sided):	0.00	**Kurtosis:**	26.58

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

In [ ]:
```
LLR_test(ar_model, ar_model_1)
```

As our data is coming from Non-Stationary data we wulll transform our data using returns. Returns are percentage change between the values for 2 consecutive periods

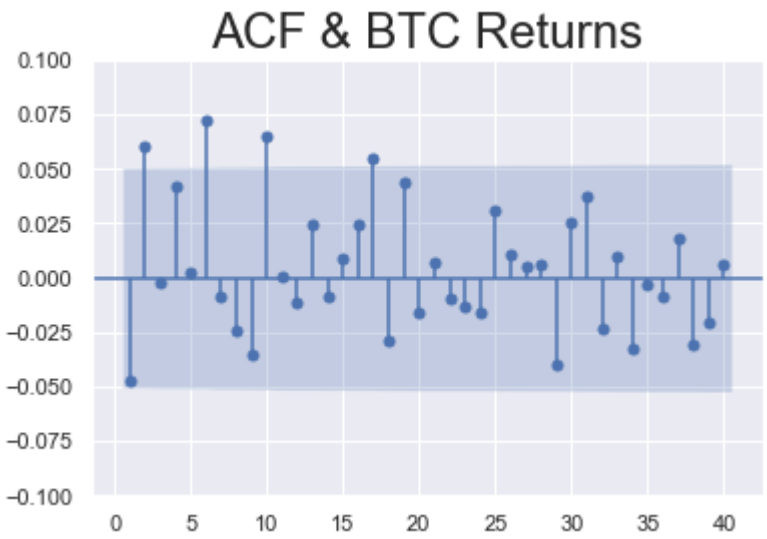
In [33]:
```
eth_train['returns'] = eth_train.Close.pct_change(1).mul(100)
eth_test['returns'] = eth_test.Close.pct_change(1).mul(100)
eth_train = eth_train.iloc[1:]
```

In [34]:
```
sts.adfuller(eth_train.returns)
```

Out[34]:
```
(-11.416317801866356,
 7.048091371371786e-21,
 9,
 1491,
 {'1%': -3.434743423170358,
  '5%': -2.8634804142964025,
  '10%': -2.567803054306163},
 9049.330073140953)
```

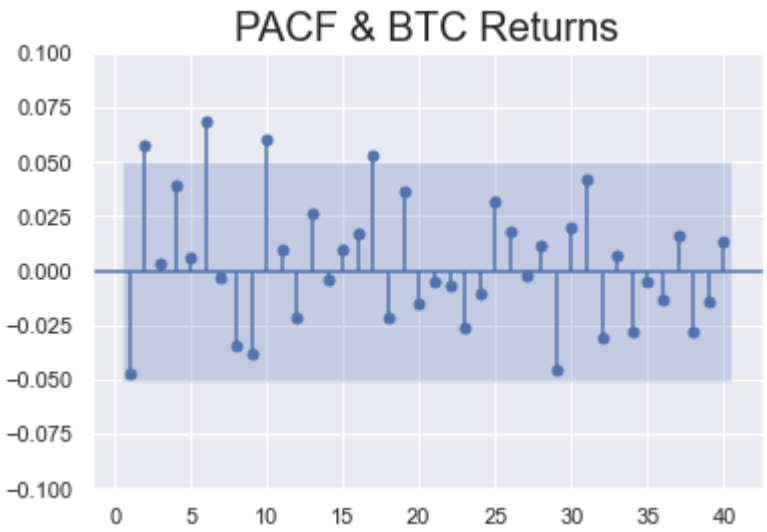
In [35]:
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.returns, lags = 40, zero = False)
plt.title("ACF & BTC Returns", size = 24)
plt.ylim(top = 0.1, bottom = -0.1)
```

Out[35]:
```
(-0.1, 0.1)
```



In [36]:
```
sgt.plot_pacf(eth_train.returns, lags = 40, zero = False, method = ('ols'))
plt.title("PACF & BTC Returns", size = 20)
plt.ylim(top = 0.1, bottom = -0.1)
```

Out[36]:
```
(-0.1, 0.1)
```



In [377…
```
os.getcwd()
```

Out[377…
```
'C:\\Users\\tural\\Downloads\\Crypto_Datasets'
```

In [57]:
```
#ORDER first - AR, Second - Differences, Third - MA
ar_model_ret_1 = ARIMA(eth_train.returns[1:], order = (4,0,0))
results_ar_model_ret_1 = ar_model_ret_1.fit()
results_ar_model_ret_1.summary()
```

Out[57]:

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	1500
Model:	ARIMA(4, 0, 0)	**Log Likelihood**	-4593.996
Date:	Sat, 04 Feb 2023	**AIC**	9199.991
Time:	20:09:57	**BIC**	9231.870
Sample:	11-12-2017	**HQIC**	9211.867
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.3060	0.143	2.133	0.033	0.025	0.587
ar.L1	-0.0444	0.020	-2.278	0.023	-0.083	-0.006
ar.L2	0.0555	0.024	2.295	0.022	0.008	0.103
ar.L3	0.0039	0.023	0.170	0.865	-0.041	0.049
ar.L4	0.0389	0.020	1.912	0.056	-0.001	0.079
sigma2	26.7740	0.552	48.481	0.000	25.692	27.856

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	1879.90
Prob(Q):	0.99	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.90	**Skew:**	-0.25

Prob(H) (two-sided):	0.23	**Kurtosis:**	8.46

Warnings:

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

In [58]:
```
#ORDER first - AR, Second - Differences, Third - MA
ar_model_ret_2 = ARIMA(eth_train.returns[1:], order = (8,0,0))
results_ar_model_ret_2 = ar_model_ret_2.fit()
results_ar_model_ret_2.summary()
```

Out[58]:

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	1500
Model:	ARIMA(8, 0, 0)	**Log Likelihood**	-4589.558
Date:	Sat, 04 Feb 2023	**AIC**	9199.117
Time:	20:10:01	**BIC**	9252.249
Sample:	11-12-2017	**HQIC**	9218.911
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.3061	0.149	2.051	0.040	0.014	0.599
ar.L1	-0.0450	0.020	-2.246	0.025	-0.084	-0.006
ar.L2	0.0552	0.025	2.191	0.028	0.006	0.105
ar.L3	0.0038	0.024	0.162	0.872	-0.042	0.050
ar.L4	0.0365	0.021	1.780	0.075	-0.004	0.077
ar.L5	0.0094	0.022	0.428	0.669	-0.034	0.052
ar.L6	0.0699	0.024	2.909	0.004	0.023	0.117
ar.L7	-0.0050	0.020	-0.250	0.803	-0.044	0.034
ar.L8	-0.0349	0.025	-1.420	0.156	-0.083	0.013
sigma2	26.6151	0.555	47.984	0.000	25.528	27.702

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	1975.82
Prob(Q):	0.96	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.90	**Skew:**	-0.26
Prob(H) (two-sided):	0.23	**Kurtosis:**	8.60

Warnings:

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

In [59]:
```
LLR_test(ar_model_ret_1, ar_model_ret_2)
```

Out[59]:
```
0.003
```

Normalizing Values

In [62]:
```
benchmark = eth_train.Close.iloc[0]
```

In [63]:
```
eth_train['norm'] = eth_train.Close.div(benchmark).mul(100)
eth_test['norm'] = eth_test.Close.div(benchmark).mul(100)
```

In [65]:
```
sts.adfuller(eth_train.norm)
```

Out[65]:
```
(0.6069429080168565,
 0.9877855155545913,
 17,
 1483,
 {'1%': -3.4347671645756304,
  '5%': -2.86349089226533,
  '10%': -2.5678086339403325},
 13543.131431078897)
```

Normalizing Returns

In [67]:
```
benchmark_ret = eth_train.returns.iloc[0]
eth_train['norm_ret'] = eth_train.returns.div(benchmark_ret).mul(100)
eth_test['norm_ret'] = eth_test.returns.div(benchmark_ret).mul(100)
```

In [68]:
```
sts.adfuller(eth_train.norm_ret)
```

Out[68]:
```
(-11.416317801866343,
 7.048091371372087e-21,
 9,
 1491,
 {'1%': -3.434743423170358,
  '5%': -2.8634804142964025,
  '10%': -2.567803054306163},
 17802.319193813113)
```

In [96]:
```
### ORDER first - AR, Second - Differences, Third - MA
ar_model_norm_ret_1 = ARIMA(eth_train.norm_ret, order = (2,0,0))
results_ar_model_norm_ret_1 = ar_model_norm_ret_1.fit()
results_ar_model_norm_ret_1.summary()
```

Out[96]:

SARIMAX Results

Dep. Variable:	norm_ret	**No. Observations:**	1501
Model:	ARIMA(2, 0, 0)	**Log Likelihood**	-9048.715
Date:	Sat, 04 Feb 2023	**AIC**	18105.430
Time:	20:21:27	**BIC**	18126.686
Sample:	11-11-2017	**HQIC**	18113.348
	- 12-20-2021		

Covariance Type: opg

	coef	std err	z	P>|z|	[0.025	0.975]
const	5.9962	2.660	2.254	0.024	0.782	11.211
ar.L1	-0.0445	0.019	-2.296	0.022	-0.082	-0.007
ar.L2	0.0579	0.024	2.404	0.016	0.011	0.105
sigma2	1.009e+04	190.612	52.954	0.000	9720.050	1.05e+04

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	2025.50
Prob(Q):	0.99	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.90	**Skew:**	-0.26
Prob(H) (two-sided):	0.24	**Kurtosis:**	8.67

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

In [94]:
```
#ORDER first - AR, Second - Differences, Third - MA
ar_model_norm_ret_2 = ARIMA(eth_train.norm_ret, order = (10,0,0))
results_ar_model_norm_ret_2 = ar_model_norm_ret_2.fit()
results_ar_model_norm_ret_2.summary()
```

Out[94]:

SARIMAX Results

Dep. Variable:	norm_ret	**No. Observations:**	1501
Model:	ARIMA(10, 0, 0)	**Log Likelihood**	-9039.367
Date:	Sat, 04 Feb 2023	**AIC**	18102.734
Time:	20:21:15	**BIC**	18166.501
Sample:	11-11-2017	**HQIC**	18126.489
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	5.9962	2.993	2.004	0.045	0.131	11.862
ar.L1	-0.0443	0.020	-2.216	0.027	-0.084	-0.005
ar.L2	0.0575	0.025	2.274	0.023	0.008	0.107
ar.L3	0.0074	0.024	0.309	0.757	-0.040	0.054
ar.L4	0.0325	0.021	1.580	0.114	-0.008	0.073
ar.L5	0.0099	0.023	0.437	0.662	-0.034	0.054
ar.L6	0.0679	0.024	2.821	0.005	0.021	0.115
ar.L7	-0.0026	0.020	-0.132	0.895	-0.042	0.036

ar.L8	-0.0399	0.025	-1.613	0.107	-0.088	0.009
ar.L9	-0.0351	0.026	-1.367	0.172	-0.085	0.015
ar.L10	0.0601	0.024	2.545	0.011	0.014	0.106
sigma2	1e+04	208.680	47.926	0.000	9592.210	1.04e+04

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	2088.42
Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.88	**Skew:**	-0.28
Prob(H) (two-sided):	0.14	**Kurtosis:**	8.75

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

In [97]:
```
LLR_test(ar_model_norm_ret_1, ar_model_norm_ret_2)
```

Out[97]:
```
0.0
```

AR model for normalized returns

AR RESIDUALS Price

In [102…
```
eth_train['res_price'] = results_ar_model_1.resid
eth_test['res_price'] = results_ar_model_1.resid
```

In [103…
```
eth_train.res_price.mean()
```

Out[103…
```
2.245538426615259
```

In [104…
```
eth_train.res_price.var()
```

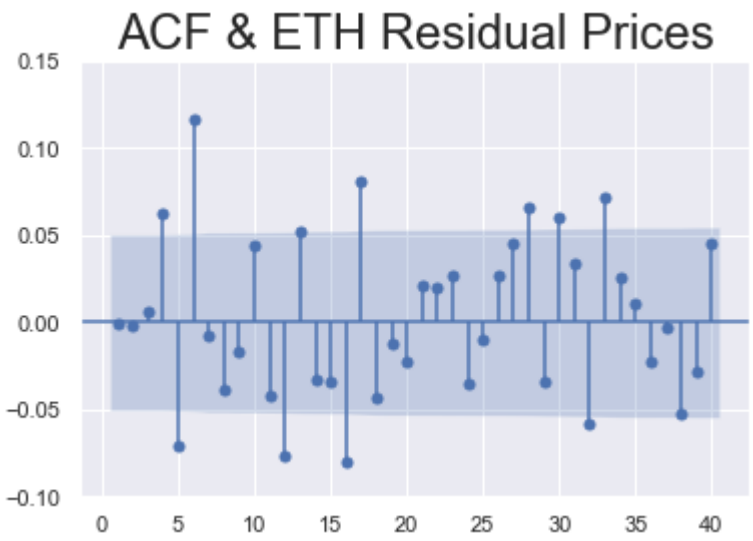
Out[104…
```
5947.85755665728
```

In [105…
```
sts.adfuller(eth_train.res_price[1:])
```

Out[105…
```
(-9.90433891325914,
 3.293878938828608e-17,
 16,
 1483,
 {'1%': -3.4347671645756304,
  '5%': -2.86349089226533,
  '10%': -2.5678086339403325},
 16916.052118069485)
```

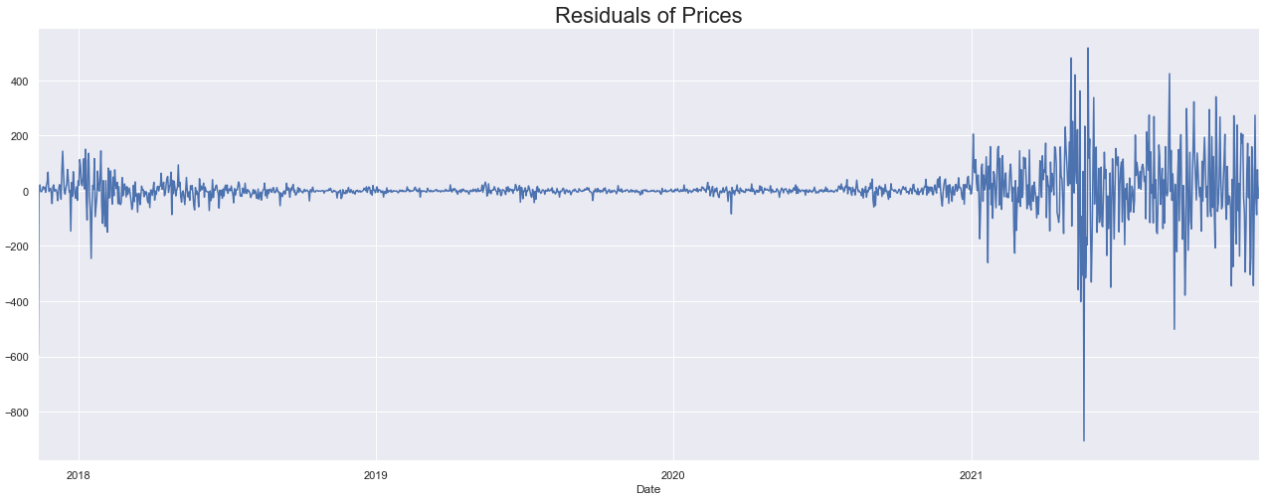
In [106…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.res_price[1:], lags = 40, zero = False)
plt.title("ACF & ETH Residual Prices", size = 24)
plt.ylim(top = 0.15, bottom = -0.1)
```

