Question 1

Take the solar radiation data from File a, and copy it into File c, and run the power spectrum tool to find out which frequencies are important.

In [34]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

solar_radia2017 = pd.read_excel('HalfHourSolarRadiation2017.xlsx',sheet_name='Sheet1')
```

Discrete Fourier transform, using the power value as a measure of power in solar radiation

In [35]:
```
import math

def calculate_dft(data, num_frequencies):
    num_data = len(data)
    n2 = num_data / 2
    ret = {'frequency': [], 'ai': [], 'bi': [], 'power': []}

    for i in range(1,num_frequencies+1):
        s1 = 0
        s2 = 0
        s0 = sum(data)

        for j in range(1,num_data):
            angle = math.pi * i / n2 * (j + 1)
            cit = math.cos(angle)
            sit = math.sin(angle)
            s1 += cit * data[j]
            s2 += sit * data[j]

        s1 /= n2
        s2 /= n2

        ret['frequency'].append(i)
        ret['ai'].append(s1)
        ret['bi'].append(s2)
        ret['power'].append(s1 ** 2 + s2 ** 2)

    ret['ai'][0] = sum(data) / len(data)
    return pd.DataFrame(ret)
```

Based on the GHI column, calculate the power corresponding to different frequencies

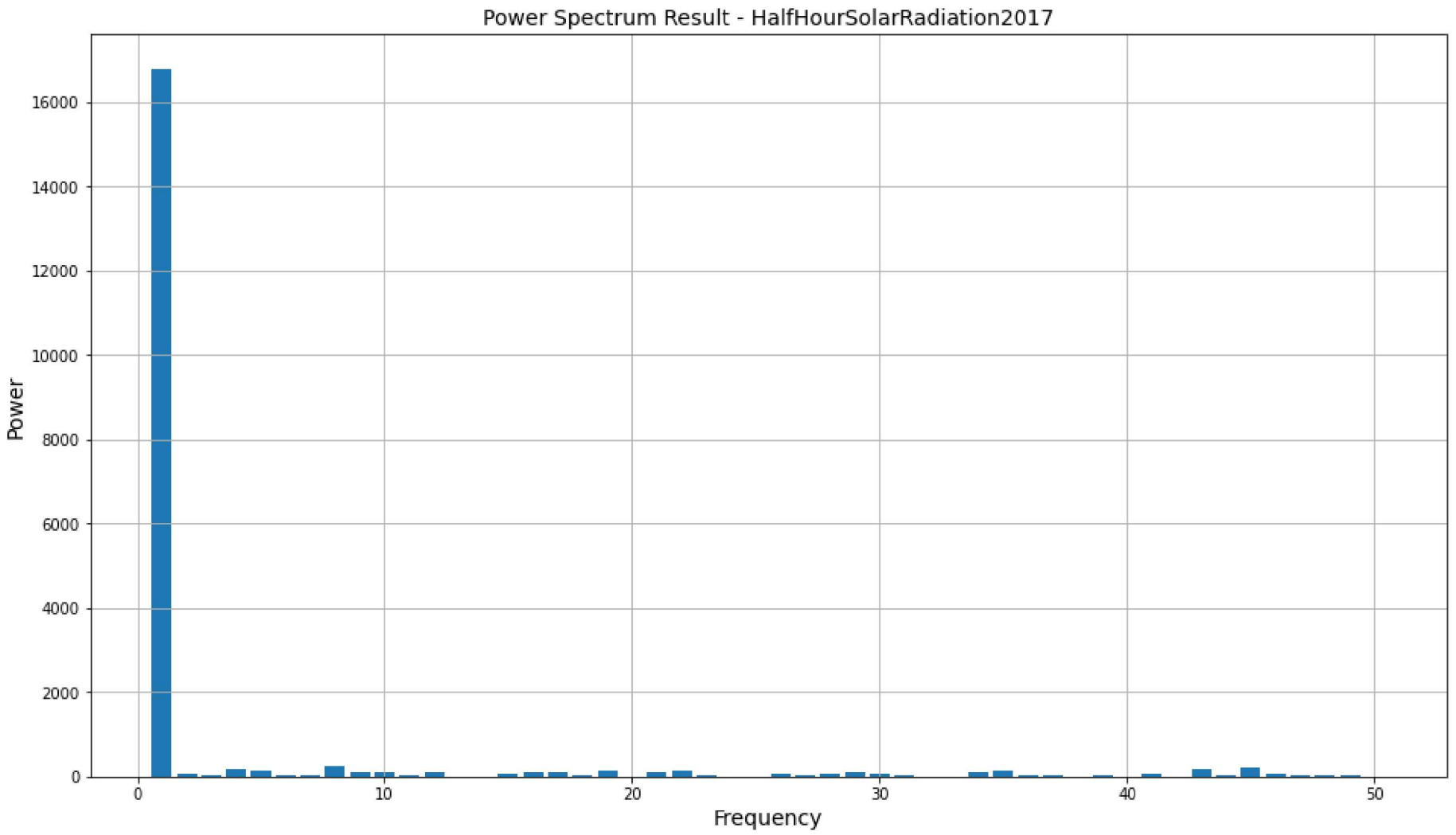
In [36]:
```
num_frequencies = 50

PS_result2017 = calculate_dft(solar_radia2017.GHI, num_frequencies)
PS_result2017
```

Out[36]:

	frequency	ai	bi	power
0	1	171.037166	14.970197	16785.192278
1	2	4.495565	5.587824	51.433878
2	3	1.914398	-3.030740	12.850306
3	4	7.799192	-9.905761	158.951499
4	5	8.008929	9.426698	153.005577
5	6	-6.382498	1.815368	44.031845
6	7	4.971753	-4.086769	41.420008
7	8	-0.540952	15.997302	256.206287
8	9	-9.827451	-0.908250	97.403710
9	10	2.797006	-9.896678	105.767476
10	11	5.743525	3.677180	46.509732
11	12	-10.574423	-2.082418	116.154878
12	13	-2.415825	-2.184167	10.606797
13	14	0.318859	-3.137737	9.947063
14	15	3.425715	-8.385859	82.058148
15	16	7.144505	7.100577	101.462139
16	17	7.758114	4.839201	83.606203
17	18	-2.206242	5.864729	39.262556
18	19	-4.101640	-10.779385	133.018597
19	20	-1.909186	-0.611762	4.019242
20	21	-3.396156	9.305048	98.117798
21	22	5.623170	10.070559	133.036195
22	23	-3.808198	-3.227466	24.918909
23	24	2.540435	-0.618453	6.836293
24	25	-0.751105	1.434853	2.622961
25	26	5.298178	5.202104	55.132573
26	27	-2.424014	-3.913956	21.194899
27	28	-3.884094	6.445883	56.635594
28	29	1.205559	-9.112865	84.497683
29	30	6.936245	5.339586	76.622677
30	31	-6.076999	1.603282	39.500425
31	32	0.740051	2.003940	4.563451
32	33	-0.402213	-2.530884	6.567149
33	34	5.017358	7.943469	88.272587
34	35	-11.768945	-1.809064	141.780788
35	36	-5.015773	-3.811240	39.683526
36	37	2.323925	-4.061629	21.897456
37	38	0.254363	-0.311400	0.161670
38	39	4.669052	-2.032619	25.931592
39	40	-1.457419	-0.858597	2.861261
40	41	-3.628303	6.944399	61.389262
41	42	-1.202092	-1.022826	2.491199
42	43	11.249041	-6.374717	167.177935
43	44	-2.979716	3.967704	24.621381
44	45	-10.536356	-9.501684	201.296783
45	46	6.878397	-0.275344	47.388155
46	47	-0.170752	3.285477	10.823517
47	48	-1.823783	-5.575093	34.407839
48	49	4.516816	-0.718047	20.917220
49	50	2.224784	1.488719	7.165950

In [37]:
```
#find out which frequencies are important

PS_result2017.sort_values(by=['power'],ascending=False,inplace=True)
print(PS_result2017)

plt.figure(figsize=(16,9))
plt.bar(PS_result2017['frequency'], PS_result2017['power'])
plt.title('Power Spectrum Result - HalfHourSolarRadiation2017', fontsize=14)
plt.xlabel('Frequency', fontsize=14)
plt.ylabel('Power', fontsize=14)
plt.grid(True)
plt.show()

#we can identify 5 important frequencies from the below result and chart
important_frequencies = PS_result2017['frequency'][:5].values
print('we can identify that 5 most important frequencies are: ',important_frequencies)
```

