Exercise 4

Task 1

Copy the 'ARIMA_example.py' code, tidy it up (remove unnecessary parts) and modify it to analyze stock exchange data. For example, go to www.marketwatch.com , seek some company using magnifying glass, click stock value and then 'Historical Quotes'. You can download .csv data from there, take 3 months (daily values) at minimum.

nvda_data.csv contains last 3kk daily values from 18.9. - 18.12.2024. Downloaded from Investing.com.

Import libraries

In [203…
```
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.ar_model import ar_select_order
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
import scipy.io.wavfile as wav
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
from statsmodels.graphics.api import qqplot
import pandas as pd
```

Read and save data into dataframe.

In [204…
```
# Load the data
df = pd.read_csv('nvda_data.csv')
```

Modify dataframe to acceptable form for future computations.

In [205…
```
# convert , to . in the in all the columns
df = df.map(lambda x: str(x).replace(',', '.'))
stock_prices = df['Viim.'].astype(float)
```

Analyze data using snippets from ARIMA_example.py.

In [206…
```
#define z-score for partial autocorrelation analysis
#https://en.wikipedia.org/wiki/Standard_score

z_score=3 #equal to 3-sigma i.e. ~99% of the data
len_segment=len(stock_prices) #length of the segment for analysis
#https://en.wikipedia.org/wiki/Partial_autocorrelation_function
reference=z_score/np.sqrt(len_segment)

print(len_segment)
nlag=16 #number of lags for partial autocorrelation analysis
y = stock_prices.values
```

```
segment=y[0:int(0.8*len_segment)]
test_segment = y[int(0.8*len_segment):]

y=y-np.mean(y)          #standardixe close price, zero mean, min-max -1...1
y=y/np.abs(np.max(y))

plt.plot(segment)
plt.title('Normalized Close Price')
plt.show()
```

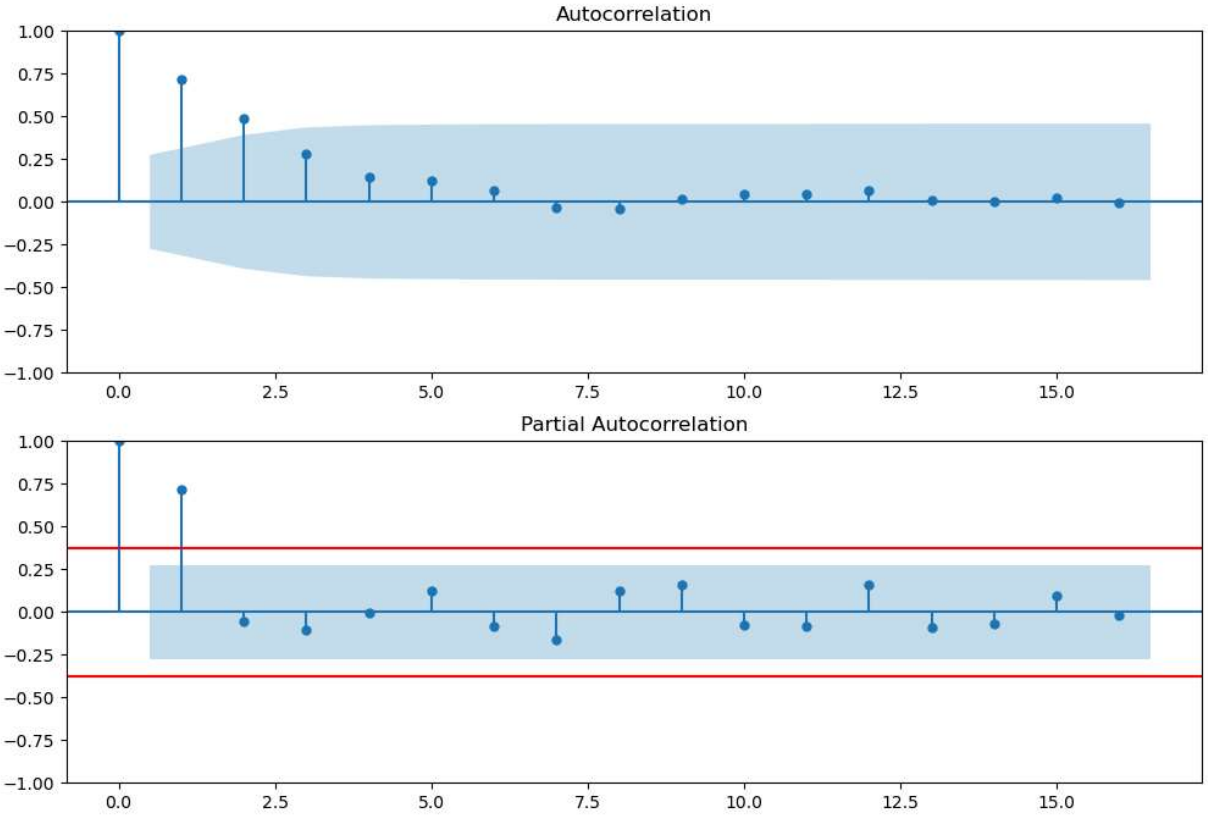
64

In [207…

```
#show autocorrelation and partial autocorrelation of the segment
#Look at the place of first change of sign in the images
print("Let us define the ARIMA(p, d, q) parameters")

print("\n*** Look at the biggest index (lag) from partial autocorrelation (pacf) im
print("Select the p = order (lag) reference value +1 ***")

print("q can be estimated similarly by looking at the ACF plot instead of the PACF
print("Looking at the number of lags crossing the threshold, we can determine how m
print("would be significant enough to consider for the future.")
print("The ones with high correlation contribute more and would be enough to predic

fig = plt.figure(figsize=(12, 8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(segment.squeeze(), lags=nlag, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(segment, lags=nlag, ax=ax2)
ax2.axhline(y=reference, color='r')
```

```
ax2.axhline(y=-reference, color='r')
plt.show()
```

```
Let us define the ARIMA(p, d, q) parameters

*** Look at the biggest index (lag) from partial autocorrelation (pacf) image that i
s still larger than reference value
Select the p = order (lag) reference value +1 ***
q can be estimated similarly by looking at the ACF plot instead of the PACF plot.
Looking at the number of lags crossing the threshold, we can determine how much of t
he past
would be significant enough to consider for the future.
The ones with high correlation contribute more and would be enough to predict future
values
```



Select p = lag +1 which would be 3 by analyzing PACF. Thus, q = 3 since there are two dots above blue threshold.

Select p, q = 3 for arima model and d = 1 since 1st order diff p-value is below 0.05 limit.