Out[106…
```
(-0.1, 0.15)
```



In [107…
```
eth_train.res_price[1:].plot(figsize = (22,8))
plt.title("Residuals of Prices", size=22)
```

Out[107…
```
Text(0.5, 1.0, 'Residuals of Prices')
```



In [108…
```
eth_train['res_price_ret'] = results_ar_model_ret_2.resid
eth_test['res_price_ret'] = results_ar_model_ret_2.resid
```

In [109…
```
eth_train.res_price_ret.mean()
```

Out[109…
```
-1.857860593810645e-05
```

In [110…
```
eth_train.res_price_ret.var()
```

Out[110…
```
26.63373497151356
```

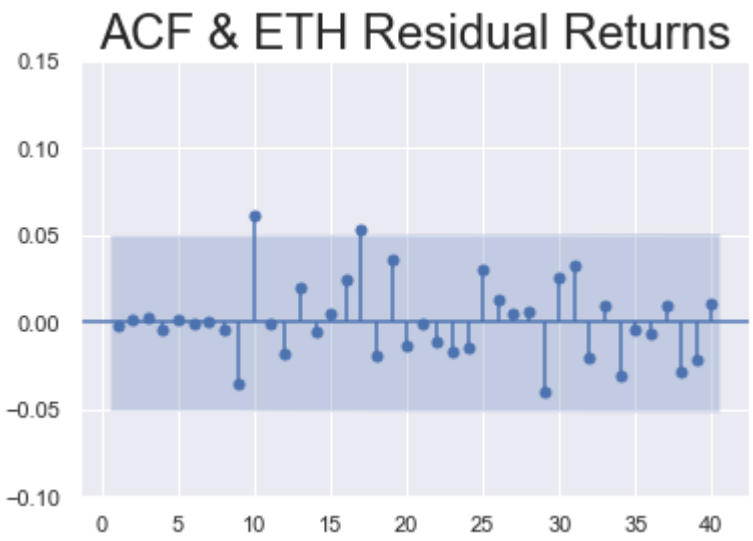
In [114…
```
sts.adfuller(eth_train.res_price_ret[1:])
```

Out[114…
```
(-38.746625407896836,
 0.0,
 0,
 1499,
 {'1%': -3.4347199356122493,
  '5%': -2.86347004827819,
```

```
 '10%': -2.567797534300163},
9031.09628608121)
```

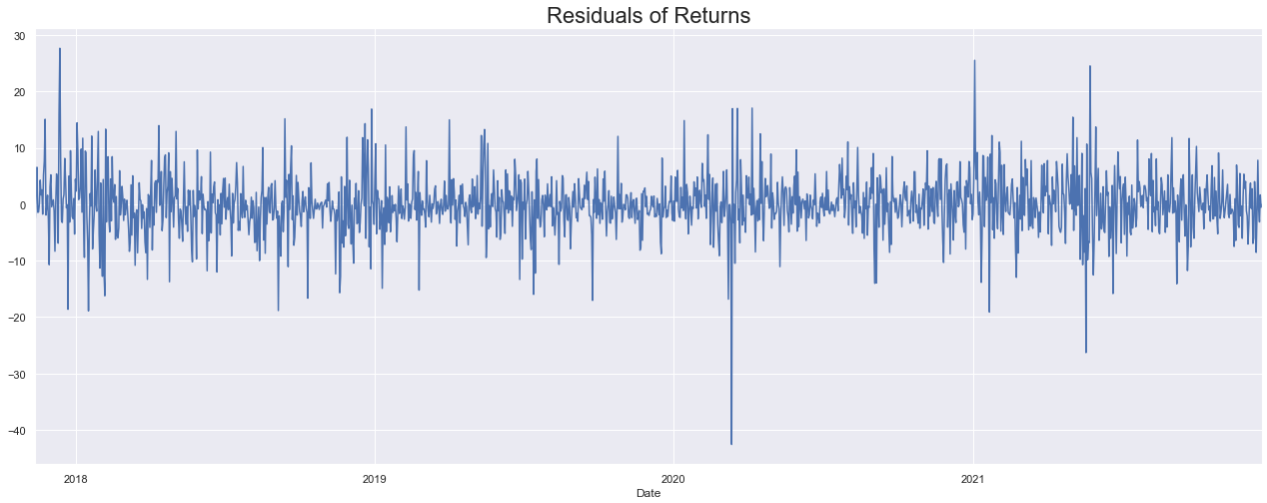
In [113…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.res_price_ret[1:], lags = 40, zero = False)
plt.title("ACF & ETH Residual Returns", size = 24)
plt.ylim(top = 0.15, bottom = -0.1)
```

Out[113…
```
(-0.1, 0.15)
```



In [115…
```
eth_train.res_price_ret[1:].plot(figsize = (22,8))
plt.title("Residuals of Returns", size=22)
```

Out[115…
```
Text(0.5, 1.0, 'Residuals of Returns')
```



In [ ]:
```
```

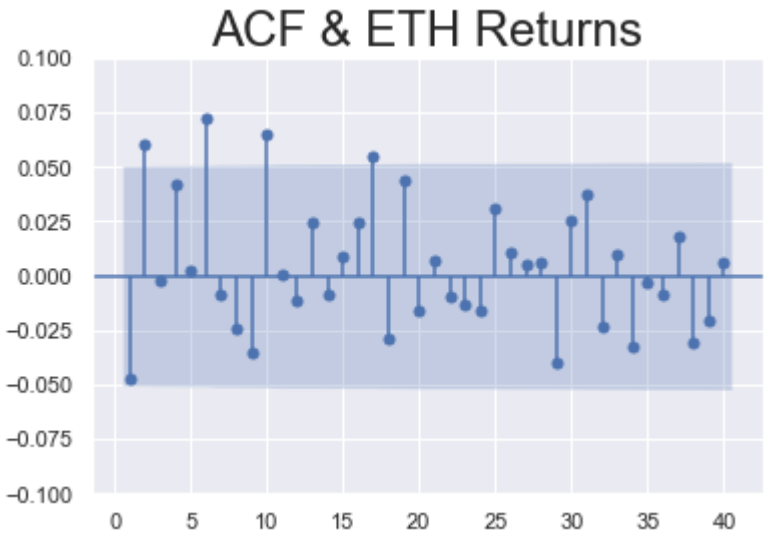
MA model

In [116…
```
from statsmodels.tsa.arima_process import ArmaProcess
```

In [117…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.returns, lags = 40, zero = False)
```

```
plt.title("ACF & ETH Returns", size = 24)
```

Out[117… (-0.1, 0.1)



In [126…
```
#ORDER first - AR, Second - Differences, Third - MA
model_ret_ma_1 = ARIMA(eth_train.returns[1:], order = (0,0,2))
results_ret_ma_1 = model_ret_ma_1.fit()
```

In [127…
```
results_ret_ma_1.summary()
```

Out[127…

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	1500
Model:	ARIMA(0, 0, 2)	**Log Likelihood**	-4595.308
Date:	Sun, 05 Feb 2023	**AIC**	9198.617
Time:	09:58:50	**BIC**	9219.869
Sample:	11-12-2017	**HQIC**	9206.534
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.3059	0.137	2.236	0.025	0.038	0.574
ma.L1	-0.0445	0.019	-2.283	0.022	-0.083	-0.006
ma.L2	0.0558	0.024	2.316	0.021	0.009	0.103
sigma2	26.8212	0.507	52.930	0.000	25.828	27.814

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	2037.74
Prob(Q):	1.00	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.90	**Skew:**	-0.26
Prob(H) (two-sided):	0.25	**Kurtosis:**	8.69

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

In [128…
```
from statsmodels.tsa.arima.model import ARIMA
```

In [143…
```
#ORDER first - AR, Second - Differences, Third - MA
model_ret_ma_2 = ARIMA(eth_train.returns[1:], order = (0,0,6))
results_ret_ma_2 = model_ret_ma_2.fit()
results_ret_ma_2.summary()
```