```
    frequency          ai         bi         power
0           1  171.037166  14.970197  16785.192278
7           8   -0.540952  15.997302    256.206287
44         45  -10.536356  -9.501684    201.296783
42         43   11.249041  -6.374717    167.177935
3           4    7.799192  -9.905761    158.951499
4           5    8.008929   9.426698    153.005577
34         35  -11.768945  -1.809064    141.780788
21         22    5.623170  10.070559    133.036195
18         19   -4.101640 -10.779385    133.018597
11         12  -10.574423  -2.082418    116.154878
9          10    2.797006  -9.896678    105.767476
15         16    7.144505   7.100577    101.462139
20         21   -3.396156   9.305048     98.117798
8           9   -9.827451  -0.908250     97.403710
33         34    5.017358   7.943469     88.272587
28         29    1.205559  -9.112865     84.497683
16         17    7.758114   4.839201     83.606203
14         15    3.425715  -8.385859     82.058148
29         30    6.936245   5.339586     76.622677
40         41   -3.628303   6.944399     61.389262
27         28   -3.884094   6.445883     56.635594
25         26    5.298178   5.202104     55.132573
1           2    4.495565   5.587824     51.433878
45         46    6.878397  -0.275344     47.388155
10         11    5.743525   3.677180     46.509732
5           6   -6.382498   1.815368     44.031845
6           7    4.971753  -4.086769     41.420008
35         36   -5.015773  -3.811240     39.683526
30         31   -6.076999   1.603282     39.500425
17         18   -2.206242   5.864729     39.262556
47         48   -1.823783  -5.575093     34.407839
38         39    4.669052  -2.032619     25.931592
22         23   -3.808198  -3.227466     24.918909
43         44   -2.979716   3.967704     24.621381
36         37    2.323925  -4.061629     21.897456
26         27   -2.424014  -3.913956     21.194899
48         49    4.516816  -0.718047     20.917220
2           3    1.914398  -3.030740     12.850306
46         47   -0.170752   3.285477     10.823517
12         13   -2.415825  -2.184167     10.606797
13         14    0.318859  -3.137737      9.947063
49         50    2.224784   1.488719      7.165950
23         24    2.540435  -0.618453      6.836293
32         33   -0.402213  -2.530884      6.567149
31         32    0.740051   2.003940      4.563451
19         20   -1.909186  -0.611762      4.019242
39         40   -1.457419  -0.858597      2.861261
24         25   -0.751105   1.434853      2.622961
41         42   -1.202092  -1.022826      2.491199
37         38    0.254363  -0.311400      0.161670
```



we can identify that 5 most important frequencies are: [1 8 45 43 4]

Use File c to find the Fourier series model for the seasonality. Note that the Template is designed for hourly data. You will have to make some adjustment to use it for half hourly data plus change the relevant frequencies if necessary.

In [38]:
```
#Define time intervals for half-hourly data
time_intervals = np.arange(1,solar_radia2017.shape[0]+1)

# setting frequencies
frequencies = [1,364,365,366,730]
a_list = [-128.6899753,87.08261942,-263.346882,90.51225331,109.2062286]
b_list = [14.97013864,1.911366892,21.43218589,-17.0102398,-17.22103819]

def calculate_fourier_series(data, t, frequencies):
    fsm_result = pd.DataFrame(columns=['data','t'])
    fsm_result['data'] = data
    fsm_result['t'] = t

    for i in range(len(frequencies)):
        freq_results = []
        w = 2 * np.pi * frequencies[i] / len(data)
        for j in range(len(data)):
            freq_result = a_list[i]*np.cos(w * t[j]) + b_list[i]*np.sin(w * t[j])
            freq_results.append(freq_result)
        fsm_result[str(frequencies[i])] = freq_results

    fsm_result['model'] = np.mean(data)+fsm_result['1']+fsm_result['364']+fsm_result['365']+fsm_result['366']+fsm_result['730'
    return fsm_result
```

In [39]:
```
#Fourier series model结果
FS_2017 = calculate_fourier_series(solar_radia2017['GHI'],time_intervals,frequencies)
FS_2017
```

Out[39]:

	data	t	1	364	365	366	730	model
0	0.0	1	-128.684598	86.590490	-258.296451	87.507336	101.027984	59.181927
1	0.0	2	-128.679205	84.624869	-248.826497	82.996940	85.964849	47.118123
2	0.0	3	-128.673795	81.219205	-235.099052	77.058663	65.043352	30.585540
3	0.0	4	-128.668368	76.431452	-217.348997	69.794666	39.689258	10.935177
4	0.0	5	-128.662925	70.343081	-195.880040	61.329920	11.630406	-10.202392
...	...	...	...	...	...	...	...	...
17515	0.0	17516	-128.711318	74.524836	-238.781183	86.847153	69.516971	34.433625
17516	0.0	17517	-128.706007	79.760109	-251.502537	90.111545	89.397578	50.097854
17517	0.0	17518	-128.700680	83.638122	-259.920612	91.825656	103.185887	61.065539
17518	0.0	17519	-128.695336	86.092883	-263.891375	91.959996	109.942249	66.445583
17519	0.0	17520	-128.689975	87.082619	-263.346882	90.512253	109.206229	65.801410

17520 rows × 8 columns

Take the difference between the data and the Fourier model - the residuals - and take them to Minitab and find the best ARMA(p,q) model.y.

In [40]:
```
#Find the best ARMA(p,q) model based on AIC。
# Specifically, use the AIC criterion to estimate the parameters of the ARMA model, ie (p, q)
# and traverse the model parameters that minimize the AIC value

import statsmodels.api as sm
from statsmodels.tsa.arima_model import ARMA

# Calculate the residuals
FS_2017['residual'] = FS_2017['data'] - FS_2017['model']

# Use the AIC criterion to estimate the parameters of the ARMA model, that is, (p, q)
# and traverse the model parameters that minimize the AIC value.

AIC_value = []

for ari in range(0,5):
    for maj in range(0,5):
        try:
            arma_object = ARMA(FS_2017['residual'],order=(ari,maj)).fit()
            AIC_value.append([ari,maj,arma_object.aic])
        except:
            print('wrong parameters')

print(AIC_value)
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
参数错误
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误


```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
参数错误
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
参数错误
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
参数错误
[[0, 0, 211388.87805141232], [0, 1, 197840.21810162504], [0, 2, 191895.1496084988], [0, 3, 188499.027497044], [0, 4, 187138.550
12778676], [1, 0, 184403.83699114464], [2, 0, 184402.88350117736], [3, 0, 184348.08741192293], [4, 0, 184318.50442182604]]
```

According to the results, when (p, q) = (4, 0), the AIC value is the smallest. Therefore, (4,0) is the best model parameter for ARMA.

In [41]:
```
# get the best model
best_arma = ARMA(FS_2017['residual'].tolist(),order=(4,0)).fit()
print(best_arma.summary())
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                17520
Model:                     ARMA(4, 0)   Log Likelihood             -92153.252
Method:                       css-mle   S.D. of innovations             46.568
Date:                Mon, 04 Sep 2023   AIC                         184318.504
Time:                        01:34:25   BIC                         184365.131
Sample:                             0   HQIC                        184333.858
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0366      2.846     -0.013      0.990      -5.616       5.542
ar.L1.y        0.8732      0.008    115.688      0.000       0.858       0.888
ar.L2.y        0.0654      0.010      6.527      0.000       0.046       0.085
ar.L3.y       -0.0197      0.010     -1.968      0.049      -0.039   -8.44e-05
ar.L4.y       -0.0424      0.008     -5.622      0.000      -0.057      -0.028
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.1699           -0.0000j            1.1699           -0.0000
AR.2            2.2343           -0.0000j            2.2343           -0.0000
AR.3           -1.9345           -2.2963j            3.0026           -0.3614
AR.4           -1.9345           +2.2963j            3.0026            0.3614
-----------------------------------------------------------------------------
```

Use the ARMA model to forecast one step ahead for the residuals and add that to the Fourier series model to get the full one step ahead forecast.

In [42]:
```
#forecast one step ahead for the residuals
FS_2017['pred_residuals'] = best_arma.predict(0,FS_2017.shape[0]-1)

#add that to the Fourier series model
FS_2017['pred_GHI'] = FS_2017['model'] + FS_2017['pred_residuals']
FS_2017
```