In [208…

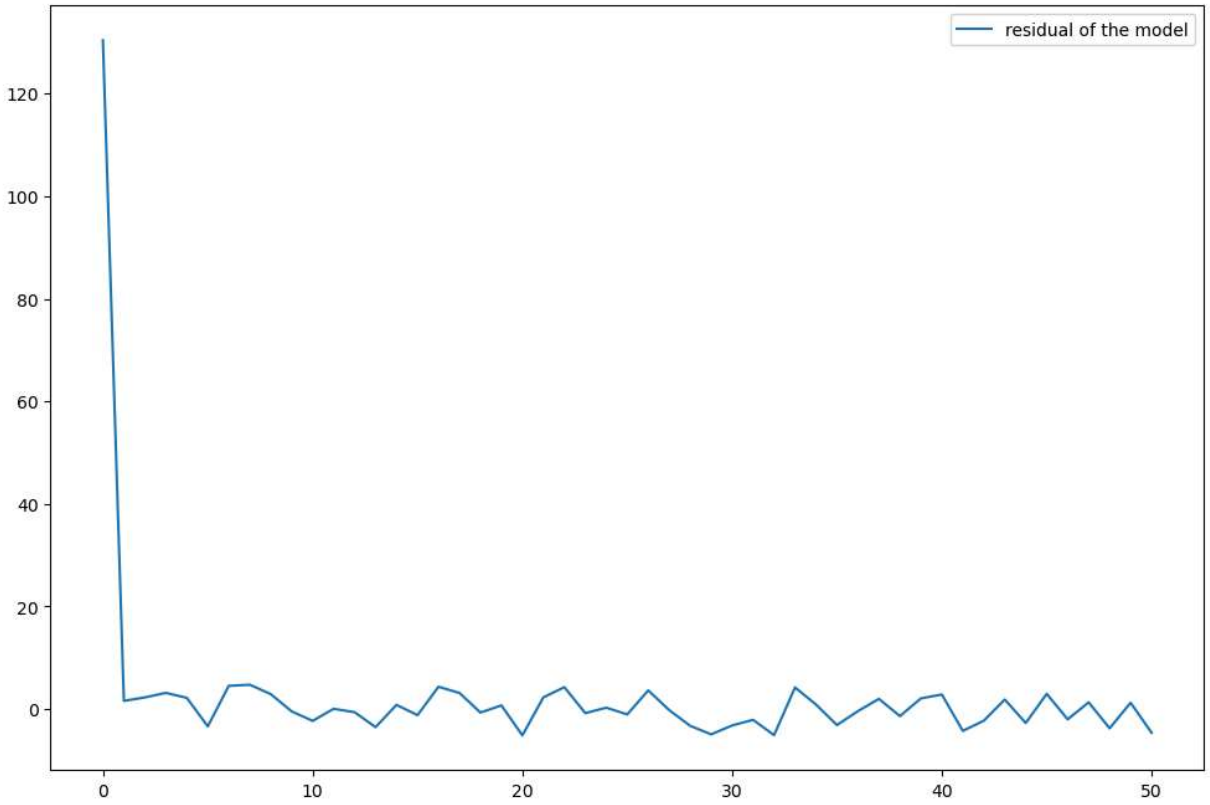
```
#https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html
print("\nLet us find the d (differencing) value, the first one that goes below 0.05
result = adfuller(segment)
print('p-value (0.05): ', result[1])
result = adfuller(np.diff(segment))
print('1st order diff p-value (0.05): ', result[1])
result = adfuller(np.diff(np.diff(segment)))
print('2nd order diff p-value (0.05): ', result[1])
result = adfuller(np.diff(np.diff(np.diff(segment))))
print('3rd order diff p-value (0.05): ', result[1])
```

```
arma_model=ARIMA(segment,order=(5,1,5)).fit()
print("\n\nModel summary")
print(arma_model.summary())

print(arma_model.params)
print("\nPrint AIC, BIC, HQIC values for full autocorrelation")
print(arma_model.aic, arma_model.bic, arma_model.hqic)
```

```
Let us find the d (differencing) value, the first one that goes below 0.05 limit:
p-value (0.05):  0.22135440682558966
1st order diff p-value (0.05):  5.6162327595840084e-09
2nd order diff p-value (0.05):  0.00015386706471009958
3rd order diff p-value (0.05):  6.348331118506215e-06


Model summary
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   51
Model:                 ARIMA(5, 1, 5)   Log Likelihood                -123.554
Date:                Fri, 20 Dec 2024   AIC                            269.108
Time:                        14:29:20   BIC                            290.140
Sample:                             0   HQIC                           277.117
                                  - 51
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7904      0.600      1.317      0.188      -0.385       1.966
ar.L2         -0.0371      0.854     -0.044      0.965      -1.710       1.636
ar.L3          0.4666      0.518      0.901      0.368      -0.549       1.482
ar.L4         -0.2626      0.691     -0.380      0.704      -1.618       1.092
ar.L5         -0.1020      0.509     -0.200      0.841      -1.099       0.895
ma.L1         -0.8505    151.551     -0.006      0.996    -297.884     296.183
ma.L2          0.1736    460.417      0.000      1.000    -902.227     902.575
ma.L3         -0.8184    896.829     -0.001      0.999   -1758.571    1756.934
ma.L4          0.0483     58.624      0.001      0.999    -114.852     114.948
ma.L5          0.5946    392.134      0.002      0.999    -767.975     769.164
sigma2         7.3259   4827.443      0.002      0.999   -9454.288    9468.940
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 2.64
Prob(Q):                              1.00   Prob(JB):                         0.27
Heteroskedasticity (H):               1.00   Skew:                            -0.14
Prob(H) (two-sided):                  0.99   Kurtosis:                         1.91
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-ste
p).
[ 0.79037365 -0.03713507  0.46660676 -0.26264073 -0.10197482 -0.85049927
  0.17359449 -0.81844127  0.04825801  0.59456412  7.32586343]

Print AIC, BIC, HQIC values for full autocorrelation
269.1079744360958 290.14022749580545 277.11717635964163
```

In [209…
```
#print residual values, we can estimate the goodness of the fit by analyzing the re
#The closer to 0 the statistic, the more evidence for positive serial correlation.
#The closer to 4, the more evidence for negative serial correlation.
#https://www.statsmodels.org/stable/generated/statsmodels.stats.stattools.durbin_wa
print("\nPrint residual values")
print(sm.stats.durbin_watson(arma_model.resid))

fig = plt.figure(figsize=(12, 8))
plt.plot(arma_model.resid,label='residual of the model')
plt.legend()
plt.show()

print("\nResidual normalization test for both")
print("Full")
print(stats.normaltest(arma_model.resid))
```

```
Print residual values
0.9973192296903002
```