Out[143…

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	1500
Model:	ARIMA(0, 0, 6)	**Log Likelihood**	-4590.325
Date:	Sun, 05 Feb 2023	**AIC**	9196.649
Time:	10:02:30	**BIC**	9239.155
Sample:	11-12-2017	**HQIC**	9212.484
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.3062	0.154	1.987	0.047	0.004	0.608
ma.L1	-0.0443	0.020	-2.219	0.027	-0.083	-0.005
ma.L2	0.0606	0.025	2.396	0.017	0.011	0.110
ma.L3	0.0038	0.024	0.161	0.872	-0.043	0.050
ma.L4	0.0321	0.020	1.565	0.118	-0.008	0.072
ma.L5	0.0093	0.022	0.428	0.669	-0.033	0.052
ma.L6	0.0737	0.024	3.100	0.002	0.027	0.120
sigma2	26.6425	0.552	48.261	0.000	25.560	27.724

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	2027.82
Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.90	**Skew:**	-0.26
Prob(H) (two-sided):	0.26	**Kurtosis:**	8.67

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

In [144…
```
print("\nLLR Test P-value = " + str(LLR_test(model_ret_ma_1,model_ret_ma_2, DF = 4)))
```

```
LLR Test P-value = 0.041
```

Residual Analysis

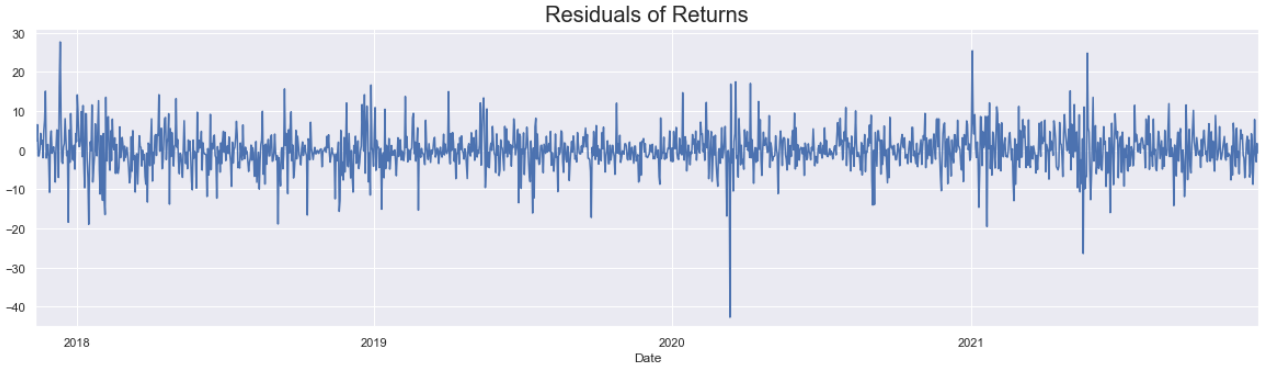
In [149…
```
eth_train['res_ret_ma_2'] = results_ret_ma_2.resid[1:]
```

In [151…
```
print("mean is " + str(round(eth_train.res_ret_ma_2.mean(),3)))
print("variance is " + str(round(eth_train.res_ret_ma_2.var(),3)))
print("Standard deviation is " + str(round(sqrt(eth_train.res_ret_ma_2.var()), 3)))
```

```
mean is 0.002
variance is 26.675
Standard deviation is 5.165
```

In [154…
```
eth_train.res_ret_ma_2[1:].plot(figsize = (20,5))
plt.title("Residuals of Returns", size = 20)
plt.ylim(bottom = -45)
```

Out[154… (-45.0, 31.210916519188622)

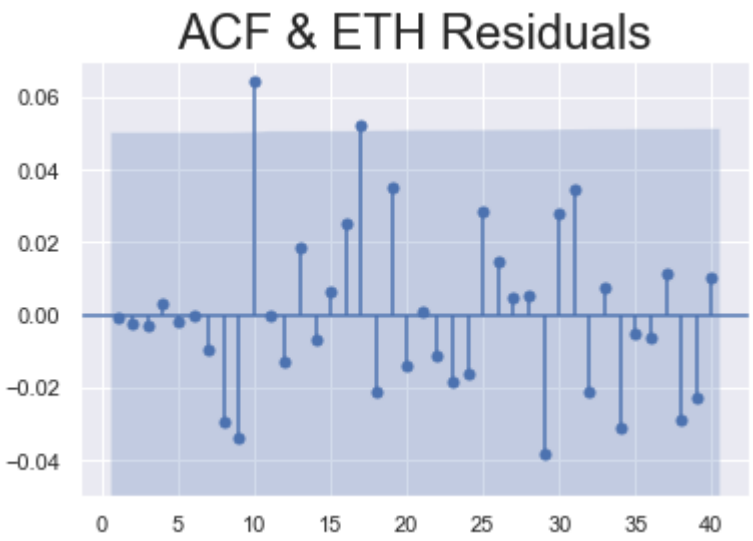


In [155…
```
sts.adfuller(eth_train.res_ret_ma_2[2:])
```

Out[155…
```
(-38.70434351882141,
 0.0,
 0,
 1498,
 {'1%': -3.4347228578139943,
  '5%': -2.863471337969528,
  '10%': -2.5677982210726897},
 9027.645156416089)
```

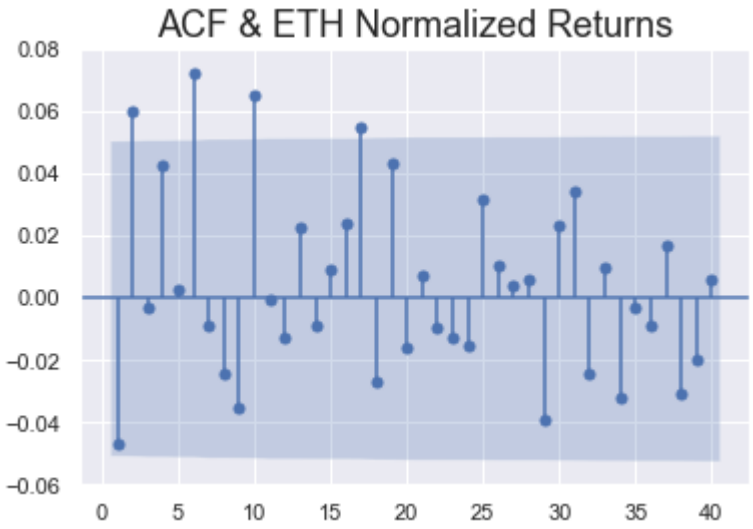
In [158…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.res_ret_ma_2[2:], lags = 40, zero = False)
plt.title("ACF & ETH Residuals", size = 24)
plt.ylim(top = 0.07, bottom = -0.05)
```

Out[158… (-0.05, 0.07)



In [161…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.norm_ret[1:], lags = 40, zero = False)
plt.title("ACF & ETH Normalized Returns", size = 18)
plt.ylim(top = 0.08, bottom = -0.06)
```

Out[161…
```
(-0.06, 0.08)
```



In [163…
```
#ORDER first - AR, Second - Differences, Third - MA
model_norm_ret_ma_1 = ARIMA(eth_train.norm_ret[1:], order = (0,0,6))
results_norm_ret_ma_1 = model_norm_ret_ma_1.fit()
results_norm_ret_ma_1.summary()
```

Out[163…

SARIMAX Results

Dep. Variable:	norm_ret	**No. Observations:**	1500
Model:	ARIMA(0, 0, 6)	**Log Likelihood**	-9037.983
Date:	Sun, 05 Feb 2023	**AIC**	18091.966
Time:	10:22:37	**BIC**	18134.472
Sample:	11-12-2017	**HQIC**	18107.801
	- 12-20-2021		

Covariance Type: opg

	coef	std err	z	P>|z|	[0.025	0.975]
const	5.9335	2.994	1.982	0.047	0.065	11.802
ma.L1	-0.0443	0.020	-2.214	0.027	-0.084	-0.005
ma.L2	0.0606	0.025	2.392	0.017	0.011	0.110
ma.L3	0.0038	0.024	0.161	0.872	-0.043	0.050
ma.L4	0.0321	0.021	1.562	0.118	-0.008	0.072
ma.L5	0.0093	0.022	0.427	0.669	-0.033	0.052
ma.L6	0.0736	0.024	3.094	0.002	0.027	0.120
sigma2	1.004e+04	208.508	48.167	0.000	9634.619	1.05e+04

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	2027.81
Prob(Q):	0.98	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.90	**Skew:**	-0.26
Prob(H) (two-sided):	0.26	**Kurtosis:**	8.67

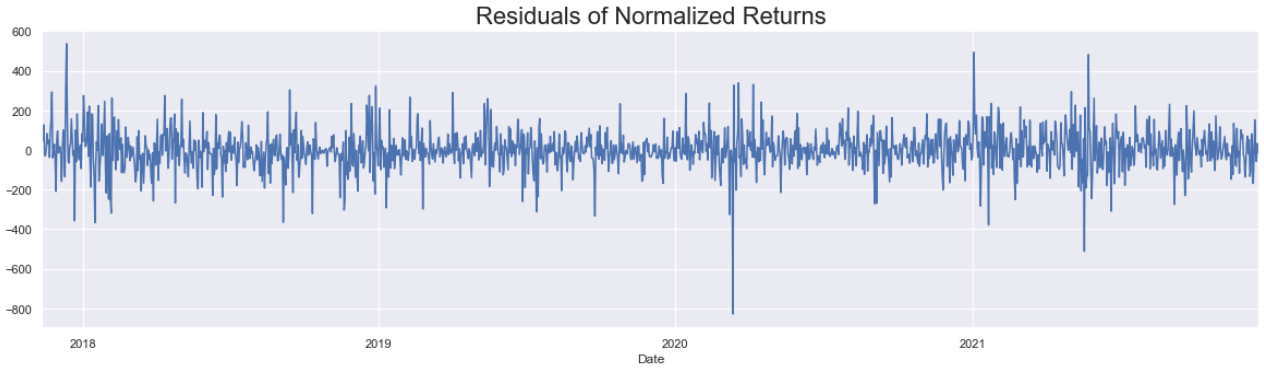
Warnings:

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

In [165…
```
eth_train['res_norm_ret_ma_1'] = results_norm_ret_ma_1.resid[1:]
```

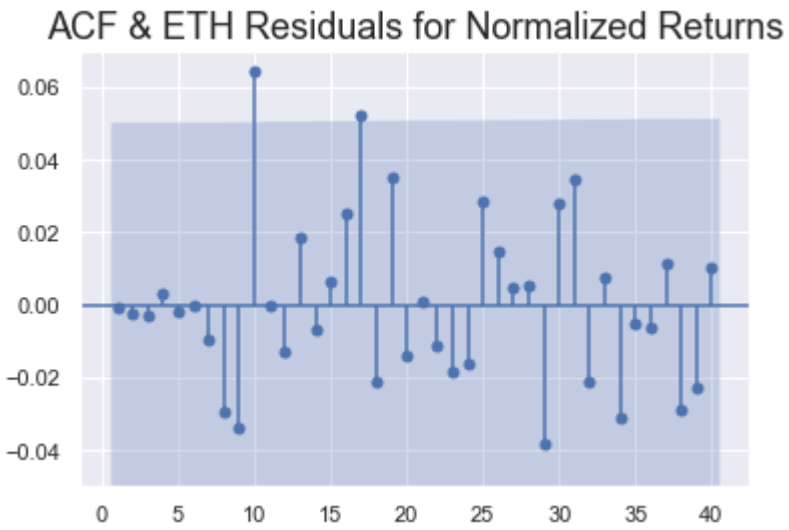
In [166…
```
eth_train.res_norm_ret_ma_1[1:].plot(figsize = (20,5))
plt.title("Residuals of Normalized Returns", size = 22)
```