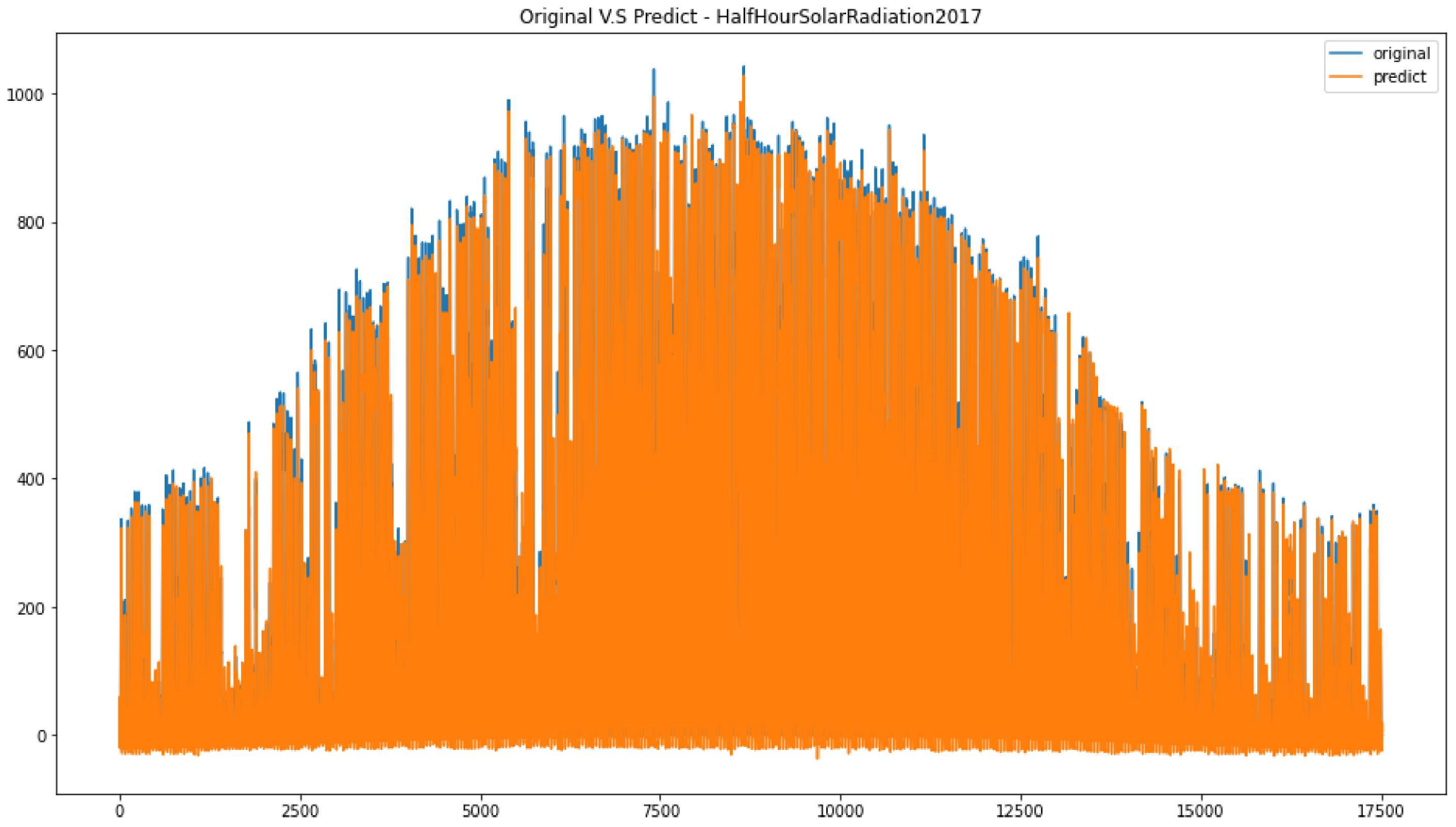
Out[42]:

	data	t	1	364	365	366	730	model	residual	pred_residuals	pred_GI
0	0.0	1	-128.684598	86.590490	-258.296451	87.507336	101.027984	59.181927	-59.181927	-0.036634	59.1452
1	0.0	2	-128.679205	84.624869	-248.826497	82.996940	85.964849	47.118123	-47.118123	-52.461725	-5.3436
2	0.0	3	-128.673795	81.219205	-235.099052	77.058663	65.043352	30.585540	-30.585540	-41.994793	-11.4092
3	0.0	4	-128.668368	76.431452	-217.348997	69.794666	39.689258	10.935177	-10.935177	-26.375489	-15.4403
4	0.0	5	-128.662925	70.343081	-195.880040	61.329920	11.630406	-10.202392	10.202392	-8.112879	-18.3152
...	...	...	...	...	...	...	...	...	...	...	...
17515	0.0	17516	-128.711318	74.524836	-238.781183	86.847153	69.516971	34.433625	-34.433625	-15.512995	18.9206
17516	0.0	17517	-128.706007	79.760109	-251.502537	90.111545	89.397578	50.097854	-50.097854	-32.311371	17.7864
17517	0.0	17518	-128.700680	83.638122	-259.920612	91.825656	103.185887	61.065539	-61.065539	-45.934854	15.1306
17518	0.0	17519	-128.695336	86.092883	-263.891375	91.959996	109.942249	66.445583	-66.445583	-55.272748	11.1728
17519	0.0	17520	-128.689975	87.082619	-263.346882	90.512253	109.206229	65.801410	-65.801410	-59.569405	6.2320

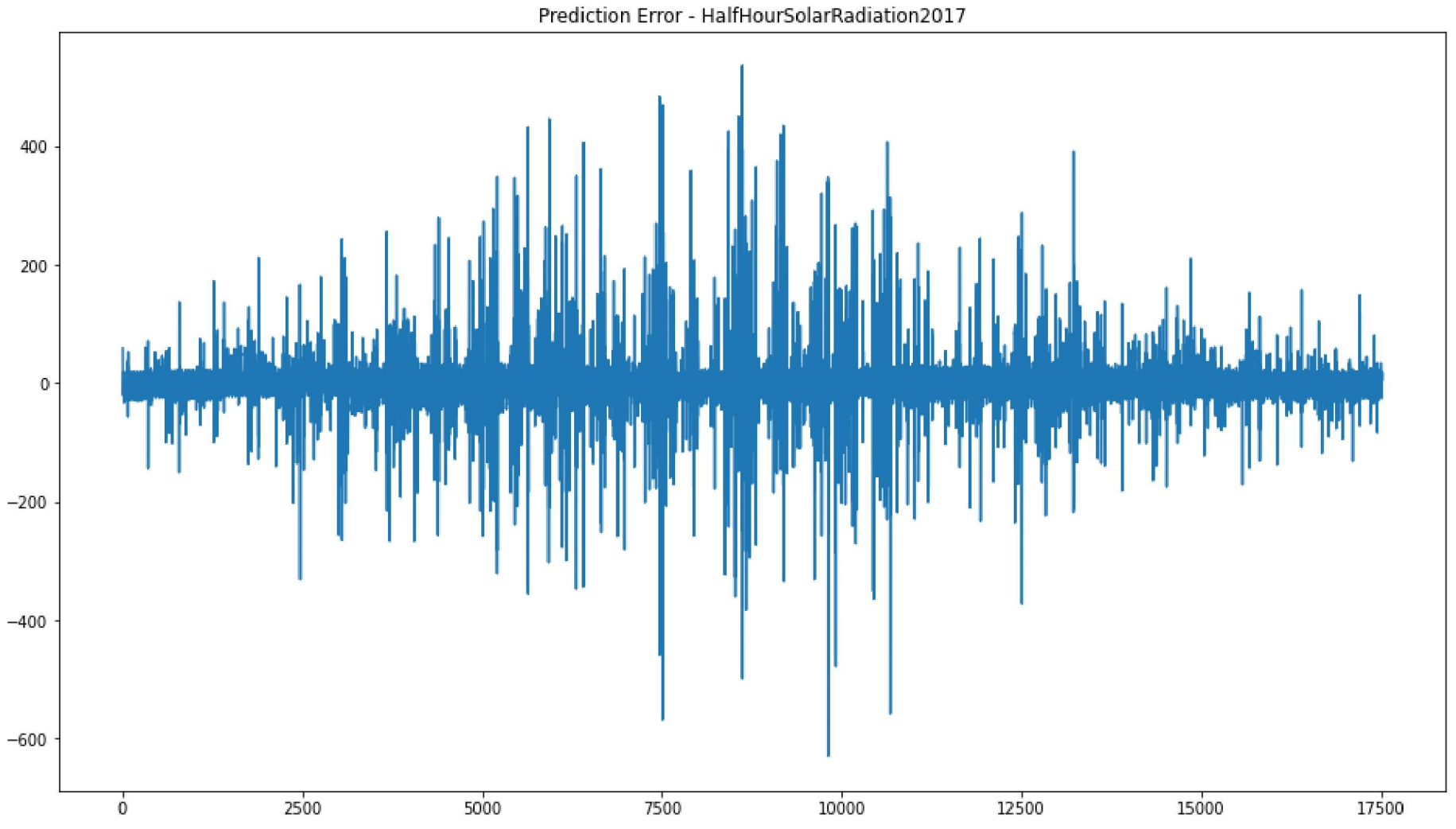
17520 rows × 11 columns

In [47]:
```
plt.figure(figsize=(16,9))
plt.plot(FS_2017['t'],FS_2017['data'],label='original')
plt.plot(FS_2017['t'],FS_2017['pred_GHI'],label='predict')
plt.title('Original V.S Predict - HalfHourSolarRadiation2017')
```

```
plt.legend()
plt.show()
```



In [48]:
```
plt.figure(figsize=(16,9))
plt.plot(FS_2017['t'],FS_2017['pred_GHI']-FS_2017['data'])
plt.title('Prediction Error - HalfHourSolarRadiation2017')
plt.show()
```



Use the error metrics defined below to evaluate the model.