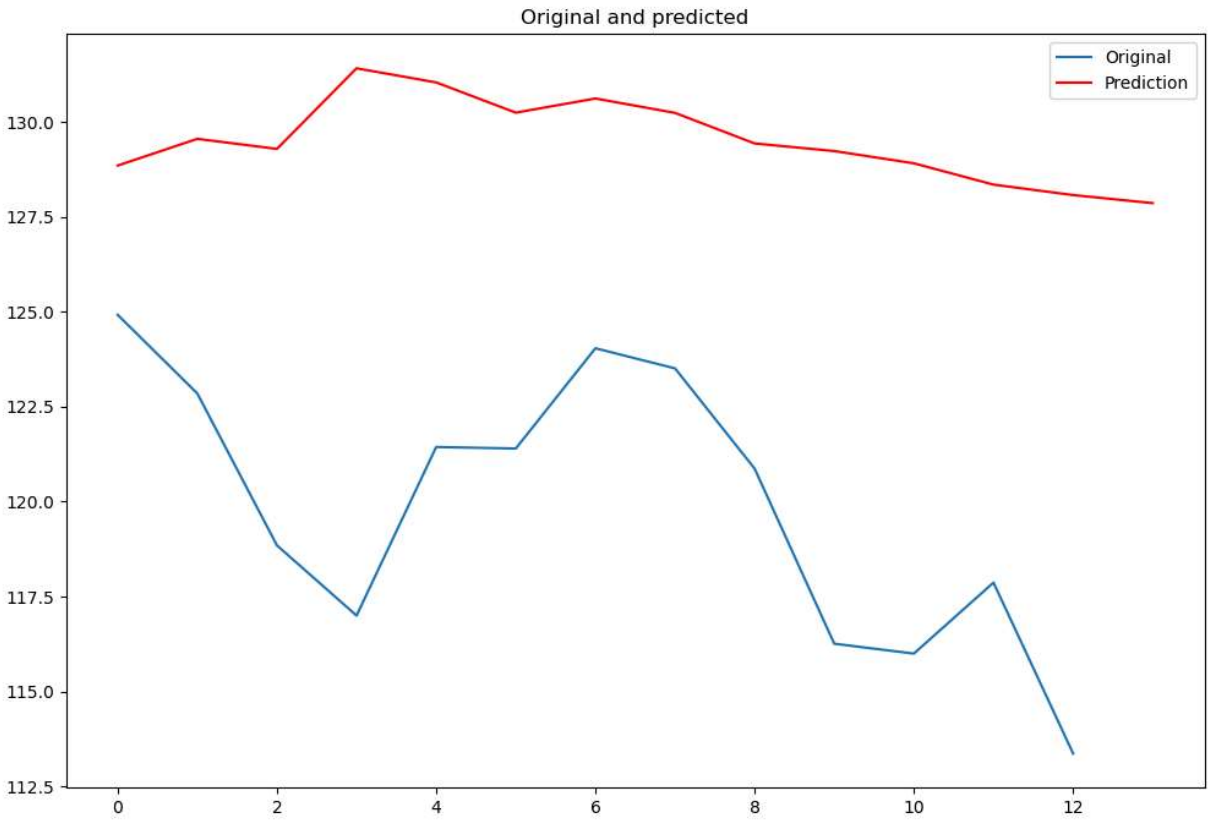

```
Residual normalization test for both
Full
NormaltestResult(statistic=111.56009713755157, pvalue=5.957070039415146e-25)
```

Residual value indicates positive autocorrelation and plot indicates that model performance is good since residual values appear near 0.

In [210…
```
#Organize data to neat datatable format
r, q, p = sm.tsa.acf(arma_model.resid.squeeze(), fft=True, qstat=True)

# Ensure all arrays have the same length
min_length = min(len(r[1:]), len(q), len(p))
r = r[1:min_length+1]
```

```
q = q[:min_length]
p = p[:min_length]

data = np.c_[np.arange(1, min_length+1), r, q, p]

table = pd.DataFrame(data, columns=["lag", "AC", "Q", "Prob(>Q)"])
print("\nDataFrame")
print(table.set_index("lag"))

#finally, predictions from model

predict_prices = arma_model.predict(start=len(segment), end=len(stock_prices), dyna
fig = plt.figure(figsize=(12, 8))
plt.plot(test_segment,label='Original')
plt.plot(predict_prices,'r', label='Prediction')
plt.title("Original and predicted")
plt.legend()
plt.show()
```

```
DataFrame
            AC         Q  Prob(>Q)
lag
1.0   0.011838  0.007576  0.930639
2.0   0.016929  0.023385  0.988376
3.0   0.022346  0.051504  0.996939
4.0   0.015513  0.065344  0.999478
5.0  -0.026245  0.105819  0.999813
6.0   0.033265  0.172287  0.999900
7.0   0.033841  0.242640  0.999951
8.0   0.018727  0.264684  0.999988
9.0  -0.004178  0.265808  0.999998
10.0 -0.017750  0.286578  1.000000
11.0 -0.004808  0.288140  1.000000
12.0 -0.006195  0.290800  1.000000
13.0 -0.033687  0.371522  1.000000
14.0  0.002379  0.371936  1.000000
15.0 -0.010807  0.380704  1.000000
16.0  0.028644  0.444068  1.000000
17.0  0.018270  0.470606  1.000000
```



By trying different parameters, the results wont come better than this. ARIMA model is not able to predict the bearish trend for the stock. Further optimization, would require more data or different features. Also different approach to model could be tested, since there are differently constructed implementations and tutorials on web.

Task 2

Copy the 'ARIMA_example.py' code, tidy it up (remove unnecessary parts) and import the pickle file: "Nepal_electricity_consumption_in_MWh.pkl" from 4th folder. The seasonality is very clear in this data.

Import pickle file

In [268…

```
# Load pickle file
import pickle

# Load pickle file, contains dataset
with open('Nepal_electricity_consumption_in_MWh.pkl', 'rb') as f:
    data = pickle.load(f)

# Load the data
print(len(data))
data.head(100)
```

8712

Out[268…

	Date	Load
23	2016-04-13 00:00:00	682.14
0	2016-04-13 01:00:00	717.49
1	2016-04-13 02:00:00	707.64
2	2016-04-13 03:00:00	706.44
3	2016-04-13 04:00:00	737.34
...	...	...
94	2016-04-16 23:00:00	771.89
119	2016-04-17 00:00:00	764.24
96	2016-04-17 01:00:00	722.89
97	2016-04-17 02:00:00	596.24
98	2016-04-17 03:00:00	595.24

100 rows × 2 columns

Preprocess data

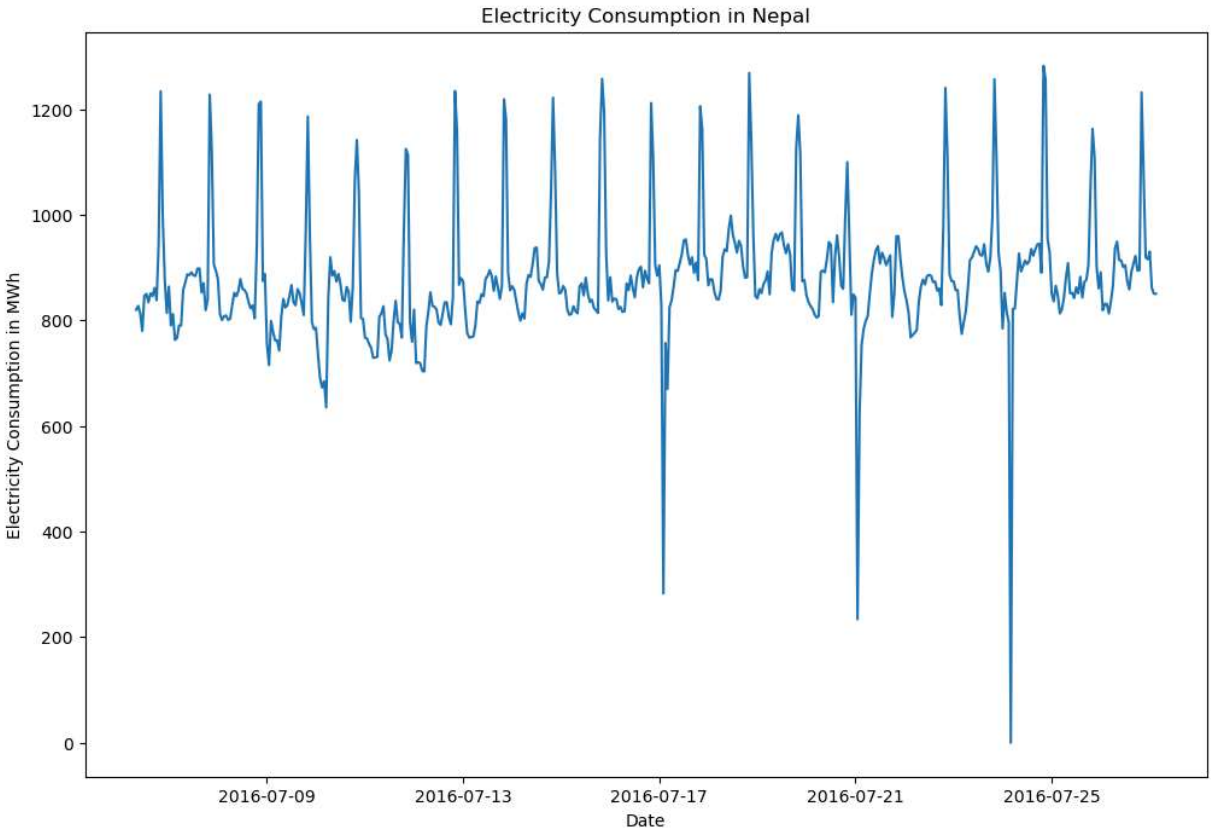
In [269…

```
# sort by date
data = data.sort_values('Date')

# convert date to datetime
data['Date'] = pd.to_datetime(data['Date'])

# see missing values
data.isnull().sum()

# fill missing values with the previous value
data = data.fillna(method='ffill')
```