Out[166…
```
Text(0.5, 1.0, 'Residuals of Normalized Returns')
```



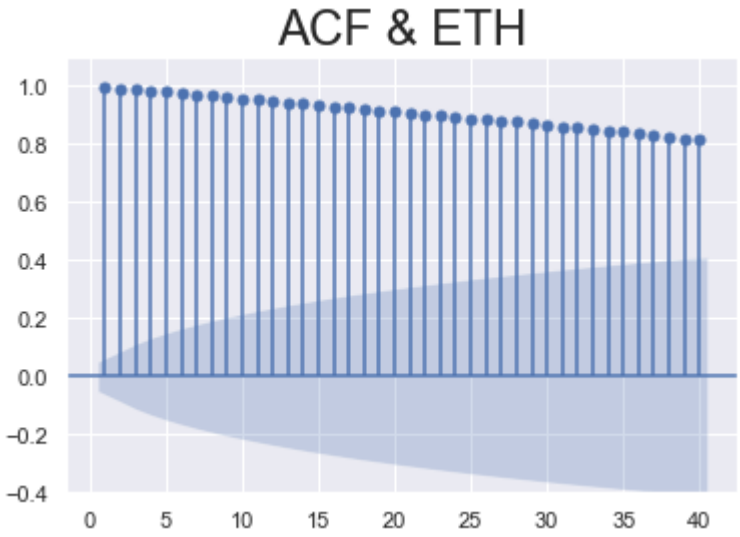
In [167…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.res_norm_ret_ma_1[2:], lags = 40, zero = False)
plt.title("ACF & ETH Residuals for Normalized Returns", size = 18)
plt.ylim(top = 0.07, bottom = -0.05)
```

Out[167…
```
(-0.05, 0.07)
```



In [168…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.Close, lags = 40, zero = False)
plt.title("ACF & ETH", size = 24)
plt.ylim(top = 1.10, bottom = -0.4)
```

Out[168… (-0.4, 1.1)



ARMA

In [193…
```
#ORDER first - AR, Second - Differences, Third - MA
model_ret_arma_1 = ARIMA(eth_train.returns[1:], order = (2,0,1))
results_ret_arma_1 = model_ret_arma_1.fit()
results_ret_arma_1.summary()
```

Out[193…

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	1500
Model:	ARIMA(2, 0, 1)	**Log Likelihood**	-4594.296
Date:	Sun, 05 Feb 2023	**AIC**	9198.593
Time:	19:28:40	**BIC**	9225.159

Sample:	11-12-2017	**HQIC**	9208.489
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.3075	0.156	1.972	0.049	0.002	0.613
ar.L1	0.7088	0.146	4.868	0.000	0.423	0.994
ar.L2	0.0790	0.022	3.637	0.000	0.036	0.122
ma.L1	-0.7570	0.146	-5.182	0.000	-1.043	-0.471
sigma2	26.7859	0.508	52.753	0.000	25.791	27.781

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	1996.32
Prob(Q):	0.96	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.91	**Skew:**	-0.25
Prob(H) (two-sided):	0.28	**Kurtosis:**	8.63

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

In [202…
```
#ORDER first - AR, Second - Differences, Third - MA
model_ret_arma_2 = ARIMA(eth_train.returns[1:], order = (4,0,3))
results_ret_arma_2 = model_ret_arma_2.fit()
results_ret_arma_2.summary()
```

Out[202…

SARIMAX Results

Dep. Variable:	returns	**No. Observations:**	1500
Model:	ARIMA(4, 0, 3)	**Log Likelihood**	-4592.187
Date:	Sun, 05 Feb 2023	**AIC**	9202.374
Time:	19:30:16	**BIC**	9250.193
Sample:	11-12-2017	**HQIC**	9220.189
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.3060	0.136	2.244	0.025	0.039	0.573
ar.L1	0.4299	0.172	2.493	0.013	0.092	0.768
ar.L2	-0.0078	0.185	-0.042	0.966	-0.371	0.355
ar.L3	-0.6632	0.161	-4.125	0.000	-0.978	-0.348
ar.L4	0.0355	0.030	1.202	0.229	-0.022	0.093

ma.L1	-0.4733	0.173	-2.737	0.006	-0.812	-0.134
ma.L2	0.0662	0.200	0.331	0.741	-0.326	0.459
ma.L3	0.6228	0.176	3.531	0.000	0.277	0.968
sigma2	26.7031	0.540	49.487	0.000	25.646	27.761

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	1830.23
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.89	**Skew:**	-0.27
Prob(H) (two-sided):	0.20	**Kurtosis:**	8.39

Warnings:

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

In [204…
```
LLR_test(model_ret_arma_1, model_ret_arma_2, DF = 4)
```

Out[204…
```
0.377
```

In [206…
```
print("ARMA(2,1) AIC Value is " + str(results_ret_arma_1.aic))
print("ARMA(4,3) AIC Value is " + str(results_ret_arma_2.aic))
```

```
ARMA(2,1) AIC Value is 9198.592608872896
ARMA(4,3) AIC Value is 9202.37437426682
```

In [165…
```
results_ret_arma_2.llf
```

Out[165…
```
-6710.809957162166
```

In [166…
```
results_ret_arma_1.aic
```

Out[166…
```
13436.094945896086
```

In [167…
```
results_ret_arma_1.llf
```

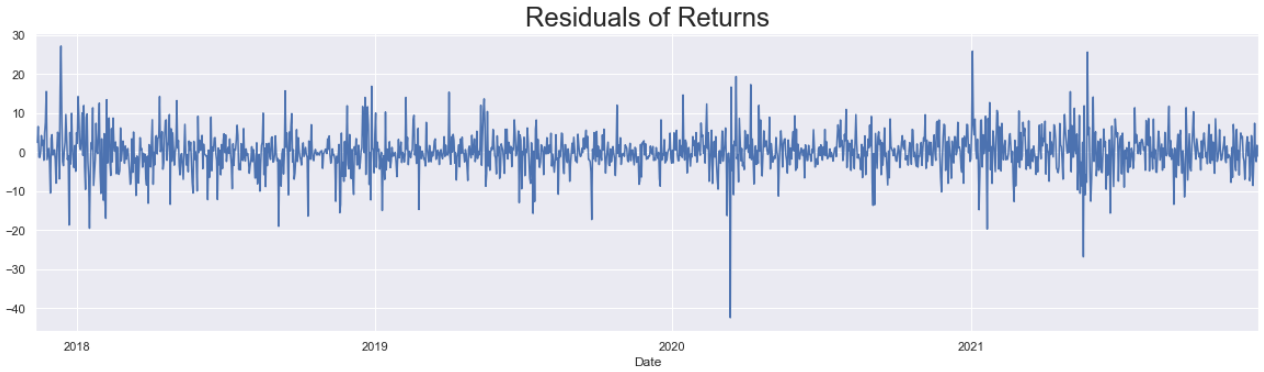
Out[167…
```
-6710.047472948043
```

Residuals for Returns

In [208…
```
eth_train['res_ret_arma_1'] = results_ret_arma_1.resid[1:]
```

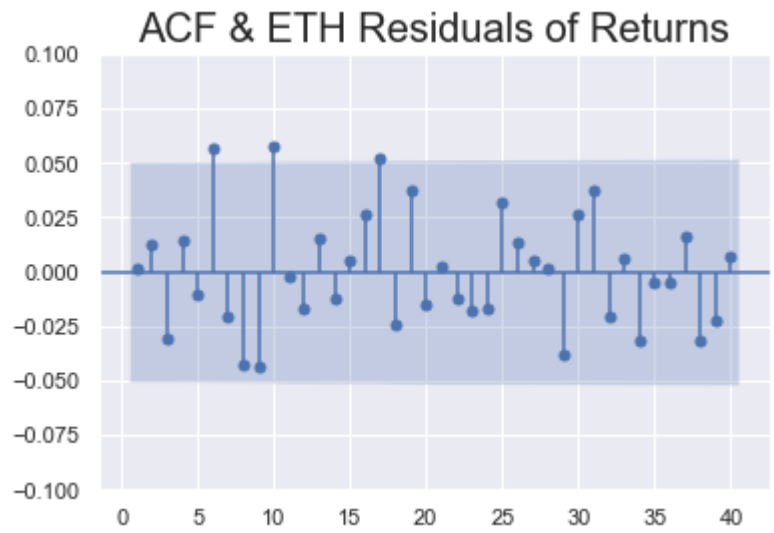
In [209…
```
eth_train.res_ret_arma_1.plot(figsize = (20,5))
plt.title('Residuals of Returns', size = 24)
```