In [49]:
```
def NMBE(true_value,model_value):
    diffs = []
    for i in range(len(true_value)):
        diffs.append(true_value[i]-model_value[i])
    NMBE_value = np.sum(diffs)/(len(true_value)*np.mean(true_value))
    return NMBE_value


def NMAE(true_value,model_value):
    diffs = []
    for i in range(len(true_value)):
        diffs.append(abs(true_value[i]-model_value[i]))
    NMAE_value = np.sum(diffs)/(len(true_value)*np.mean(true_value))
    return NMAE_value
```

In [50]:
```
#for solar radiation, we only do the calculation for solar elevation greater than or equal to 10 degrees
FS_2017['Elevation'] = solar_radia2017['Elevation']
solar_radia_pred_df = FS_2017[FS_2017['Elevation']>=10].reset_index(drop=True)

#calculate NMBE
NMBE_value = NMBE(solar_radia_pred_df['data'],solar_radia_pred_df['pred_GHI'])
print('NMBE: ',NMBE_value)

#calculate NMAE
NMAE_value = NMAE(solar_radia_pred_df['data'],solar_radia_pred_df['pred_GHI'])
print('NMAE: ',NMAE_value)
```

```
NMBE:  0.0005764058614954701
NMAE:  0.10672634392416983
```

Use the models you have developed for 2017 to see how they perform for the 2018 data, the out of sample data. Comment on the differences in the error metrics

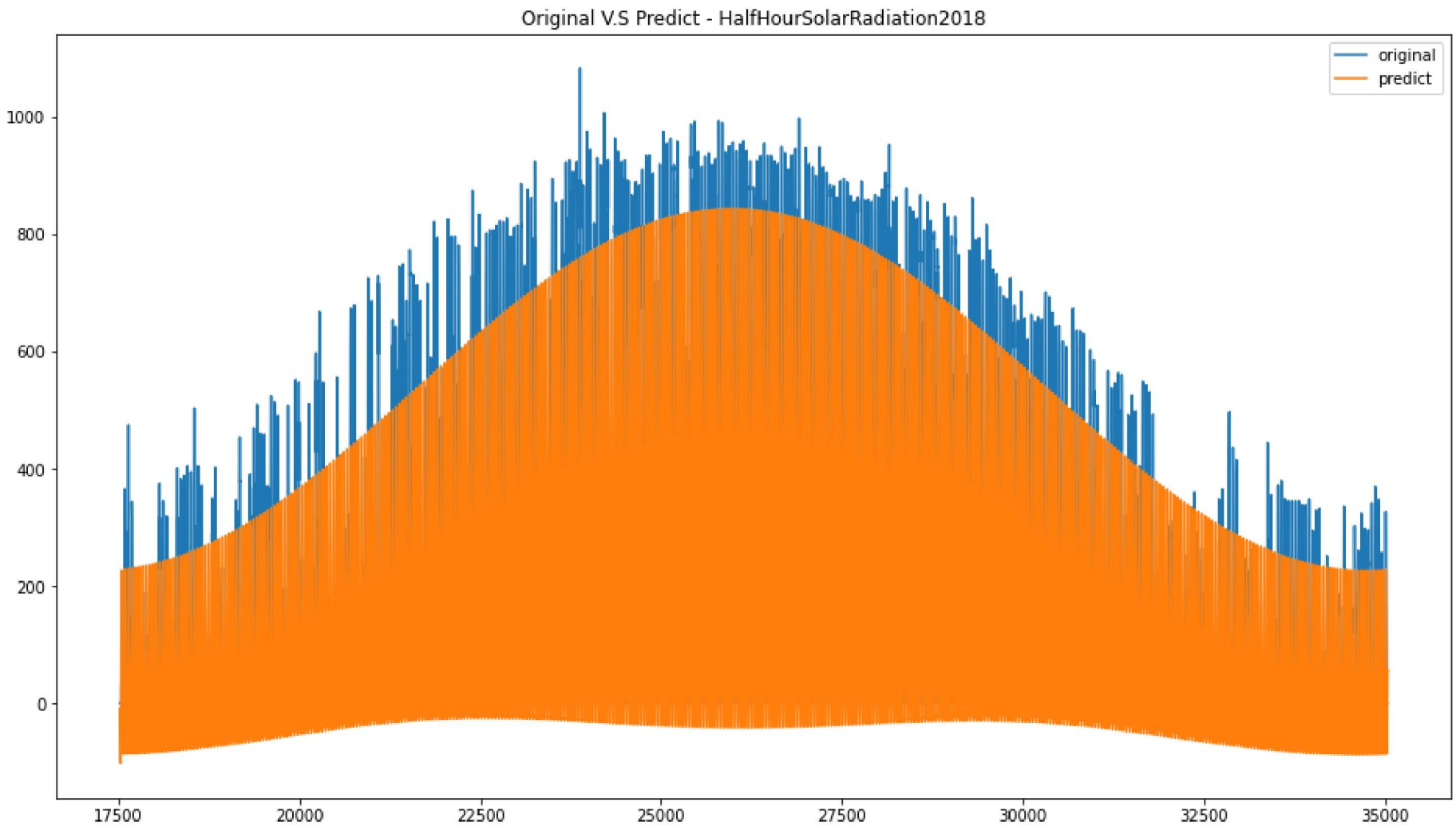
In [51]:
```
solar_radia2018 = pd.read_excel('HalfHourSolarRadiation2018.xlsx',sheet_name='Sheet1')

FS_2018 = calculate_fourier_series(solar_radia2018['Solar'],FS_2017.shape[0]+time_intervals,frequencies)
FS_2018['pred_Solar'] = FS_2018['model'] + best_arma.predict(FS_2018['t'][0],FS_2017.shape[0]+solar_radia2018.shape[0]-1)
FS_2018
```

Out[51]:

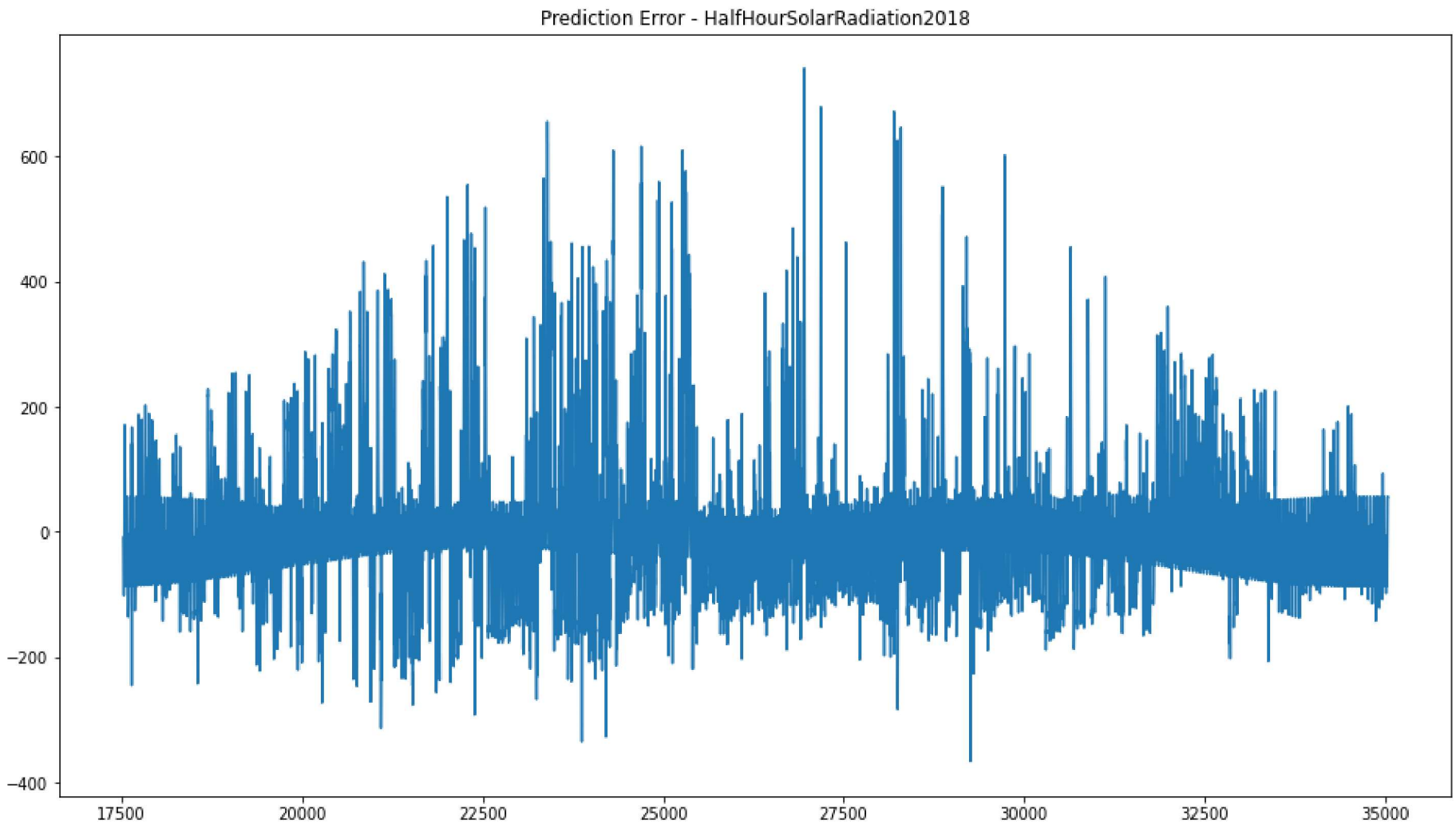
	data	t	1	364	365	366	730	model	pred_Solar
0	0.0	17521	-128.684598	86.590490	-258.296451	87.507336	101.027984	49.067076	-9.410699
1	0.0	17522	-128.679205	84.624869	-248.826497	82.996940	85.964849	37.003272	-14.465873
2	0.0	17523	-128.673795	81.219205	-235.099052	77.058663	65.043352	20.470689	-24.183895
3	0.0	17524	-128.668368	76.431452	-217.348997	69.794666	39.689258	0.820326	-37.597180
4	0.0	17525	-128.662925	70.343081	-195.880040	61.329920	11.630406	-20.317243	-53.291848
...	...	...	...	...	...	...	...	...	...
17515	0.0	35036	-128.711318	74.524836	-238.781183	86.847153	69.516971	24.318774	24.282140
17516	0.0	35037	-128.706007	79.760109	-251.502537	90.111545	89.397578	39.983003	39.946370
17517	0.0	35038	-128.700680	83.638122	-259.920612	91.825656	103.185887	50.950688	50.914054
17518	0.0	35039	-128.695336	86.092883	-263.891375	91.959996	109.942249	56.330733	56.294099
17519	0.0	35040	-128.689975	87.082619	-263.346882	90.512253	109.206229	55.686559	55.649926