Plot data

In [270…

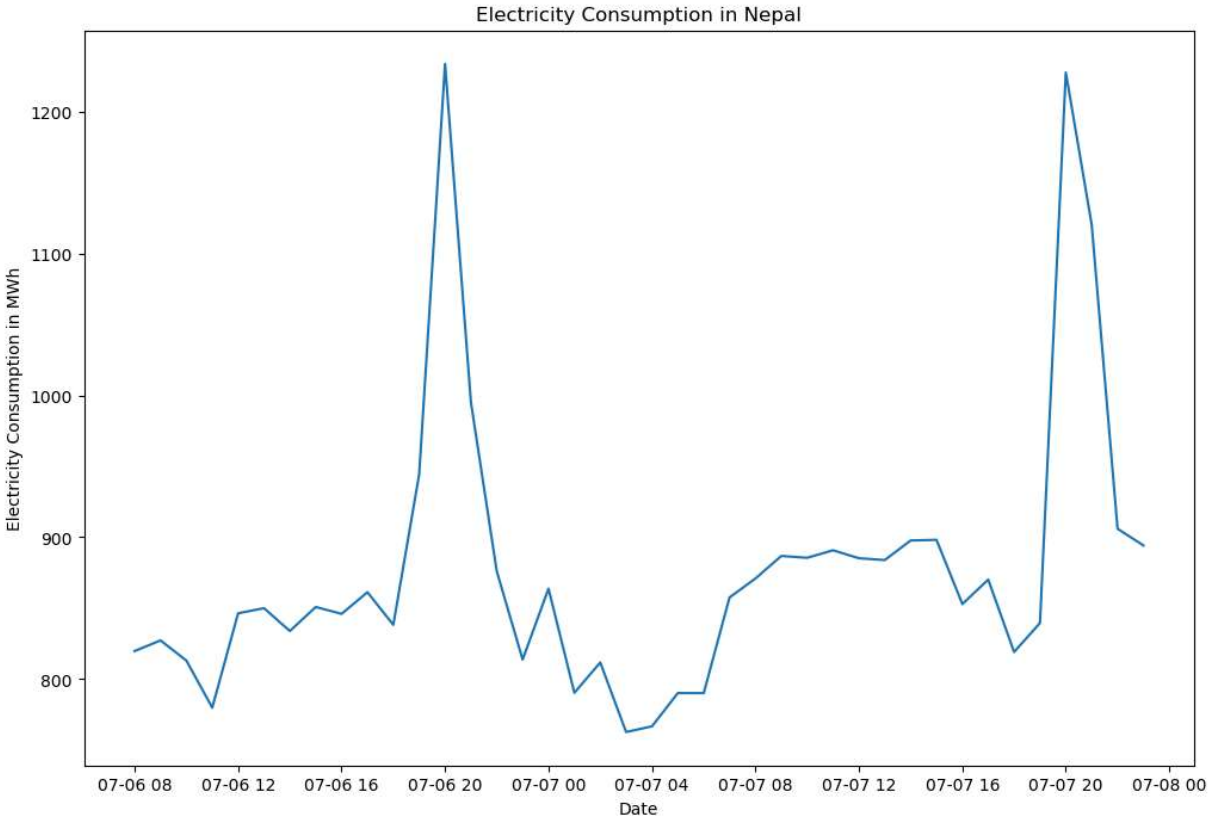
```
# plot the data
date = data['Date']
X = data['Load']

plt.figure(figsize=(12, 8))
plt.plot(date[2000:2500], X[2000:2500])
plt.xlabel('Date')
plt.ylabel('Electricity Consumption in MWh')
plt.title('Electricity Consumption in Nepal')
plt.show()
```



Take closer look to peaks

In [271…

```
plt.figure(figsize=(12, 8))
plt.plot(date[2000:2040], X[2000:2040])
plt.xlabel('Date')
plt.ylabel('Electricity Consumption in MWh')
plt.title('Electricity Consumption in Nepal')
plt.show()
```