Out[209…
```
Text(0.5, 1.0, 'Residuals of Returns')
```



In [210…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.res_ret_arma_1[2:], lags = 40, zero = False)
plt.title("ACF & ETH Residuals of Returns", size = 20)
plt.ylim(top = 0.1, bottom = -0.1)
```

Out[210…
```
(-0.1, 0.1)
```



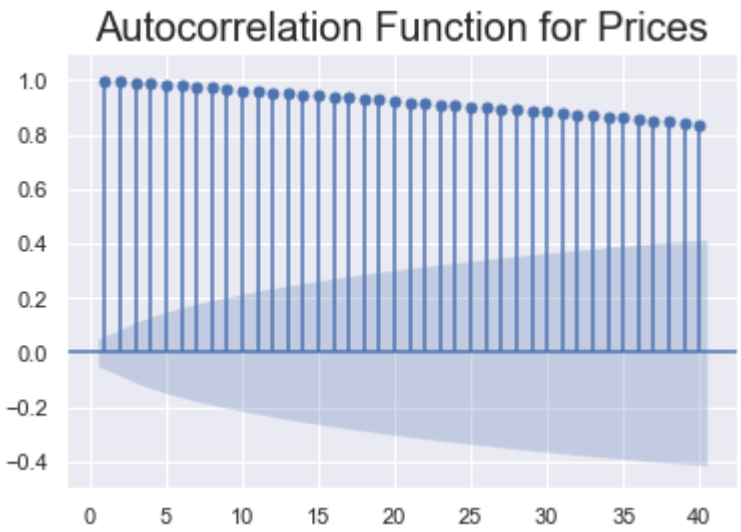
if the model is not sufficient try with different lags in ARMA and follow above steps again

Reevaluating Model

ARMA MODEL FOR Close Prices

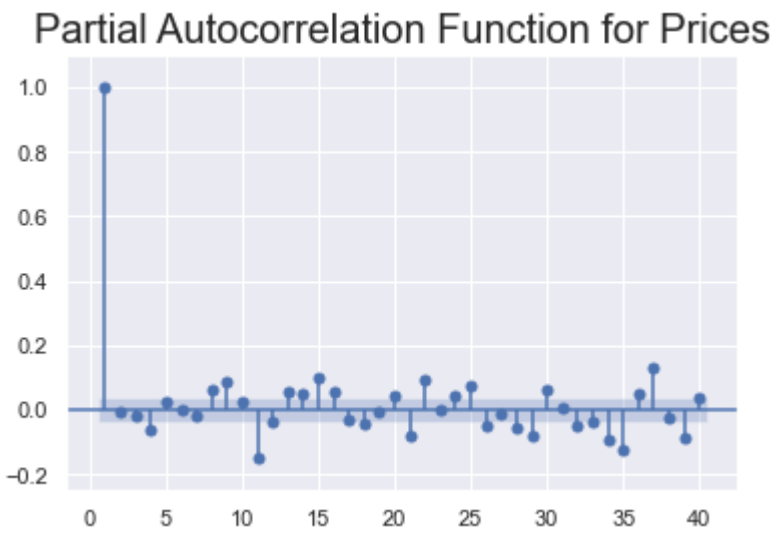
In [211…
```
sgt.plot_acf(eth_train.Close, unbiased=True, zero = False, lags = 40)
plt.title("Autocorrelation Function for Prices",size=20)
plt.ylim(top = 1.1, bottom = -0.5)
```

Out[211…
```
(-0.5, 1.1)
```



In [181…
```
sgt.plot_pacf(btc_train.Close, lags = 40, alpha = 0.05, zero = False , method = ('ols')
plt.title("Partial Autocorrelation Function for Prices",size=20)
plt.ylim(top = 1.1, bottom = -0.25)
```

Out[181… (-0.25, 1.1)



In [213…
```
#ORDER first - AR, Second - Differences, Third - MA
model_arma_1 = ARIMA(eth_train.Close, order = (2,0,1))
results_arma_1 = model_arma_1.fit()
results_arma_1.summary()
```

Out[213…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	1501
Model:	ARIMA(2, 0, 1)	**Log Likelihood**	-8621.952
Date:	Sun, 05 Feb 2023	**AIC**	17253.903
Time:	19:50:01	**BIC**	17280.473
Sample:	11-11-2017	**HQIC**	17263.801
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	901.9933	4246.570	0.212	0.832	-7421.130	9225.117
ar.L1	0.1776	0.038	4.723	0.000	0.104	0.251
ar.L2	0.8206	0.038	21.762	0.000	0.747	0.895
ma.L1	0.7546	0.045	16.701	0.000	0.666	0.843
sigma2	5689.9125	64.767	87.852	0.000	5562.972	5816.853

Ljung-Box (L1) (Q):	0.94	**Jarque-Bera (JB):**	32269.73
Prob(Q):	0.33	**Prob(JB):**	0.00
Heteroskedasticity (H):	13.78	**Skew:**	-0.87
Prob(H) (two-sided):	0.00	**Kurtosis:**	25.65

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

In [214…
```
#ORDER first - AR, Second - Differences, Third - MA
model_arma_2 = ARIMA(eth_train.Close, order = (4,0,3))
results_arma_2 = model_arma_2.fit()
results_arma_2.summary()
```

Out[214…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	1501
Model:	ARIMA(4, 0, 3)	**Log Likelihood**	-8604.261
Date:	Sun, 05 Feb 2023	**AIC**	17226.521
Time:	19:50:11	**BIC**	17274.346
Sample:	11-11-2017	**HQIC**	17244.337
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	901.9887	7667.143	0.118	0.906	-1.41e+04	1.59e+04
ar.L1	-0.4027	0.039	-10.316	0.000	-0.479	-0.326
ar.L2	-0.0239	0.021	-1.150	0.250	-0.065	0.017
ar.L3	0.6351	0.021	30.485	0.000	0.594	0.676
ar.L4	0.7892	0.040	19.785	0.000	0.711	0.867
ma.L1	1.2980	0.046	28.035	0.000	1.207	1.389
ma.L2	1.3148	0.033	40.120	0.000	1.251	1.379
ma.L3	0.6887	0.047	14.568	0.000	0.596	0.781

sigma2	5628.1247	69.517	80.961	0.000	5491.874	5764.375

Ljung-Box (L1) (Q):	0.50	**Jarque-Bera (JB):**	29975.64
Prob(Q):	0.48	**Prob(JB):**	0.00
Heteroskedasticity (H):	13.15	**Skew:**	-0.84
Prob(H) (two-sided):	0.00	**Kurtosis:**	24.83

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

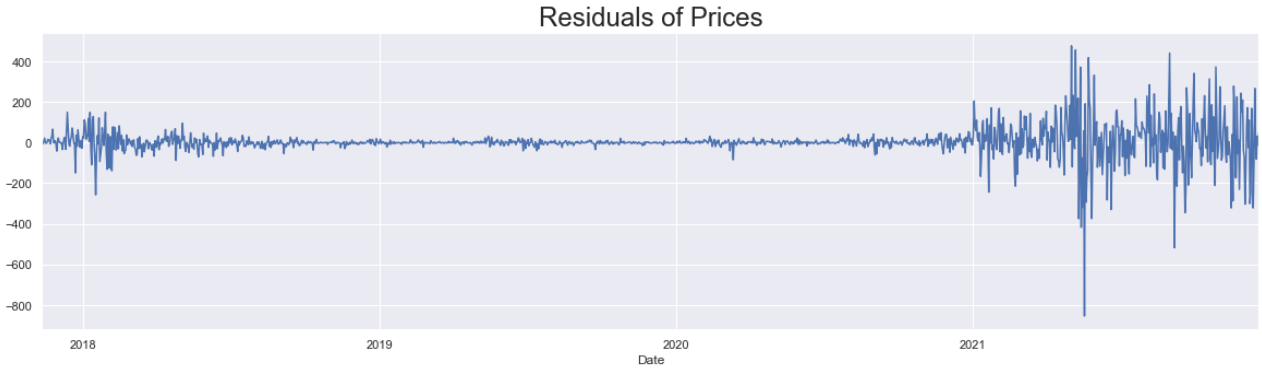
In [215…
```
LLR_test(model_arma_1, model_arma_2, DF = 4)
```

Out[215…
```
0.0
```

In [216…
```
eth_train['res_arma'] = results_arma_2.resid[1:]
```

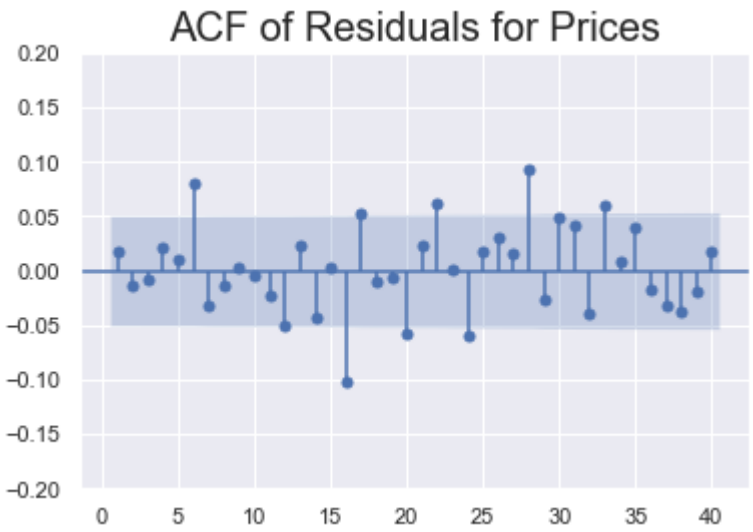
In [217…
```
eth_train.res_arma.plot(figsize = (20,5))
plt.title('Residuals of Prices', size = 24)
```

Out[217…
```
Text(0.5, 1.0, 'Residuals of Prices')
```



In [218…
```
sgt.plot_acf(eth_train.res_arma[1:], zero = False, lags = 40)
plt.title("ACF of Residuals for Prices",size=20)
plt.ylim(top = 0.2, bottom = -0.2)
```

Out[218…
```
(-0.2, 0.2)
```



Choose the best lags model and do AIC and LLF comparison as we have done in simple lag

ARIMA Model

In [220…

```
model_arima_1 = ARIMA(eth_train.Close[1:], order = (1,1,1))
results_model_arima1 = model_arima_1.fit()
results_model_arima1.summary()
```

Out[220…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	1500
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-8610.795
Date:	Mon, 06 Feb 2023	**AIC**	17227.591
Time:	06:10:09	**BIC**	17243.528
Sample:	11-12-2017	**HQIC**	17233.528
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.1412	0.102	-1.389	0.165	-0.340	0.058
ma.L1	0.0417	0.104	0.403	0.687	-0.161	0.245
sigma2	5729.8158	62.758	91.301	0.000	5606.813	5852.819

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	35473.23
Prob(Q):	0.96	**Prob(JB):**	0.00
Heteroskedasticity (H):	13.86	**Skew:**	-1.03
Prob(H) (two-sided):	0.00	**Kurtosis:**	26.74

Warnings:

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

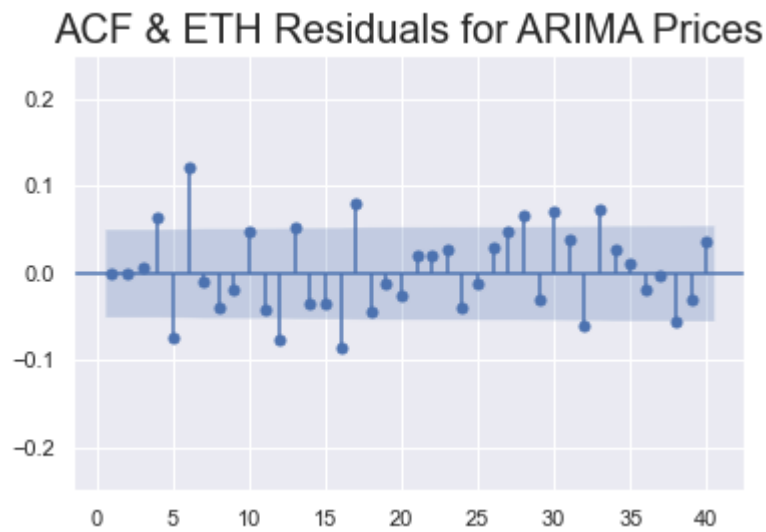
Residuals of simple ARIMA Model

In [221…
```
eth_train['results_model_arima1'] = results_model_arima1.resid[1:]
eth_train.results_model_arima1.plot(figsize = (20,5))
plt.title('Residuals of Close Prices', size = 24)
```

Out[221…
```
Text(0.5, 1.0, 'Residuals of Close Prices')
```

In [222…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.results_model_arima1[2:], lags = 40, zero = False)
plt.title("ACF & ETH Residuals for ARIMA Prices", size = 20)
plt.ylim(top = 0.25, bottom = -0.25)
```

Out[222…
```
(-0.25, 0.25)
```



In [226…
```
# 1
model_arima_1 = ARIMA(eth_train.Close[1:], order = (1,1,1))
results_model_arima_1 = model_arima_1.fit()
results_model_arima1.summary()
# 2
model_arima_2 = ARIMA(eth_train.Close[1:], order = (1,1,2))
results_model_arima_2 = model_arima_2.fit()
results_model_arima_2.summary()
# 3
model_arima_3 = ARIMA(eth_train.Close[1:], order = (1,1,3))
results_model_arima_3 = model_arima_3.fit()
results_model_arima_3.summary()
```

```
# 4
model_arima_4 = ARIMA(eth_train.Close[1:], order = (2,1,1))
results_model_arima_4 = model_arima_4.fit()
# 5
model_arima_5 = ARIMA(eth_train.Close[1:], order = (3,1,1))
results_model_arima_5 = model_arima_5.fit()

# 6
model_arima_6 = ARIMA(eth_train.Close[1:], order = (3,1,2))
results_model_arima_6 = model_arima_6.fit()
```

FIND LL AND AIC

In [227…

```
print("ARIMA(1,1,1):  \t LL = ", results_model_arima_1.llf, "\t AIC = ", results_model_
print("ARIMA(1,1,2):  \t LL = ", results_model_arima_2.llf, "\t AIC = ", results_model_
print("ARIMA(1,1,3):  \t LL = ", results_model_arima_3.llf, "\t AIC = ", results_model_
print("ARIMA(2,1,1):  \t LL = ", results_model_arima_4.llf, "\t AIC = ", results_model_
print("ARIMA(3,1,1):  \t LL = ", results_model_arima_5.llf, "\t AIC = ", results_model_
print("ARIMA(3,1,2):  \t LL = ", results_model_arima_6.llf, "\t AIC = ", results_model_
```

```
ARIMA(1,1,1):    LL =  -8610.795256554851        AIC =  17227.590513109702
ARIMA(1,1,2):    LL =  -8610.12597071617         AIC =  17228.25194143234
ARIMA(1,1,3):    LL =  -8605.290078714881        AIC =  17220.580157429762
ARIMA(2,1,1):    LL =  -8607.247994566285        AIC =  17222.49598913257
ARIMA(3,1,1):    LL =  -8610.15517355717         AIC =  17230.31034711434
ARIMA(3,1,2):    LL =  -8607.247371915331        AIC =  17226.494743830663
```