17520 rows × 9 columns

In [52]:
```
plt.figure(figsize=(16,9))
plt.plot(FS_2018['t'],FS_2018['data'],label='original')
plt.plot(FS_2018['t'],FS_2018['pred_Solar'],label='predict')
plt.title('Original V.S Predict - HalfHourSolarRadiation2018')
plt.legend()
plt.show()
```



In [53]:
```
plt.figure(figsize=(16,9))
plt.plot(FS_2018['t'],FS_2018['pred_Solar']-FS_2018['data'])
plt.title('Prediction Error - HalfHourSolarRadiation2018')
plt.show()
```



In [35]:
```
FS_2018['Elevation'] = solar_radia2018['Elevation']

solar_radia_pred_df2018 = FS_2018[FS_2018['Elevation']>=10].reset_index(drop=True)

#calculate NMBE
NMBE_value = NMBE(solar_radia_pred_df2018['data'],solar_radia_pred_df2018['pred_Solar'])
print('NMBE: ',NMBE_value)

#calculate NMAE
NMAE_value = NMAE(solar_radia_pred_df2018['data'],solar_radia_pred_df2018['pred_Solar'])
print('NMAE: ',NMAE_value)
```

```
NMBE:  -0.03731455570097423
NMAE:  0.3165742182418225
```

The forecast data results for 2018 are relatively accurate.

But compared with the error metrics of the ARMA model on the 2017 data (NMBE: 0.0005764058614954701; NMAE: 0.10672634392416983), its performance on the 2018 data is worse.

Question 2

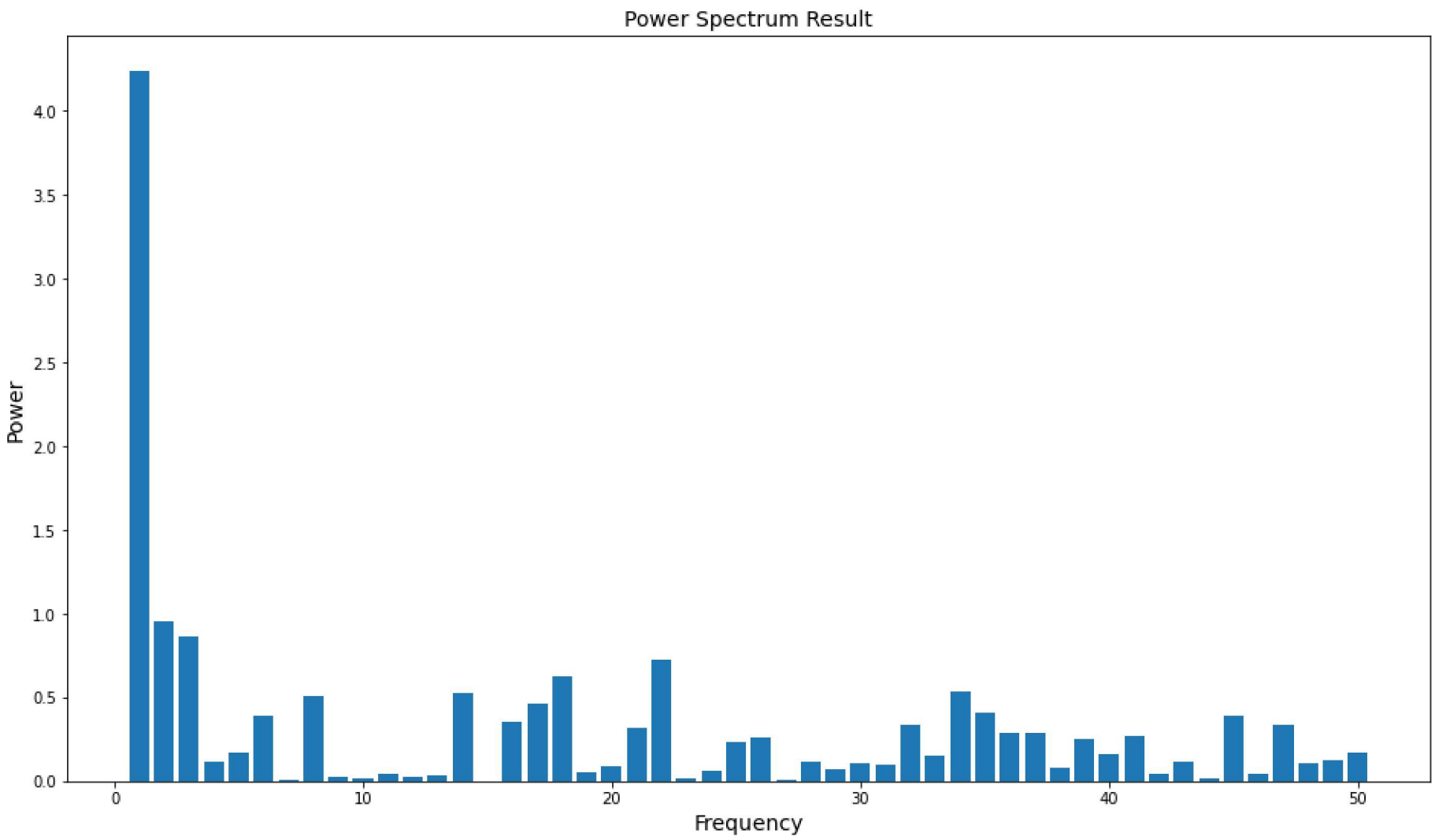
In [8]:
```
#use the power spectrum to decide on the necessary frequencies
solar_farm = pd.read_excel('SolarFarm.xlsx',sheet_name='Sheet1')

num_frequencies = 50
PS_result_SolarFarm = calculate_dft(solar_farm.Output, num_frequencies)
PS_result_SolarFarm
```

Out[8]:

	frequency	ai	bi	power
0	1	14.789658	0.146161	4.238366
1	2	-0.654833	0.725198	0.954719
2	3	0.188253	-0.908364	0.860564
3	4	0.295985	-0.158863	0.112845
4	5	0.376397	-0.178873	0.173670
5	6	0.609140	-0.127232	0.387239
6	7	-0.085552	0.011051	0.007441
7	8	0.471443	0.537855	0.511546
8	9	-0.070038	0.140644	0.024686
9	10	0.092156	0.076483	0.014342
10	11	-0.041695	-0.209945	0.045815
11	12	-0.034183	0.148884	0.023335
12	13	-0.076381	-0.160276	0.031522
13	14	-0.449367	-0.567574	0.524071
14	15	0.023546	-0.006046	0.000591
15	16	0.330498	-0.494301	0.353563
16	17	0.109381	-0.669173	0.459757
17	18	-0.731622	-0.298799	0.624552
18	19	0.156278	0.165745	0.051894
19	20	0.246235	0.161421	0.086689
20	21	-0.513910	0.220348	0.312657
21	22	0.105736	0.845238	0.725607
22	23	0.116900	0.009992	0.013765
23	24	0.244219	0.074966	0.065263
24	25	-0.271806	0.398671	0.232817
25	26	0.417810	0.290567	0.258994
26	27	-0.088538	-0.016542	0.008113
27	28	-0.024912	-0.340409	0.116499
28	29	0.094252	0.252475	0.072627
29	30	0.303112	0.138229	0.110984
30	31	0.302653	-0.066050	0.095961
31	32	-0.432586	0.380604	0.331990
32	33	-0.183501	0.340598	0.149680
33	34	0.221506	0.700209	0.539357
34	35	-0.298174	-0.564355	0.407404
35	36	-0.417969	-0.344050	0.293069
36	37	-0.425556	-0.332561	0.291695
37	38	0.177228	-0.227421	0.083130
38	39	-0.498321	-0.070083	0.253236
39	40	0.397390	0.054352	0.160873
40	41	0.385312	0.345111	0.267567
41	42	0.152556	0.150711	0.045987
42	43	0.321197	0.124025	0.118550
43	44	-0.113881	-0.024682	0.013578
44	45	-0.569391	0.251811	0.387615
45	46	-0.110359	0.170574	0.041274
46	47	0.188940	0.543942	0.331571
47	48	-0.309960	0.104127	0.106918
48	49	0.169284	-0.317936	0.129740
49	50	0.098403	-0.405488	0.174103