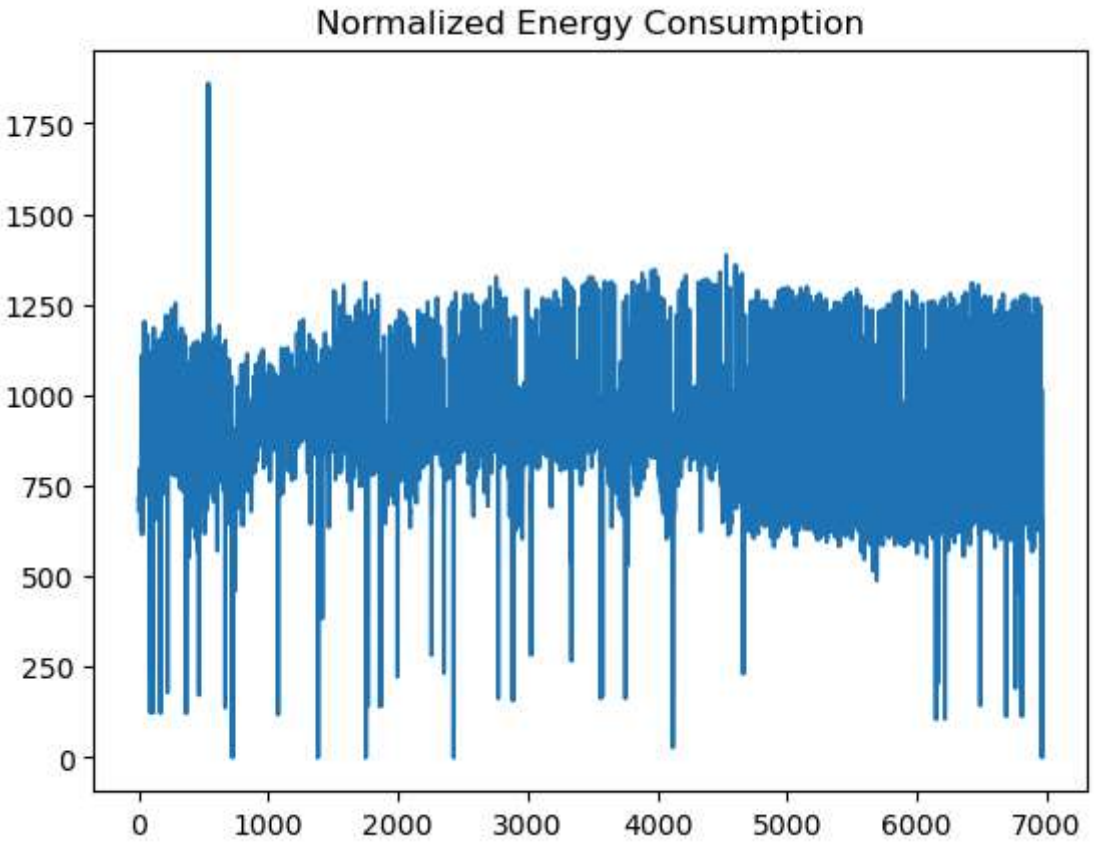


Data appears to have high peaks during evening, approximately during hours 20-22. Use 30 as number of lags to capture daily patterns

In [276…

```
z_score=3 #equal to 3-sigma i.e. ~99% of the data
len_segment=len(data) #length of the segment for analysis
#https://en.wikipedia.org/wiki/Partial_autocorrelation_function
reference=z_score/np.sqrt(len_segment)

print(len_segment)
nlag= 48 #number of lags for partial autocorrelation analysis
y = data['Load'].values

segment=y[0:int(0.8*len_segment)]
test_segment = y[int(0.8*len_segment):]

print(len(segment), len(test_segment))
y=y-np.mean(y)          #standardixe close price, zero mean, min-max -1...1
y=y/np.abs(np.max(y))

plt.plot(segment)
plt.title('Normalized Energy Consumption')
plt.show()
```

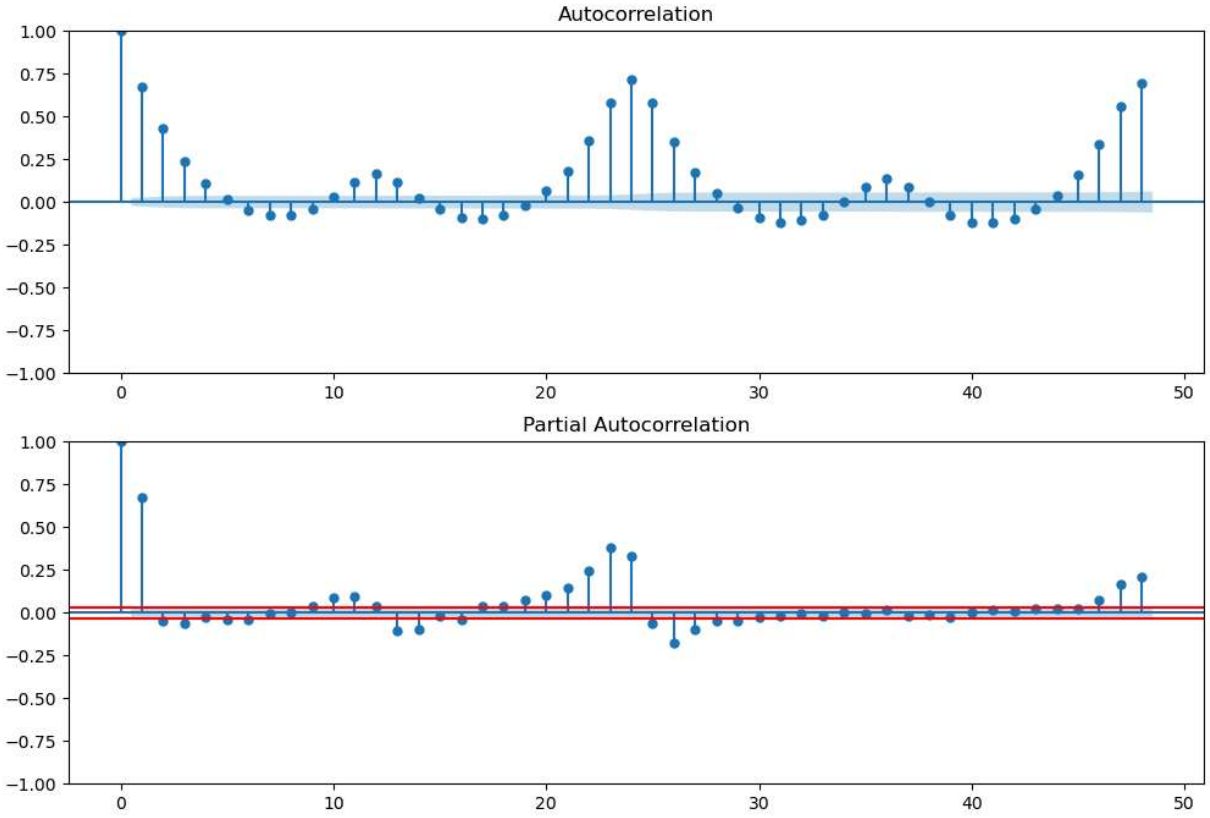
```
8712
6969 1743
```



Define parameters for ARIMA model

In [277…

```
fig = plt.figure(figsize=(12, 8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(segment.squeeze(), lags=nlag, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(segment, lags=nlag, ax=ax2)
ax2.axhline(y=reference, color='r')
ax2.axhline(y=-reference, color='r')
plt.show()
```



nlag=30 analysis: Select 28 as order of p and q, since it is the last statistically significant order in the plots. It keeps two peaks inside.

nlag = 48 analysis: Select, 48 as order of p and q. It holds three peaks inside and should increase prediction for longer seasonal trends.

p-value seems to be below the 0.05 limit, select p = 0.