Do the LLR test with and without DF

In [231…

```
print("\nLLR test p-value = " + str(LLR_test(model_arima_1, model_arima_3, DF = 2)))
print("\nLLR test p-value = " + str(LLR_test(model_arima_2, model_arima_3)))
print("\nLLR test p-value = " + str(LLR_test(model_arima_4, model_arima_3)))
print("\nLLR test p-value = " + str(LLR_test(model_arima_5, model_arima_3)))
print("\nLLR test p-value = " + str(LLR_test(model_arima_6, model_arima_3)))
```

```
LLR test p-value = 0.004

LLR test p-value = 0.002

LLR test p-value = 0.048

LLR test p-value = 0.002

LLR test p-value = 0.048
```

In [232…

```
eth_train['delta_prices']=eth_train.Close.diff(1)
```

In [233…

```
sts.adfuller(eth_train.delta_prices[1:])
```

Out[233…

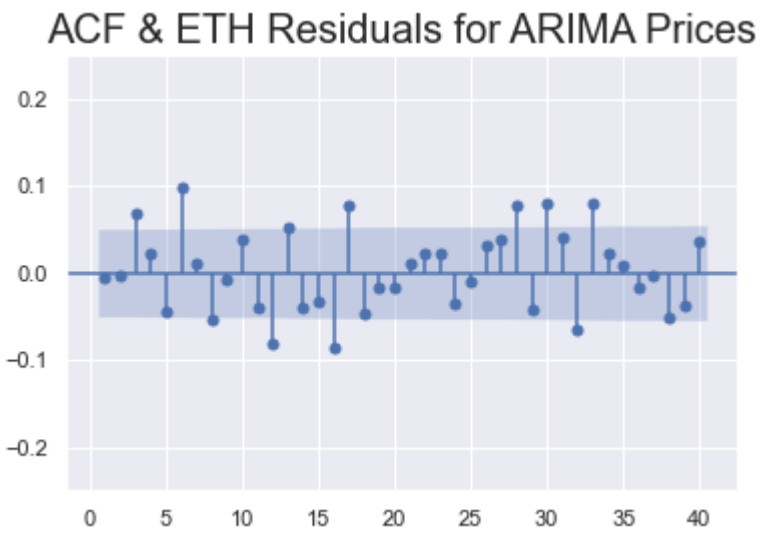
```
(-10.144321399562491,
 8.262347504369866e-18,
 16,
 1483,
 {'1%': -3.4347671645756304,
  '5%': -2.86349089226533,
  '10%': -2.5678086339403325},
 16914.969030305292)
```

In [235…

```
eth_train['res_arima'] = results_model_arima_3.resid[1:]
```

In [236…
```
#Auto Correlation, 40 lags means the last 40 lags before the current one
sgt.plot_acf(eth_train.res_arima[2:], lags = 40, zero = False)
plt.title("ACF & ETH Residuals for ARIMA Prices", size = 20)
plt.ylim(top = 0.25, bottom = -0.25)
```

Out[236…
```
(-0.25, 0.25)
```



In [246…
```
btc_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2421 entries, 2014-09-18 to 2021-05-04
Freq: D
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Open                  2421 non-null   float64
 1   High                  2421 non-null   float64
 2   Low                   2421 non-null   float64
 3   Close                 2421 non-null   float64
 4   Volume                2421 non-null   int64  
 5   returns               2421 non-null   float64
 6   norm                  2421 non-null   float64
 7   norm_ret              2421 non-null   float64
 8   res_price             2421 non-null   float64
 9   res_price_ret         2420 non-null   float64
 10  res_ret_ma_2          2419 non-null   float64
 11  res_norm_ret_ma_1     2419 non-null   float64
 12  res_ret_arma_1        2419 non-null   float64
 13  res_arma              2420 non-null   float64
 14  results_model_arima1  2419 non-null   float64
 15  delta_prices          2420 non-null   float64
 16  res_arima             2419 non-null   float64
dtypes: float64(16), int64(1)
memory usage: 340.5 KB
```

In [247…
```
eth_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1503 entries, 0 to 1502
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   Date       1503 non-null   datetime64[ns]
 1   Open       1503 non-null   float64       
 2   High       1503 non-null   float64       
```

```
 3   Low        1503 non-null   float64
 4   Close      1503 non-null   float64
 5   Adj Close  1503 non-null   float64
 6   Volume     1503 non-null   int64
dtypes: datetime64[ns](1), float64(5), int64(1)
memory usage: 82.3 KB
```

ARIMAX

In [245…
```
btc_eth = pd.DataFrame(btcdata['2017-11-09':])
btc_eth.head()
```

Out[245…

Date	Open	High	Low	Close	Volume
2017-11-09	7446.830078	7446.830078	7101.520020	7143.580078	3226249984
2017-11-10	7173.729980	7312.000000	6436.870117	6618.140137	5208249856
2017-11-11	6618.609863	6873.149902	6204.220215	6357.600098	4908680192
2017-11-12	6295.450195	6625.049805	5519.009766	5950.069824	8957349888
2017-11-13	5938.250000	6811.189941	5844.290039	6559.490234	6263249920

In [249…
```
eth_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1501 entries, 2017-11-11 to 2021-12-20
Freq: D
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Open                  1501 non-null   float64
 1   High                  1501 non-null   float64
 2   Low                   1501 non-null   float64
 3   Close                 1501 non-null   float64
 4   Volume                1501 non-null   int64
 5   returns               1501 non-null   float64
 6   norm                  1501 non-null   float64
 7   norm_ret              1501 non-null   float64
 8   res_price             1500 non-null   float64
 9   res_price_ret         1500 non-null   float64
 10  res_ret_ma_2          1499 non-null   float64
 11  res_norm_ret_ma_1     1499 non-null   float64
 12  res_ret_arma_1        1499 non-null   float64
 13  res_arma              1500 non-null   float64
 14  results_model_arima1  1499 non-null   float64
 15  delta_prices          1500 non-null   float64
 16  res_arima             1499 non-null   float64
dtypes: float64(16), int64(1)
memory usage: 243.4 KB
```

In [250…
```
btc_eth.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1879 entries, 2017-11-09 to 2022-12-31
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Open    1879 non-null   float64
```

```
 1   High    1879 non-null   float64
 2   Low     1879 non-null   float64
 3   Close   1879 non-null   float64
 4   Volume  1879 non-null   int64
dtypes: float64(4), int64(1)
memory usage: 152.6 KB
```

In [366…

```
#will be done for other Cryptos
model_arimax = ARIMA(eth_train.Close, exog = btc_eth['2017-11-11':"2021-12-20"].Close,
results_arimax = model_arimax.fit()
results_arimax.summary()
```

Out[366…

SARIMAX Results

Dep. Variable:	Close	**No. Observations:**	1501
Model:	ARIMA(1, 1, 1)	**Log Likelihood**	-8070.893
Date:	Thu, 09 Feb 2023	**AIC**	16149.787
Time:	07:19:59	**BIC**	16171.040
Sample:	11-11-2017	**HQIC**	16157.704
	- 12-20-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
Close	0.0539	0.001	91.295	0.000	0.053	0.055
ar.L1	-0.8277	0.040	-20.582	0.000	-0.906	-0.749
ma.L1	0.7707	0.048	16.070	0.000	0.677	0.865
sigma2	2760.8465	36.778	75.067	0.000	2688.762	2832.931

Ljung-Box (L1) (Q):	1.53	**Jarque-Bera (JB):**	24575.34
Prob(Q):	0.22	**Prob(JB):**	0.00
Heteroskedasticity (H):	8.57	**Skew:**	0.31
Prob(H) (two-sided):	0.00	**Kurtosis:**	22.82

Warnings:

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

ARCH Model

In [280…

```
from arch import arch_model
```

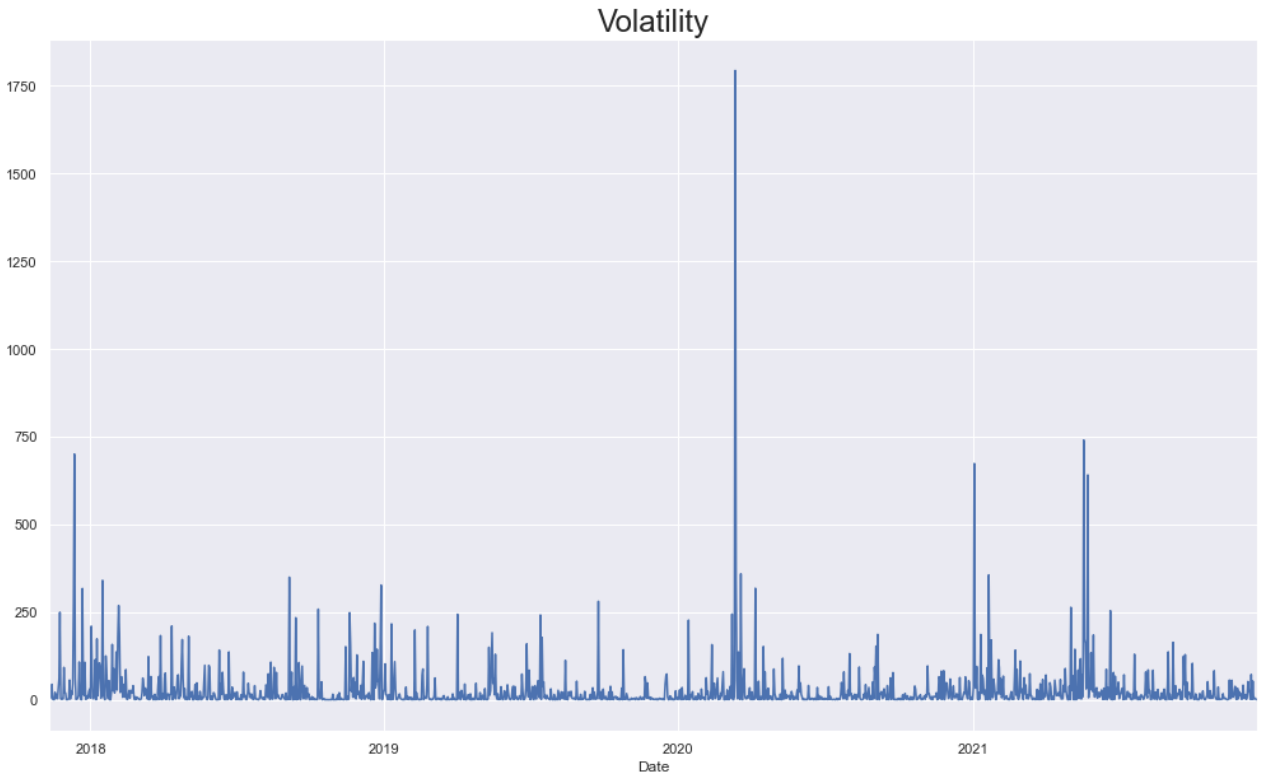
In [297…

```
eth_train["sq_returns"] = eth_train.returns.mul(eth_train.returns)
eth_test["sq_returns"] = eth_test.returns.mul(eth_test.returns)
```