In [9]:
```
#find out which frequencies are important

PS_result_SolarFarm.sort_values(by=['power'],ascending=False,inplace=True)
print(PS_result_SolarFarm)

plt.figure(figsize=(16,9))
plt.bar(PS_result_SolarFarm['frequency'], PS_result_SolarFarm['power'])
plt.title('Power Spectrum Result', fontsize=14)
plt.xlabel('Frequency', fontsize=14)
plt.ylabel('Power', fontsize=14)
plt.show()

#we can identify important frequencies from the below result and chart
important_frequencies = PS_result_SolarFarm['frequency'][:5].values
print('we can identify that the important frequencies for solar farm output are: ',important_frequencies)
```

```
    frequency         ai        bi     power
0           1  14.789658  0.146161  4.238366
1           2  -0.654833  0.725198  0.954719
2           3   0.188253 -0.908364  0.860564
21         22   0.105736  0.845238  0.725607
17         18  -0.731622 -0.298799  0.624552
33         34   0.221506  0.700209  0.539357
13         14  -0.449367 -0.567574  0.524071
7           8   0.471443  0.537855  0.511546
16         17   0.109381 -0.669173  0.459757
34         35  -0.298174 -0.564355  0.407404
44         45  -0.569391  0.251811  0.387615
5           6   0.609140 -0.127232  0.387239
15         16   0.330498 -0.494301  0.353563
31         32  -0.432586  0.380604  0.331990
46         47   0.188940  0.543942  0.331571
20         21  -0.513910  0.220348  0.312657
35         36  -0.417969 -0.344050  0.293069
36         37  -0.425556 -0.332561  0.291695
40         41   0.385312  0.345111  0.267567
25         26   0.417810  0.290567  0.258994
38         39  -0.498321 -0.070083  0.253236
24         25  -0.271806  0.398671  0.232817
49         50   0.098403 -0.405488  0.174103
4           5   0.376397 -0.178873  0.173670
39         40   0.397390  0.054352  0.160873
32         33  -0.183501  0.340598  0.149680
48         49   0.169284 -0.317936  0.129740
42         43   0.321197  0.124025  0.118550
27         28  -0.024912 -0.340409  0.116499
3           4   0.295985 -0.158863  0.112845
29         30   0.303112  0.138229  0.110984
47         48  -0.309960  0.104127  0.106918
30         31   0.302653 -0.066050  0.095961
19         20   0.246235  0.161421  0.086689
37         38   0.177228 -0.227421  0.083130
28         29   0.094252  0.252475  0.072627
23         24   0.244219  0.074966  0.065263
18         19   0.156278  0.165745  0.051894
41         42   0.152556  0.150711  0.045987
10         11  -0.041695 -0.209945  0.045815
45         46  -0.110359  0.170574  0.041274
12         13  -0.076381 -0.160276  0.031522
8           9  -0.070038  0.140644  0.024686
11         12  -0.034183  0.148884  0.023335
9          10   0.092156  0.076483  0.014342
22         23   0.116900  0.009992  0.013765
43         44  -0.113881 -0.024682  0.013578
26         27  -0.088538 -0.016542  0.008113
6           7  -0.085552  0.011051  0.007441
14         15   0.023546 -0.006046  0.000591
```



we can identify that the important frequencies for solar farm output are: [1 2 3 22 18]

In [10]:
```
#find the Fourier series model

time_intervals = np.arange(1,solar_farm.shape[0]+1)

FS_solar_farm = calculate_fourier_series(solar_farm['Output'],time_intervals,frequencies)
FS_solar_farm
```

Out[10]:

	data	t	1	364	365	366	730	model
0	0.0	1	-128.689080	87.103592	-262.816672	90.118503	108.351117	-91.142883
1	0.0	2	-128.688185	87.083334	-262.161375	89.681625	107.289754	-92.005190
2	0.0	3	-128.687289	87.021856	-261.381304	89.201830	106.024158	-93.031091
3	0.0	4	-128.686392	86.919187	-260.476830	88.679347	104.556739	-94.218292
4	0.0	5	-128.685496	86.775376	-259.448382	88.114425	102.890290	-95.564129
...	...	...	...	...	...	...	...	...
105115	0.0	105116	-128.693551	86.586925	-264.212706	91.652530	110.537543	-89.339602
105116	0.0	105117	-128.692658	86.772521	-264.184768	91.432963	110.519751	-89.362533
105117	0.0	105118	-128.691764	86.917045	-264.031093	91.169641	110.291578	-89.554935
105118	0.0	105119	-128.690870	87.020427	-263.751753	90.862689	109.853460	-89.916390
105119	0.0	105120	-128.689975	87.082619	-263.346882	90.512253	109.206229	-90.446098

105120 rows × 8 columns

Similarly, use the AIC criterion to estimate the parameters of the ARMA model, ie (p, q), and traverse the model parameters that minimize the AIC value

In [11]:
```
import statsmodels.api as sm
from statsmodels.tsa.arima_model import ARMA

# Calculate the residuals
FS_solar_farm['residual'] = FS_solar_farm['data'] - FS_solar_farm['model']

# Use the AIC criterion to estimate the parameters of the ARMA model, ie (p,q)

AIC_value = []

for ari in range(0,5):
    for maj in range(0,5):
        try:
            arma_object = ARMA(FS_solar_farm['residual'],order=(ari,maj)).fit()
            AIC_value.append([ari,maj,arma_object.aic])
        except:
            print('Wrong Parameters')

print(AIC_value)
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/base/tsa_model.py:7: FutureWarnin
g: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate d
type instead.
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/base/tsa_model.py:7: FutureWarnin
g: pandas.Float64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
```

According to the results, it can be seen that when (p, q) = (2, 1), the AIC value is the smallest. Therefore, (2,1) is the optimal model parameter of ARMA.

In [12]:
```
# get the best armr model
#best_arma = ARMA(FS_solar_farm['residual'].tolist(),order=(2,1)).fit()

#forecast one step ahead for the residuals
FS_solar_farm['pred_residuals'] = best_arma.predict(0,FS_solar_farm.shape[0]-1)

#add that to the Fourier series model
FS_solar_farm['pred_Output'] = FS_solar_farm['model'] + FS_solar_farm['pred_residuals']
FS_solar_farm
```

Out[12]:

	data	t	1	364	365	366	730	model	residual	pred_residuals	pred_O
0	0.0	1	-128.689080	87.103592	-262.816672	90.118503	108.351117	-91.142883	91.142883	5.315180	-85.82
1	0.0	2	-128.688185	87.083334	-262.161375	89.681625	107.289754	-92.005190	92.005190	5.314167	-86.69
2	0.0	3	-128.687289	87.021856	-261.381304	89.201830	106.024158	-93.031091	93.031091	5.311093	-87.71
3	0.0	4	-128.686392	86.919187	-260.476830	88.679347	104.556739	-94.218292	94.218292	5.305959	-88.91
4	0.0	5	-128.685496	86.775376	-259.448382	88.114425	102.890290	-95.564129	95.564129	5.298769	-90.26
...	...	...	...	...	...	...	...	...	...	...	...
105115	0.0	105116	-128.693551	86.586925	-264.212706	91.652530	110.537543	-89.339602	89.339602	5.289346	-84.05
105116	0.0	105117	-128.692658	86.772521	-264.184768	91.432963	110.519751	-89.362533	89.362533	5.298626	-84.06
105117	0.0	105118	-128.691764	86.917045	-264.031093	91.169641	110.291578	-89.554935	89.554935	5.305852	-84.24
105118	0.0	105119	-128.690870	87.020427	-263.751753	90.862689	109.853460	-89.916390	89.916390	5.311021	-84.60
105119	0.0	105120	-128.689975	87.082619	-263.346882	90.512253	109.206229	-90.446098	90.446098	5.314131	-85.13