In [278…
```
print("\nLet us find the d (differencing) value, the first one that goes below 0.05
result = adfuller(segment)
print('p-value (0.05): ', result[1])
result = adfuller(np.diff(segment))
print('1st order diff p-value (0.05): ', result[1])
result = adfuller(np.diff(np.diff(segment)))
print('2nd order diff p-value (0.05): ', result[1])
result = adfuller(np.diff(np.diff(np.diff(segment))))
print('3rd order diff p-value (0.05): ', result[1])
```

```
Let us find the d (differencing) value, the first one that goes below 0.05 limit:
p-value (0.05):  1.045837912904957e-10
1st order diff p-value (0.05):  0.0
2nd order diff p-value (0.05):  0.0
3rd order diff p-value (0.05):  0.0
```

Initialize ARIMA model

In [279…
```
arma_model=ARIMA(segment,order=(48,0,48)).fit()
print("\n\nModel summary")
print(arma_model.summary())
```

```
print(arma_model.params)
print("\nPrint AIC, BIC, HQIC values for full autocorrelation")
print(arma_model.aic, arma_model.bic, arma_model.hqic)
```

Model summary

SARIMAX Results

Dep. Variable:	y	No. Observations:	6969
Model:	ARIMA(48, 0, 48)	Log Likelihood	-40696.220
Date:	Fri, 20 Dec 2024	AIC	81588.439
Time:	15:00:11	BIC	82259.663
Sample:	0	HQIC	81819.784
	- 6969		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	861.9449	624.397	1.380	0.167	-361.851	2085.741
ar.L1	-0.3514	0.188	-1.865	0.062	-0.721	0.018
ar.L2	0.0101	0.177	0.057	0.954	-0.337	0.357
ar.L3	0.0460	0.173	0.266	0.790	-0.292	0.384
ar.L4	0.0303	0.161	0.188	0.850	-0.285	0.345
ar.L5	0.0400	0.164	0.244	0.808	-0.282	0.362
ar.L6	0.0321	0.165	0.194	0.846	-0.292	0.356
ar.L7	0.0728	0.148	0.491	0.623	-0.218	0.364
ar.L8	0.0318	0.171	0.187	0.852	-0.302	0.366
ar.L9	0.0105	0.160	0.066	0.948	-0.303	0.325
ar.L10	-0.0380	0.161	-0.236	0.813	-0.354	0.278
ar.L11	-0.0926	0.170	-0.545	0.586	-0.426	0.241
ar.L12	-0.0972	0.151	-0.644	0.519	-0.393	0.199
ar.L13	-0.0503	0.156	-0.322	0.748	-0.357	0.256
ar.L14	-0.0509	0.165	-0.308	0.758	-0.375	0.273
ar.L15	0.0402	0.149	0.270	0.787	-0.252	0.332
ar.L16	0.0296	0.146	0.203	0.839	-0.256	0.315
ar.L17	0.0481	0.145	0.332	0.740	-0.236	0.332
ar.L18	0.0768	0.130	0.592	0.554	-0.177	0.331
ar.L19	0.0050	0.130	0.039	0.969	-0.250	0.260
ar.L20	0.0326	0.130	0.250	0.803	-0.223	0.288
ar.L21	0.0670	0.127	0.528	0.597	-0.182	0.316
ar.L22	-0.0963	0.146	-0.660	0.509	-0.382	0.190
ar.L23	-0.3116	0.147	-2.119	0.034	-0.600	-0.023
ar.L24	0.1606	0.152	1.056	0.291	-0.137	0.459
ar.L25	0.3944	0.174	2.262	0.024	0.053	0.736
ar.L26	0.0176	0.155	0.114	0.910	-0.287	0.322
ar.L27	-0.0831	0.151	-0.552	0.581	-0.378	0.212
ar.L28	-0.0191	0.137	-0.139	0.889	-0.287	0.249
ar.L29	-0.0174	0.140	-0.124	0.901	-0.291	0.256
ar.L30	-0.0706	0.144	-0.490	0.624	-0.353	0.212
ar.L31	-0.0398	0.127	-0.313	0.755	-0.289	0.209
ar.L32	-0.0480	0.143	-0.335	0.738	-0.329	0.233
ar.L33	-0.0283	0.136	-0.208	0.835	-0.295	0.239
ar.L34	0.0709	0.138	0.513	0.608	-0.200	0.342
ar.L35	0.0594	0.148	0.402	0.687	-0.230	0.349
ar.L36	0.1116	0.135	0.827	0.408	-0.153	0.376
ar.L37	0.0714	0.140	0.508	0.611	-0.204	0.347
ar.L38	0.0281	0.148	0.189	0.850	-0.262	0.318
ar.L39	-0.0328	0.135	-0.244	0.807	-0.297	0.231
ar.L40	-0.0228	0.132	-0.172	0.863	-0.282	0.237
ar.L41	-0.0685	0.128	-0.536	0.592	-0.319	0.182

ar.L42	-0.0527	0.116	-0.453	0.650	-0.280	0.175
ar.L43	-0.0317	0.123	-0.258	0.796	-0.272	0.209
ar.L44	-0.0185	0.121	-0.153	0.878	-0.255	0.218
ar.L45	-0.0461	0.115	-0.401	0.689	-0.272	0.179
ar.L46	0.0256	0.131	0.196	0.844	-0.230	0.282
ar.L47	0.4401	0.124	3.536	0.000	0.196	0.684
ar.L48	0.6957	0.136	5.111	0.000	0.429	0.963
ma.L1	0.5932	0.188	3.147	0.002	0.224	0.963
ma.L2	0.2642	0.181	1.457	0.145	-0.091	0.620
ma.L3	0.2124	0.141	1.507	0.132	-0.064	0.489
ma.L4	0.1976	0.144	1.372	0.170	-0.085	0.480
ma.L5	0.1603	0.162	0.991	0.322	-0.157	0.477
ma.L6	0.1426	0.146	0.976	0.329	-0.144	0.429
ma.L7	0.1074	0.139	0.773	0.440	-0.165	0.380
ma.L8	0.1410	0.161	0.873	0.382	-0.175	0.457
ma.L9	0.1371	0.134	1.026	0.305	-0.125	0.399
ma.L10	0.1672	0.151	1.106	0.269	-0.129	0.464
ma.L11	0.2298	0.164	1.398	0.162	-0.092	0.552
ma.L12	0.2733	0.132	2.077	0.038	0.015	0.531
ma.L13	0.2405	0.140	1.723	0.085	-0.033	0.514
ma.L14	0.2173	0.168	1.295	0.195	-0.112	0.546
ma.L15	0.1794	0.142	1.267	0.205	-0.098	0.457
ma.L16	0.1534	0.127	1.209	0.227	-0.095	0.402
ma.L17	0.1311	0.141	0.932	0.351	-0.145	0.407
ma.L18	0.0960	0.121	0.791	0.429	-0.142	0.334
ma.L19	0.1396	0.119	1.173	0.241	-0.094	0.373
ma.L20	0.1213	0.131	0.926	0.355	-0.136	0.378
ma.L21	0.0888	0.117	0.759	0.448	-0.140	0.318
ma.L22	0.2442	0.125	1.954	0.051	-0.001	0.489
ma.L23	0.4795	0.149	3.225	0.001	0.188	0.771
ma.L24	0.1850	0.158	1.174	0.240	-0.124	0.494
ma.L25	-0.1106	0.122	-0.909	0.363	-0.349	0.128
ma.L26	0.1036	0.115	0.899	0.369	-0.122	0.329
ma.L27	0.2155	0.125	1.724	0.085	-0.030	0.461
ma.L28	0.1698	0.108	1.569	0.117	-0.042	0.382
ma.L29	0.1576	0.106	1.481	0.139	-0.051	0.366
ma.L30	0.2281	0.115	1.984	0.047	0.003	0.453
ma.L31	0.1963	0.098	2.012	0.044	0.005	0.387
ma.L32	0.2230	0.113	1.976	0.048	0.002	0.444
ma.L33	0.2077	0.117	1.774	0.076	-0.022	0.437
ma.L34	0.1259	0.106	1.187	0.235	-0.082	0.334
ma.L35	0.1174	0.104	1.125	0.261	-0.087	0.322
ma.L36	0.0628	0.113	0.555	0.579	-0.159	0.284
ma.L37	0.0505	0.115	0.439	0.660	-0.175	0.276
ma.L38	0.1082	0.108	0.998	0.318	-0.104	0.321
ma.L39	0.1581	0.106	1.490	0.136	-0.050	0.366
ma.L40	0.1349	0.112	1.207	0.227	-0.084	0.354
ma.L41	0.1980	0.099	1.997	0.046	0.004	0.392
ma.L42	0.1936	0.097	2.005	0.045	0.004	0.383
ma.L43	0.1898	0.103	1.843	0.065	-0.012	0.392
ma.L44	0.1811	0.090	2.013	0.044	0.005	0.357
ma.L45	0.1890	0.096	1.978	0.048	0.002	0.376
ma.L46	0.1377	0.115	1.195	0.232	-0.088	0.364
ma.L47	-0.2401	0.087	-2.771	0.006	-0.410	-0.070
ma.L48	-0.4897	0.097	-5.057	0.000	-0.679	-0.300
sigma2	6568.1486	37.954	173.055	0.000	6493.760	6642.537