In [302…

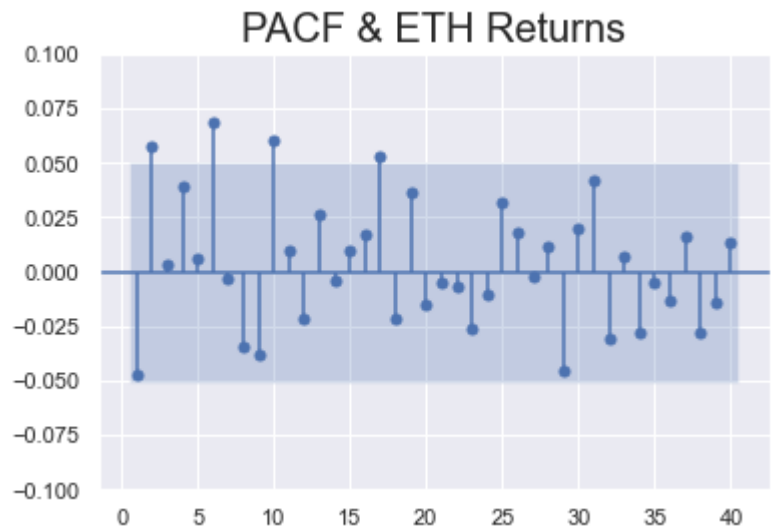
```
eth_train.sq_returns.plot(figsize = (17,10))
plt.title('Volatility', size = 24)
```

Out[302…
```
Text(0.5, 1.0, 'Volatility')
```



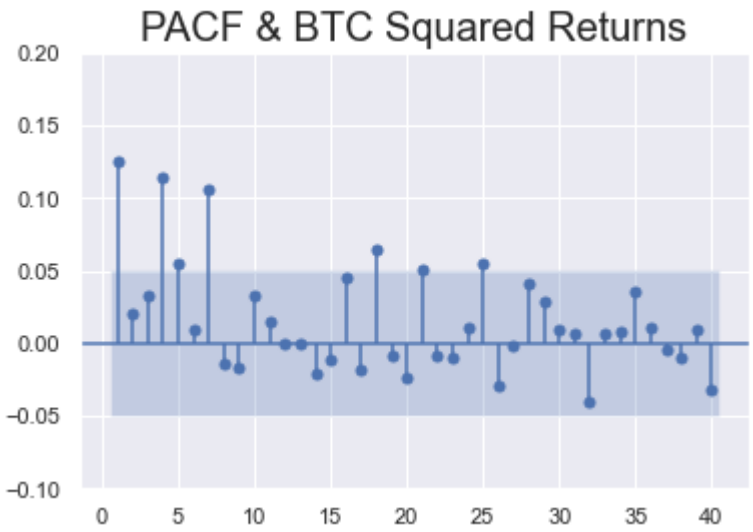
In [303…
```
sgt.plot_pacf(eth_train.returns, lags = 40, zero = False, method = ('ols'))
plt.title("PACF & ETH Returns", size = 20)
plt.ylim(top = 0.1, bottom = -0.1)
```

Out[303…
```
(-0.1, 0.1)
```



In [304…
```
sgt.plot_pacf(eth_train.sq_returns, lags = 40, zero = False, method = ('ols'))
plt.title("PACF & BTC Squared Returns", size = 20)
plt.ylim(top = 0.2, bottom = -0.1)
```

Out[304…
```
(-0.1, 0.2)
```



Simple ARCH MODEL

In [305…

```
model_arch_1 = arch_model(eth_train.returns[1:], mean = "Constant", vol = "ARCH", p = 1
results_model_arch_1 = model_arch_1.fit(update_freq= 5)
results_model_arch_1.summary()
```

```
Iteration:      5,   Func. Count:     24,   Neg. LLF: 4584.4271084422235
Optimization terminated successfully    (Exit mode 0)
            Current function value: 4584.427108442574
            Iterations: 5
            Function evaluations: 24
            Gradient evaluations: 5
```

Out[305…

Constant Mean - ARCH Model Results

Dep. Variable:	returns	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	ARCH	**Log-Likelihood:**	-4584.43
Distribution:	Normal	**AIC:**	9174.85
Method:	Maximum Likelihood	**BIC:**	9190.79
		No. Observations:	1500
Date:	Mon, Feb 06 2023	**Df Residuals:**	1499
Time:	18:18:59	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	0.2893	0.128	2.267	2.339e-02	[3.919e-02, 0.539]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	24.0554	2.094	11.485	1.566e-30	[19.950, 28.160]
alpha[1]	0.1080	4.793e-02	2.253	2.429e-02	[1.403e-02, 0.202]

Covariance estimator: robust

Change mean from constant to "ZERO" or "AR" and see the results and interpret. If you you use mean AR then mention lags

In [307…
```
model_arch_2 = arch_model(eth_train.returns[1:], mean = "Constant", vol = "ARCH", p = 2
results_model_arch_2 = model_arch_2.fit(update_freq= 5)
results_model_arch_2.summary()
```

```
Iteration:      5,   Func. Count:     33,   Neg. LLF: 4581.118867061585
Optimization terminated successfully    (Exit mode 0)
            Current function value: 4581.118692035994
            Iterations: 8
            Function evaluations: 47
            Gradient evaluations: 8
```

Out[307…

Constant Mean - ARCH Model Results

Dep. Variable:	returns	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	ARCH	**Log-Likelihood:**	-4581.12
Distribution:	Normal	**AIC:**	9170.24
Method:	Maximum Likelihood	**BIC:**	9191.49
		No. Observations:	1500
Date:	Mon, Feb 06 2023	**Df Residuals:**	1499
Time:	18:24:40	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	0.3124	0.126	2.477	1.325e-02	[6.522e-02, 0.560]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	22.5594	2.610	8.643	5.458e-18	[17.444, 27.675]
alpha[1]	0.1037	4.948e-02	2.097	3.600e-02	[6.777e-03, 0.201]
alpha[2]	0.0644	5.161e-02	1.247	0.212	[-3.678e-02, 0.166]

Covariance estimator: robust

Garch Model

In [309…
```
model_arch_1 = arch_model(eth_train.returns[1:], mean = "Constant", vol = "GARCH", p =
results_model_arch_1 = model_arch_1.fit(update_freq= 5)
results_model_arch_1.summary()
```

```
Iteration:      5,   Func. Count:     37,   Neg. LLF: 4587.226593391046
Iteration:     10,   Func. Count:     70,   Neg. LLF: 4539.310877518201
Optimization terminated successfully    (Exit mode 0)
```

```
Current function value: 4539.289679038354
Iterations: 13
Function evaluations: 89
Gradient evaluations: 13
```

Out[309…

Constant Mean - GARCH Model Results

Dep. Variable:	returns	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	GARCH	**Log-Likelihood:**	-4539.29
Distribution:	Normal	**AIC:**	9088.58
Method:	Maximum Likelihood	**BIC:**	9115.15
		No. Observations:	1500
Date:	Mon, Feb 06 2023	**Df Residuals:**	1499
Time:	18:28:00	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	0.2692	0.119	2.263	2.367e-02	[3.600e-02, 0.502]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	1.6650	0.918	1.813	6.985e-02	[-0.135, 3.465]
alpha[1]	0.0825	3.462e-02	2.382	1.721e-02	[1.462e-02, 0.150]
beta[1]	0.6945	0.384	1.809	7.050e-02	[-5.809e-02, 1.447]
beta[2]	0.1625	0.362	0.449	0.654	[-0.547, 0.872]

Covariance estimator: robust

try higher lags if necessary

Forecasting

In [310…

```
import yfinance
```

In [312…

```
eth_train.tail()
```

Out[312…

Date	Open	High	Low	Close	Volume	returns	norm	norm
2021-12-16	4020.415039	4110.368652	3956.057129	3962.469727	19825531254	-1.391576	1259.202089	-26.992
2021-12-17	3959.012451	3992.792480	3711.424561	3879.486572	23143541098	-2.094228	1232.831525	-40.621

Date	Open	High	Low	Close	Volume	returns	norm	norm
2021-12-18	3880.291504	3993.829834	3774.614990	3960.860107	19530895889	2.097534	1258.690581	40.685
2021-12-19	3960.872314	4018.658447	3894.398682	3922.592529	16167785597	-0.966143	1246.529828	-18.740
2021-12-20	3923.695801	3980.098633	3759.403320	3933.844482	21589690675	0.286850	1250.105498	5.563

In [315…
```
start_date = "2021-12-21"
end_date = "2022-12-31"
```

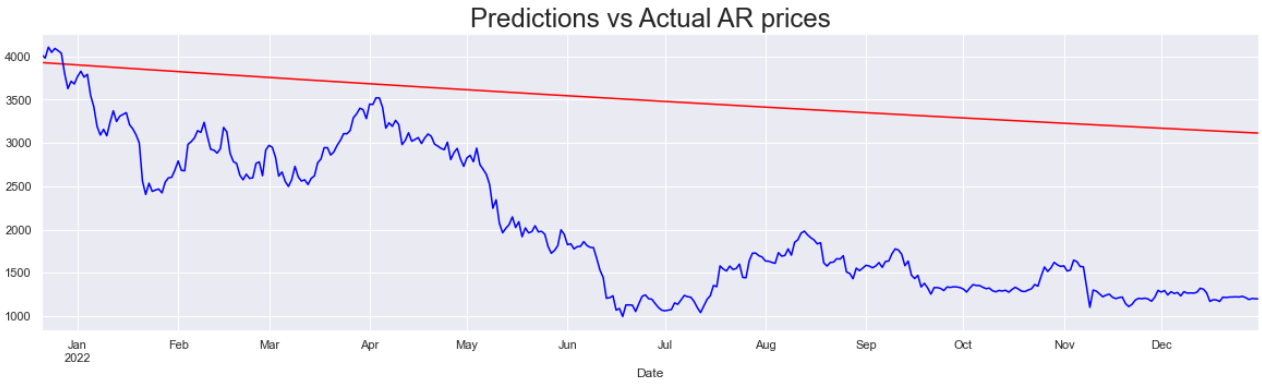
AR Forecasting

In [316…
```
pred_ar = results_ar_model_1.predict(start = start_date, end = end_date)
```

In [317…
```
pred_ar
```

Out[317…
```
2021-12-21    3929.776522
2021-12-22    3927.460277
2021-12-23    3924.900475
2021-12-24    3922.375380
2021-12-25    3919.848014
                 ...
2022-12-27    3121.739494
2022-12-28    3119.882545
2022-12-29    3118.027149
2022-12-30    3116.173306
2022-12-31    3114.321014
Freq: D, Name: predicted_mean, Length: 376, dtype: float64
```

In [319…
```
pred_ar[start_date:end_date].plot(figsize = (20,5), color = "red")
eth_test.Close[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual AR prices", size = 24)
plt.show()
```

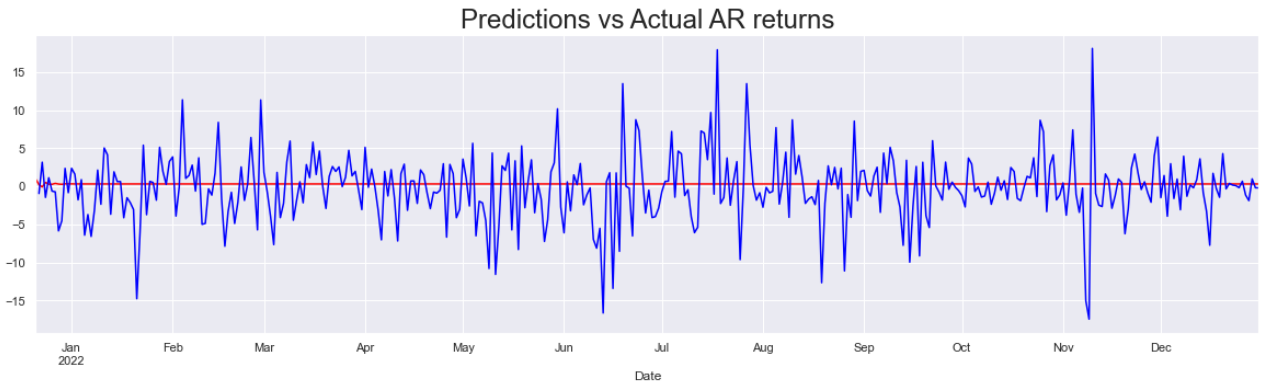


Using Returns for Forecasting

In [320…
```
pred_ret_ar = results_ar_model_ret_2.predict(start = start_date, end = end_date)
```