105120 rows × 11 columns

In [13]:
```
solar_farm_pred = FS_solar_farm[FS_solar_farm['data']>0].reset_index(drop=True)

#calculate NMBE
NMBE_value = NMBE(solar_farm_pred['data'],solar_farm_pred['pred_Output'])
print('NMBE: ',NMBE_value)

#calculate NMAE
NMAE_value = NMAE(solar_farm_pred['data'],solar_farm_pred['pred_Output'])
print('NMAE: ',NMAE_value)
```

```
NMBE:  -4.589187327708524
NMAE:  7.175711416139246
```

Question 3

In [56]:
```
wind_farm = pd.read_excel('SnowtownWindFarm.xlsx',sheet_name='Sheet1')
wind_farm
```

Out[56]:

	Snowtown
0	84.198322
1	84.117132
2	81.609137
3	79.826108
4	75.148393
...	...
17515	38.392713
17516	46.934388
17517	53.936237
17518	58.116488
17519	58.070410

17520 rows × 1 columns

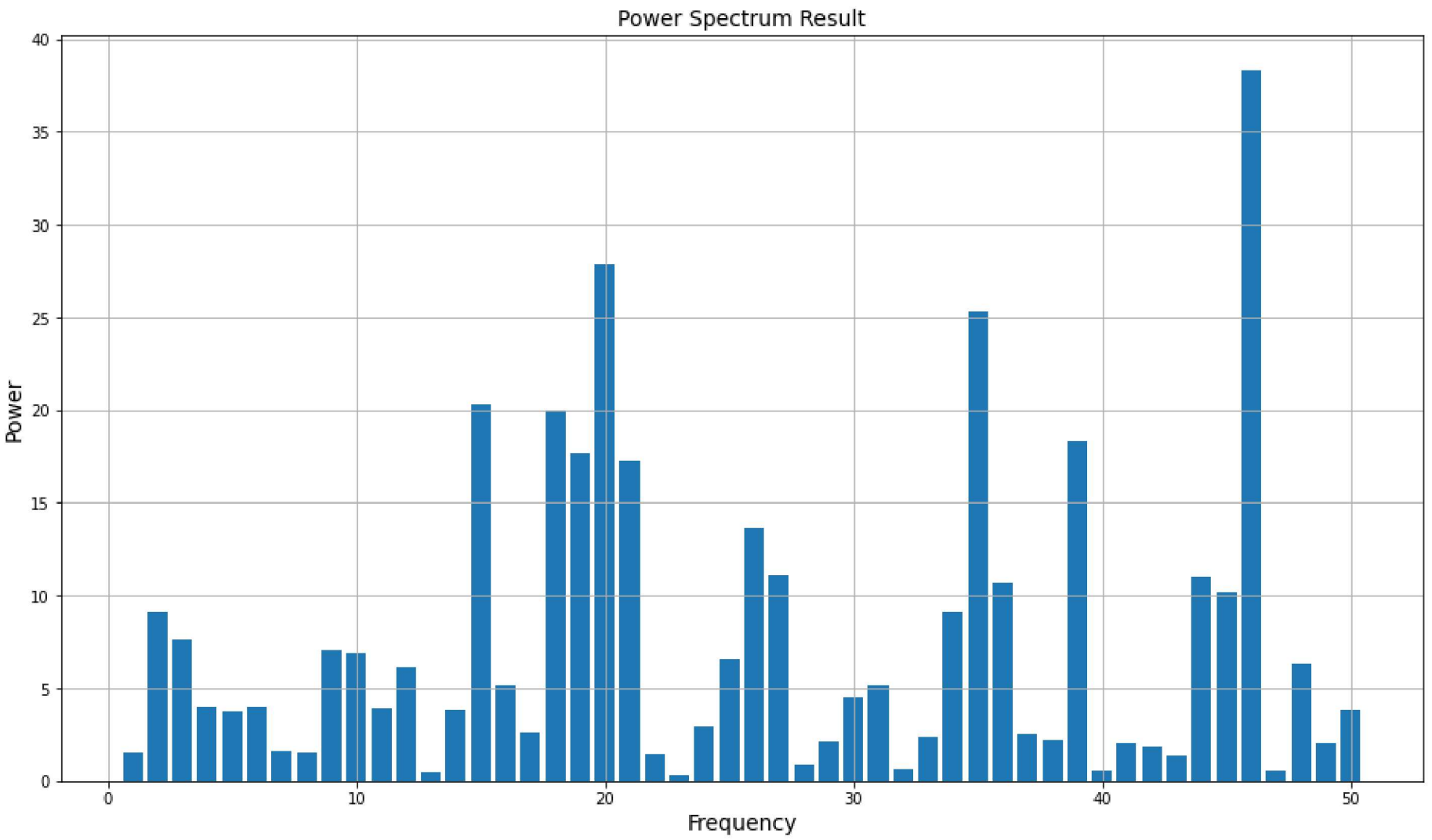
In [57]:
```
#Use the power spectrum file from above to show that there is no significant seasonality in the data

num_frequencies = 50
PS_result_windFarm = calculate_dft(wind_farm.Snowtown, num_frequencies)
PS_result_windFarm
```

Out[57]:

	frequency	ai	bi	power
0	1	36.605356	0.756815	1.520784
1	2	2.995149	0.352905	9.095458
2	3	1.909071	1.988578	7.598997
3	4	1.435534	1.411307	4.052545
4	5	-1.581604	1.127824	3.773458
5	6	1.341553	-1.480244	3.990889
6	7	0.758636	-1.026324	1.628870
7	8	-0.435229	1.177705	1.576414
8	9	-1.719059	-2.022673	7.046370
9	10	1.534883	2.130692	6.895716
10	11	-1.663518	-1.077311	3.927893
11	12	2.262251	1.021960	6.162180
12	13	0.616326	0.277349	0.456780
13	14	0.870656	-1.763459	3.867831
14	15	-3.419908	2.938028	20.327782
15	16	-1.360392	-1.828568	5.194326
16	17	-0.998102	-1.263728	2.593218
17	18	4.184620	-1.569606	19.974710
18	19	-4.195564	0.177137	17.634132
19	20	4.103955	-3.316293	27.840249
20	21	2.196765	3.527584	17.269620
21	22	-0.849624	-0.869396	1.477711
22	23	0.125770	0.508548	0.274440
23	24	0.215775	1.700361	2.937786
24	25	2.482703	0.624639	6.553988
25	26	-3.555153	-0.983289	13.605972
26	27	2.834060	1.757737	11.121537
27	28	-0.854909	-0.348624	0.852408
28	29	-1.434278	-0.178809	2.089127
29	30	1.534130	-1.454036	4.467775
30	31	-0.316538	2.251581	5.169813
31	32	0.540457	-0.608305	0.662129
32	33	0.018394	-1.540963	2.374907
33	34	3.010750	0.119057	9.078790
34	35	3.449838	3.658955	25.289334
35	36	-0.065630	-3.263489	10.654670
36	37	0.869409	1.318868	2.495286
37	38	1.467529	0.123240	2.168829
38	39	-1.027784	-4.157671	18.342569
39	40	0.362237	-0.624291	0.520955
40	41	-1.403006	-0.189705	2.004414
41	42	1.110790	-0.814845	1.897827
42	43	0.024746	1.183959	1.402371
43	44	-3.289056	0.461414	11.030790
44	45	2.455552	-2.048016	10.224104
45	46	-3.636081	5.010479	38.325991
46	47	-0.749762	0.063575	0.566185
47	48	2.209391	-1.196004	6.311831
48	49	-1.376302	-0.365793	2.028011
49	50	-0.494605	1.898355	3.848387