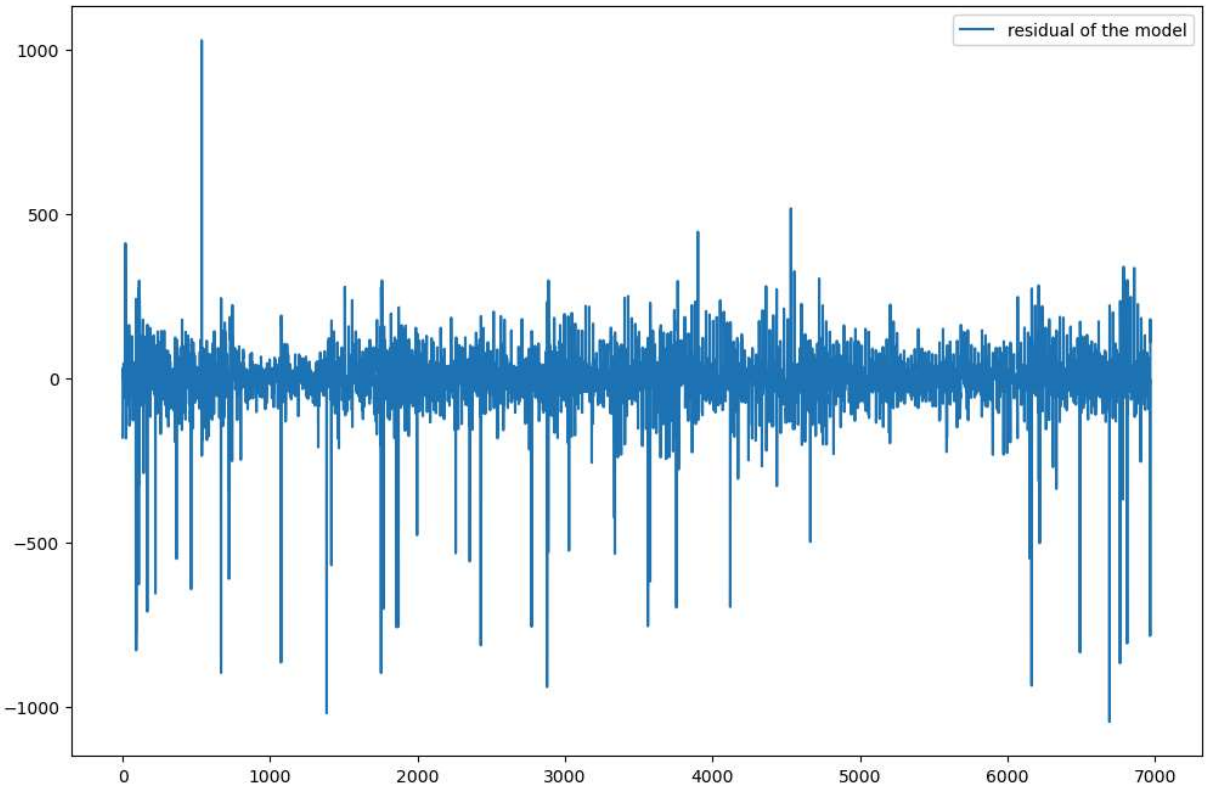
```
==============================================================================
Ljung-Box (L1) (Q):                  16.31   Jarque-Bera (JB):            524562.52
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               0.64   Skew:                            -3.65
Prob(H) (two-sided):                  0.00   Kurtosis:                        44.87
==============================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-ste
p).
[ 8.61944862e+02 -3.51397177e-01  1.01497831e-02  4.59504946e-02
  3.02844404e-02  4.00365891e-02  3.20982403e-02  7.28317142e-02
  3.18442340e-02  1.05392077e-02 -3.80472087e-02 -9.26062255e-02
 -9.72493931e-02 -5.03006230e-02 -5.08963978e-02  4.01517998e-02
  2.96317161e-02  4.80902214e-02  7.67727310e-02  5.01983898e-03
  3.25556286e-02  6.70432154e-02 -9.63051897e-02 -3.11601612e-01
  1.60603909e-01  3.94358573e-01  1.76396902e-02 -8.30799729e-02
 -1.90518464e-02 -1.73564032e-02 -7.05730529e-02 -3.97503597e-02
 -4.79839991e-02 -2.83206070e-02  7.08984747e-02  5.93958484e-02
  1.11583327e-01  7.13760635e-02  2.80511044e-02 -3.28422249e-02
 -2.28303710e-02 -6.85305402e-02 -5.26682079e-02 -3.16784477e-02
 -1.85063342e-02 -4.61080303e-02  2.56440948e-02  4.40108363e-01
  6.95731936e-01  5.93154159e-01  2.64238051e-01  2.12377770e-01
  1.97646124e-01  1.60268664e-01  1.42620927e-01  1.07365584e-01
  1.41020589e-01  1.37133579e-01  1.67210856e-01  2.29822034e-01
  2.73347555e-01  2.40542157e-01  2.17341564e-01  1.79443588e-01
  1.53444456e-01  1.31121793e-01  9.60472736e-02  1.39642994e-01
  1.21281862e-01  8.87543432e-02  2.44180420e-01  4.79524232e-01
  1.85013186e-01 -1.10636323e-01  1.03600288e-01  2.15525354e-01
  1.69753981e-01  1.57626545e-01  2.28107763e-01  1.96310247e-01
  2.22962491e-01  2.07700354e-01  1.25888836e-01  1.17397502e-01
  6.27609588e-02  5.04742325e-02  1.08226624e-01  1.58120035e-01
  1.34900162e-01  1.97952354e-01  1.93617990e-01  1.89847905e-01
  1.81102102e-01  1.88989108e-01  1.37658535e-01 -2.40052335e-01
 -4.89658101e-01  6.56814863e+03]

Print AIC, BIC, HQIC values for full autocorrelation
81588.43900782688 82259.66325592785 81819.78370997388
```