In [321…
```
pred_ret_ar[start_date:end_date].plot(figsize = (20,5), color = "red")
eth_test.returns[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual AR returns", size = 24)
```

Out[321… Text(0.5, 1.0, 'Predictions vs Actual AR returns')



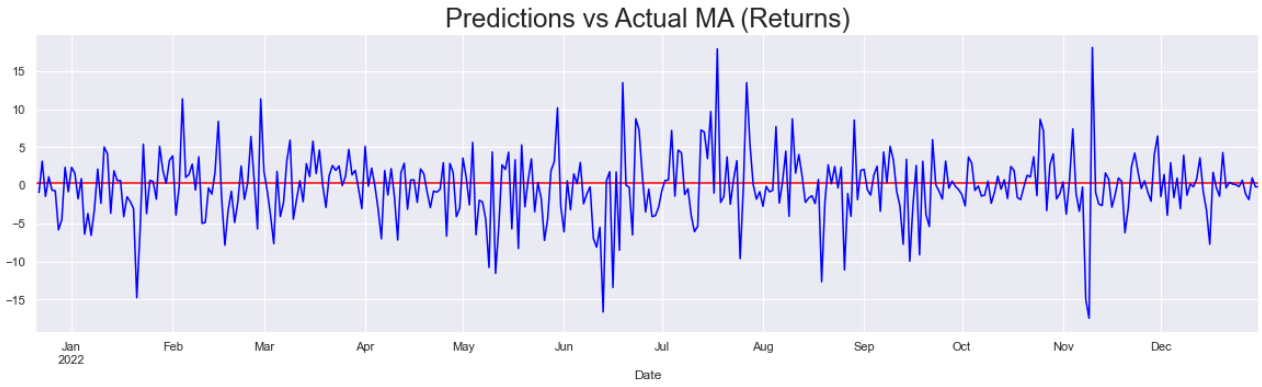
In [ ]:

MA Forecasting

In [322…
```
pred_ma_ret = results_ret_ma_1.predict(start = start_date, end = end_date)
```

In [323…
```
pred_ma_ret[start_date:end_date].plot(figsize = (20,5), color = "red")
eth_test.returns[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual MA (Returns)", size = 24)
```

Out[323… Text(0.5, 1.0, 'Predictions vs Actual MA (Returns)')

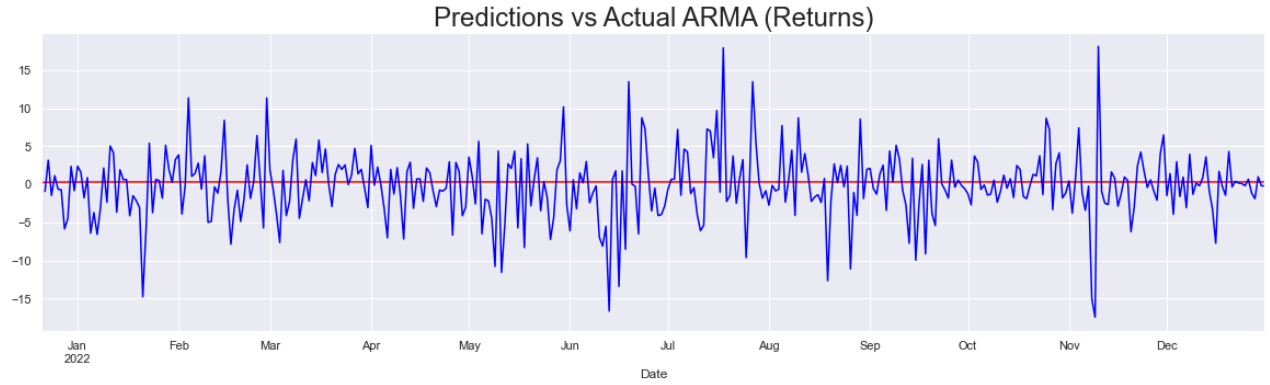


ARMA Forecasting

In [324…
```
pred_arma_ret = results_ret_arma_1.predict(start = start_date, end = end_date)
```

In [325…
```
pred_arma_ret[start_date:end_date].plot(figsize = (20,5), color = "red")
eth_test.returns[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual ARMA (Returns)", size = 24)
```

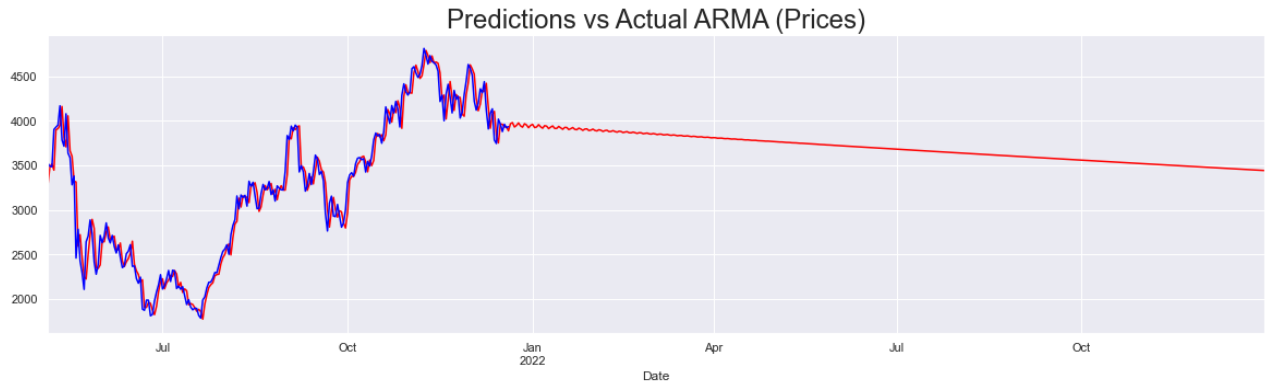
Out[325… Text(0.5, 1.0, 'Predictions vs Actual ARMA (Returns)')



In [266…
```
pred_arma_prices = results_arma_2.predict(start = start_date, end = end_date)
```

In [267…
```
pred_arma_prices[start_date:end_date].plot(figsize = (20,5), color = "red")
eth_train.Close[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual ARMA (Prices)", size = 24)
```

Out[267…
```
Text(0.5, 1.0, 'Predictions vs Actual ARMA (Prices)')
```



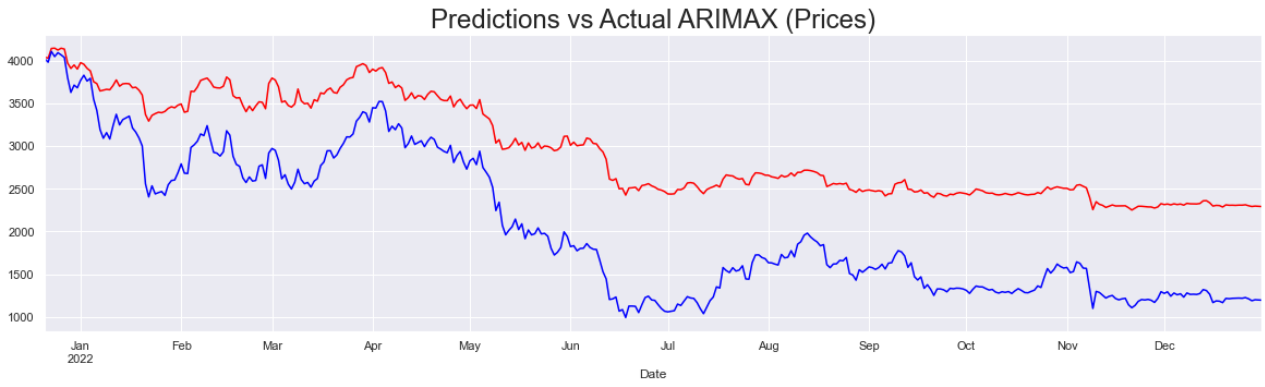
In [ ]:
```
```

ARMAX Forecasting

In [362…
```
pred_armax = results_arimax.predict(start = start_date, end = end_date,
                                    exog = btcdata.Close[start_date:end_date])

pred_armax[start_date:end_date].plot(figsize = (20,5), color = "red")
eth_test.Close[start_date:end_date].plot(color = "blue")
plt.title("Predictions vs Actual ARIMAX (Prices)", size = 24)
```

Out[362…
```
Text(0.5, 1.0, 'Predictions vs Actual ARIMAX (Prices)')
```



In [363…
```
pred_armax.tail()
```

Out[363…
```
2022-12-27    2301.959332
2022-12-28    2293.057829
2022-12-29    2297.912484
2022-12-30    2295.762523
2022-12-31    2292.783315
Freq: D, Name: predicted_mean, dtype: float64
```

Forecasting Volatility

In [516…
```
mod_garch = arch_model(btc_train.returns[1:], vol = 'GARCH', p = 1, q = 1, mean = 'cons
res_garch = mod_garch.fit(last_obs=start_date, update_freq = 10)
```

```
Iteration:     10,   Func. Count:     64,   Neg. LLF: 6486.725847041355
Optimization terminated successfully    (Exit mode 0)
            Current function value: 6486.725847041356
            Iterations: 10
            Function evaluations: 64
            Gradient evaluations: 10
```

In [518…
```
pred_garch = res_garch.forecast()
```

In [523…
```
pred_garch.mean
```

Out[523…

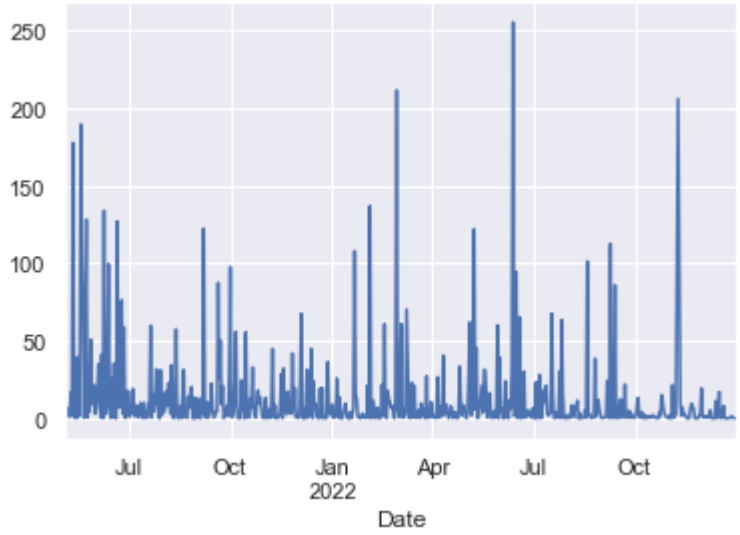
Date	h.1
2014-09-19	NaN
2014-09-20	NaN
2014-09-21	NaN
2014-09-22	NaN
2014-09-23	NaN
…	…
2021-04-30	NaN
2021-05-01	NaN
2021-05-02	NaN
2021-05-03	NaN

	h.1
Date	
2021-05-04	0.239181

2420 rows × 1 columns

In [487…
```
btc_test.sq_returns.plot()
pred_garch
```

Out[487…
```
<AxesSubplot:xlabel='Date'>
```



In [474…
```
start_date = "2021-05-05"
end_date = "2022-12-31"
```