Show residual values

In [280…

```
print("\nPrint residual values")
print(sm.stats.durbin_watson(arma_model.resid))

fig = plt.figure(figsize=(12, 8))
plt.plot(arma_model.resid,label='residual of the model')
plt.legend()
plt.show()

print("\nResidual normalization test for both")
print("Full")
print(stats.normaltest(arma_model.resid))
```

```
Print residual values
1.9022854622464798
```




```
Residual normalization test for both
Full
NormaltestResult(statistic=5964.1018811476815, pvalue=0.0)
```

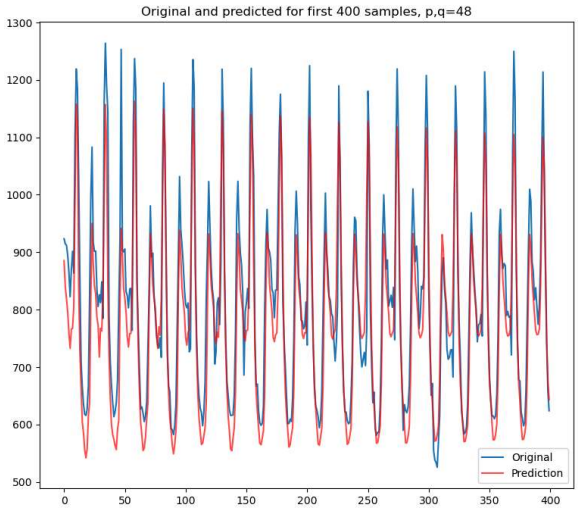
Residual values for both nlag = 30 and 48 are quite the same, majority of values remains near zero excluding trend peaks.

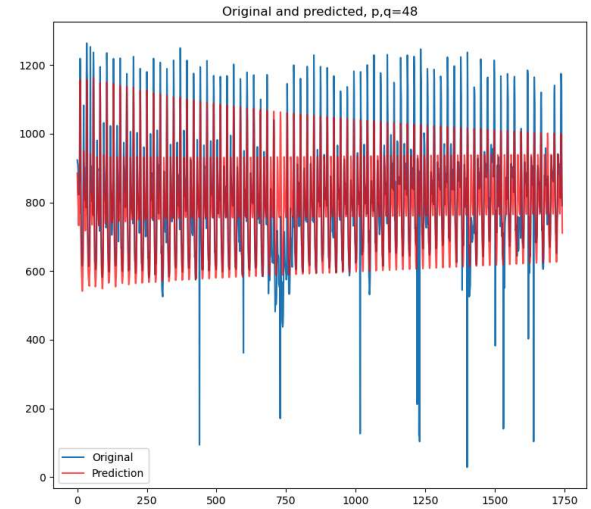
Make predictions

In [291…

```
#Organize data to neat datatable format
r, q, p = sm.tsa.acf(arma_model.resid.squeeze(), fft=True, qstat=True)

# Ensure all arrays have the same length
min_length = min(len(r[1:]), len(q), len(p))
r = r[1:min_length+1]
q = q[:min_length]
p = p[:min_length]

data = np.c_[np.arange(1, min_length+1), r, q, p]

table = pd.DataFrame(data, columns=["lag", "AC", "Q", "Prob(>Q)"])

#finally, predictions from model

predict_prices = arma_model.predict(start=len(segment), end=len(y), dynamic=True)

fig, ax = plt.subplots(1, 2, figsize=(20, 8))
ax[0].plot(test_segment[:400], label='Original')
ax[0].plot(predict_prices[:400], 'r', label='Prediction', alpha=0.7)
ax[0].set_title("Original and predicted for first 400 samples, p,q=48")
ax[0].legend()
```

```
ax[1].plot(test_segment, label='Original')
ax[1].plot(predict_prices, 'r', label='Prediction', alpha=0.7)
ax[1].set_title("Original and predicted, p,q=48")
ax[1].legend()
plt.show()
```

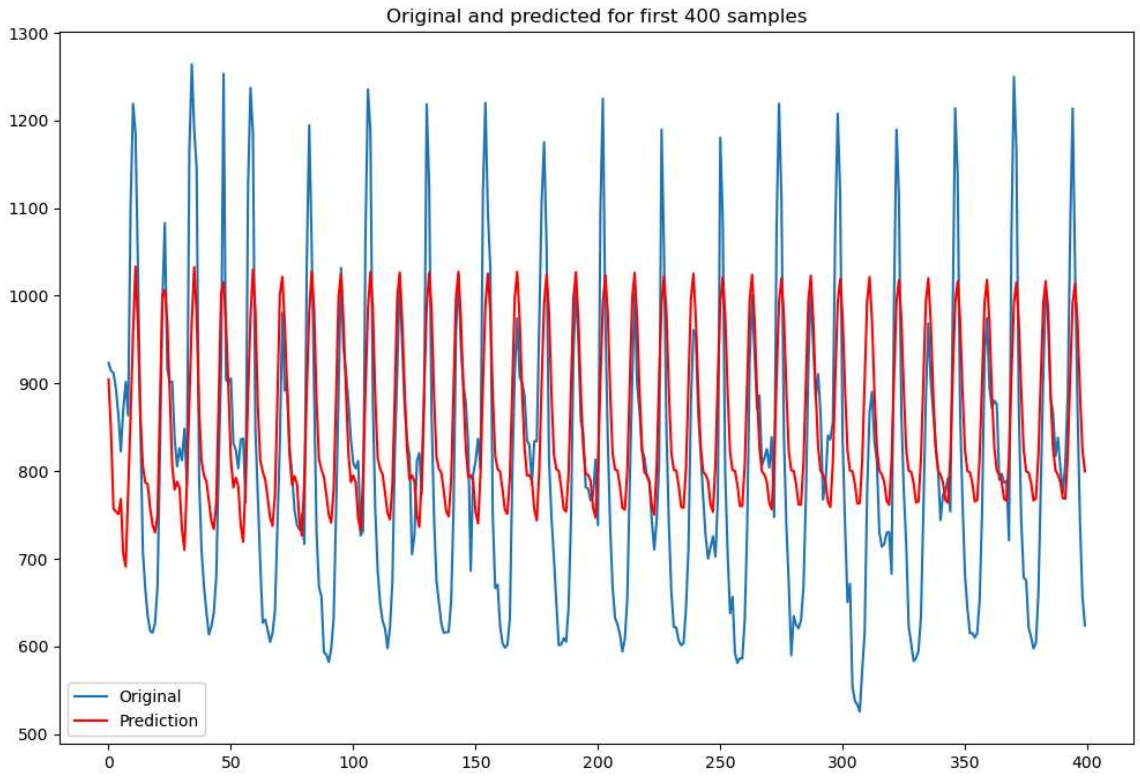




It seems that increased value for p and q have made the model work better and now it recognizes higher peaks between smaller peaks. However, looking at the total predicted, the peaks have decreasing trend and downside peaks tends to drop slightly under the orignal. It seems that prediction window may be too large and model could work well predicting small window into.

Prediction using p, q = 28:

ARIMA is able to recognize smaller peaks well but predictions for peaks are in the same level. Model fails to predict consumption drops. Another test is made after this with increased lag window and p and q values of ARIMA.


Original and predicted for first 400 samples
1300
1200
1100
1000
900
800
700
600
500
0
50
100
150
200
250
300
350
400
Original
Prediction