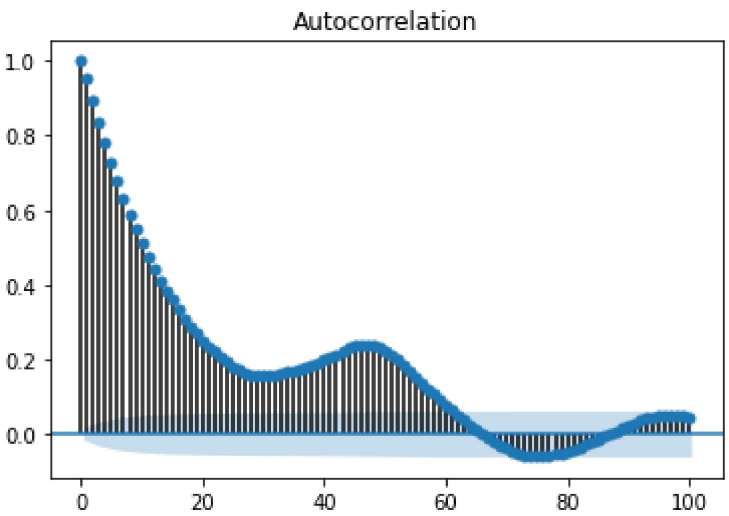
In [58]:
```
#find out which frequencies are important

PS_result_windFarm.sort_values(by=['power'],ascending=False,inplace=True)
print(PS_result_windFarm)

plt.figure(figsize=(16,9))
plt.bar(PS_result_windFarm['frequency'], PS_result_windFarm['power'])
plt.title('Power Spectrum Result', fontsize=14)
plt.xlabel('Frequency', fontsize=14)
plt.ylabel('Power', fontsize=14)
plt.grid(True)
plt.show()

#we can identify important frequencies from the below result and chart
important_frequencies = PS_result_windFarm['frequency'][:5].values
print('we can identify that 5 most important frequencies are: ',important_frequencies)
```

```
    frequency         ai        bi      power
45         46  -3.636081  5.010479  38.325991
19         20   4.103955 -3.316293  27.840249
34         35   3.449838  3.658955  25.289334
14         15  -3.419908  2.938028  20.327782
17         18   4.184620 -1.569606  19.974710
38         39  -1.027784 -4.157671  18.342569
18         19  -4.195564  0.177137  17.634132
20         21   2.196765  3.527584  17.269620
25         26  -3.555153 -0.983289  13.605972
26         27   2.834060  1.757737  11.121537
43         44  -3.289056  0.461414  11.030790
35         36  -0.065630 -3.263489  10.654670
44         45   2.455552 -2.048016  10.224104
1           2   2.995149  0.352905   9.095458
33         34   3.010750  0.119057   9.078790
2           3   1.909071  1.988578   7.598997
8           9  -1.719059 -2.022673   7.046370
9          10   1.534883  2.130692   6.895716
24         25   2.482703  0.624639   6.553988
47         48   2.209391 -1.196004   6.311831
11         12   2.262251  1.021960   6.162180
15         16  -1.360392 -1.828568   5.194326
30         31  -0.316538  2.251581   5.169813
29         30   1.534130 -1.454036   4.467775
3           4   1.435534  1.411307   4.052545
5           6   1.341553 -1.480244   3.990889
10         11  -1.663518 -1.077311   3.927893
13         14   0.870656 -1.763459   3.867831
49         50  -0.494605  1.898355   3.848387
4           5  -1.581604  1.127824   3.773458
23         24   0.215775  1.700361   2.937786
16         17  -0.998102 -1.263728   2.593218
36         37   0.869409  1.318868   2.495286
32         33   0.018394 -1.540963   2.374907
37         38   1.467529  0.123240   2.168829
28         29  -1.434278 -0.178809   2.089127
48         49  -1.376302 -0.365793   2.028011
40         41  -1.403006 -0.189705   2.004414
41         42   1.110790 -0.814845   1.897827
6           7   0.758636 -1.026324   1.628870
7           8  -0.435229  1.177705   1.576414
0           1  36.605356  0.756815   1.520784
21         22  -0.849624 -0.869396   1.477711
42         43   0.024746  1.183959   1.402371
27         28  -0.854909 -0.348624   0.852408
31         32   0.540457 -0.608305   0.662129
46         47  -0.749762  0.063575   0.566185
39         40   0.362237 -0.624291   0.520955
12         13   0.616326  0.277349   0.456780
22         23   0.125770  0.508548   0.274440
```



we can identify that 5 most important frequencies are: [46 20 35 15 18]

In [59]:
```
import statsmodels.api as sm
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Calculate the acf and pacf of the wind_farm data to determine the optimal order p of the AR model
acf_plot = plot_acf(wind_farm['Snowtown'], lags=100)

# According to the autoregressive graph, the lag value should be selected as [1,2,3,4,5,45]
lags = [1,2,3,4,5,45]
```



In [60]:
```
best_ar = AutoReg(wind_farm['Snowtown'], lags = lags, seasonal = False).fit()
print(best_ar.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:               Snowtown   No. Observations:                17520
Model:             Restr. AutoReg(45)   Log Likelihood              -63139.604
Method:                Conditional MLE   S.D. of innovations              8.973
Date:                Mon, 04 Sep 2023   AIC                              4.389
Time:                        01:44:03   BIC                              4.393
Sample:                            45   HQIC                             4.390
                                17520
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept        1.4934      0.127     11.793      0.000       1.245       1.742
Snowtown.L1      1.1481      0.008    151.708      0.000       1.133       1.163
Snowtown.L2     -0.2630      0.012    -22.838      0.000      -0.286      -0.240
Snowtown.L3      0.0893      0.012      7.654      0.000       0.066       0.112
Snowtown.L4     -0.0274      0.012     -2.383      0.017      -0.050      -0.005
Snowtown.L5     -0.0057      0.008     -0.760      0.447      -0.021       0.009
Snowtown.L45     0.0179      0.002      7.843      0.000       0.013       0.022
                                    Roots
==============================================================================
                  Real          Imaginary           Modulus         Frequency
------------------------------------------------------------------------------
AR.1            1.0201           -0.0000j            1.0201           -0.0000
AR.2            1.0305           -0.1168j            1.0371           -0.0180
AR.3            1.0305           +0.1168j            1.0371            0.0180
AR.4            1.0224           -0.2583j            1.0545           -0.0394
AR.5            1.0224           +0.2583j            1.0545            0.0394
AR.6            0.9869           -0.4017j            1.0655           -0.0615
AR.7            0.9869           +0.4017j            1.0655            0.0615
AR.8            0.9280           -0.5396j            1.0735           -0.0838
AR.9            0.9280           +0.5396j            1.0735            0.0838
AR.10           0.8481           -0.6682j            1.0797           -0.1062
AR.11           0.8481           +0.6682j            1.0797            0.1062
AR.12           0.7493           -0.7844j            1.0848           -0.1286
AR.13           0.7493           +0.7844j            1.0848            0.1286
AR.14           0.6339           -0.8853j            1.0889           -0.1511
AR.15           0.6339           +0.8853j            1.0889            0.1511
AR.16           0.5047           -0.9686j            1.0922           -0.1736
AR.17           0.5047           +0.9686j            1.0922            0.1736
AR.18           0.3645           -1.0326j            1.0951           -0.1960
AR.19           0.3645           +1.0326j            1.0951            0.1960
AR.20           0.2164           -1.0760j            1.0975           -0.2184
AR.21           0.2164           +1.0760j            1.0975            0.2184
AR.22           0.0634           -1.0980j            1.0998           -0.2408
AR.23           0.0634           +1.0980j            1.0998            0.2408
AR.24          -0.0914           -1.0983j            1.1021           -0.2632
AR.25          -0.0914           +1.0983j            1.1021            0.2632
AR.26          -0.2450           -1.0768j            1.1043           -0.2856
AR.27          -0.2450           +1.0768j            1.1043            0.2856
AR.28          -0.3946           -1.0339j            1.1066           -0.3080
AR.29          -0.3946           +1.0339j            1.1066            0.3080
AR.30          -0.5372           -0.9701j            1.1089           -0.3305
AR.31          -0.5372           +0.9701j            1.1089            0.3305
AR.32          -0.6698           -0.8865j            1.1111           -0.3530
AR.33          -0.6698           +0.8865j            1.1111            0.3530
AR.34          -0.7897           -0.7845j            1.1131           -0.3755
AR.35          -0.7897           +0.7845j            1.1131            0.3755
AR.36          -1.1157           -0.0796j            1.1185           -0.4887
AR.37          -1.1157           +0.0796j            1.1185            0.4887
AR.38          -1.0927           -0.2370j            1.1181           -0.4660
AR.39          -1.0927           +0.2370j            1.1181            0.4660
AR.40          -1.0474           -0.3894j            1.1174           -0.4434
AR.41          -1.0474           +0.3894j            1.1174            0.4434
AR.42          -0.9806           -0.5334j            1.1163           -0.4207
AR.43          -0.9806           +0.5334j            1.1163            0.4207
AR.44          -0.8941           -0.6660j            1.1148           -0.3981
AR.45          -0.8941           +0.6660j            1.1148            0.3981
------------------------------------------------------------------------------
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/ar_model.py:248: FutureWarning: T
he parameter names will change after 0.12 is released. Set old_names to False to use the new names now. Set old_names to True t
o use the old names.
  warnings.warn(
```

According to the results, it can be seen that the results are significantly related to lag1, lag2, lag3, and lag45.

Use the AIC criterion to estimate the parameters of the ARMA model, that is, (p, q), and traverse the model parameters that minimize the AIC value.

In [61]:
```
from statsmodels.tsa.arima_model import ARMA

AIC_value = []

for ari in range(0,5):
    for maj in range(0,5):
        try:
```

```
            arma_object = ARMA(wind_farm['Snowtown'],order=(ari,maj)).fit()
            AIC_value.append([ari,maj,arma_object.aic])
        except:
            print('Wrong Parameters')

print(AIC_value)
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

参数错误

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误


```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/deterministic.py:1451: FutureWarn
ing: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate
dtype instead.
  elif isinstance(self._index, pd.Int64Index):
```

参数错误

```
[[0, 0, 169528.62942035173], [0, 1, 151001.85716047406], [1, 0, 127402.38612839137], [2, 0, 126718.32153901298], [3, 0, 126671.
94064500957], [4, 0, 126654.26618513638]]
```

According to the results, when (p, q) = (4, 0), the AIC value is the smallest. Therefore, (4,0) is the best model parameter for ARMA.

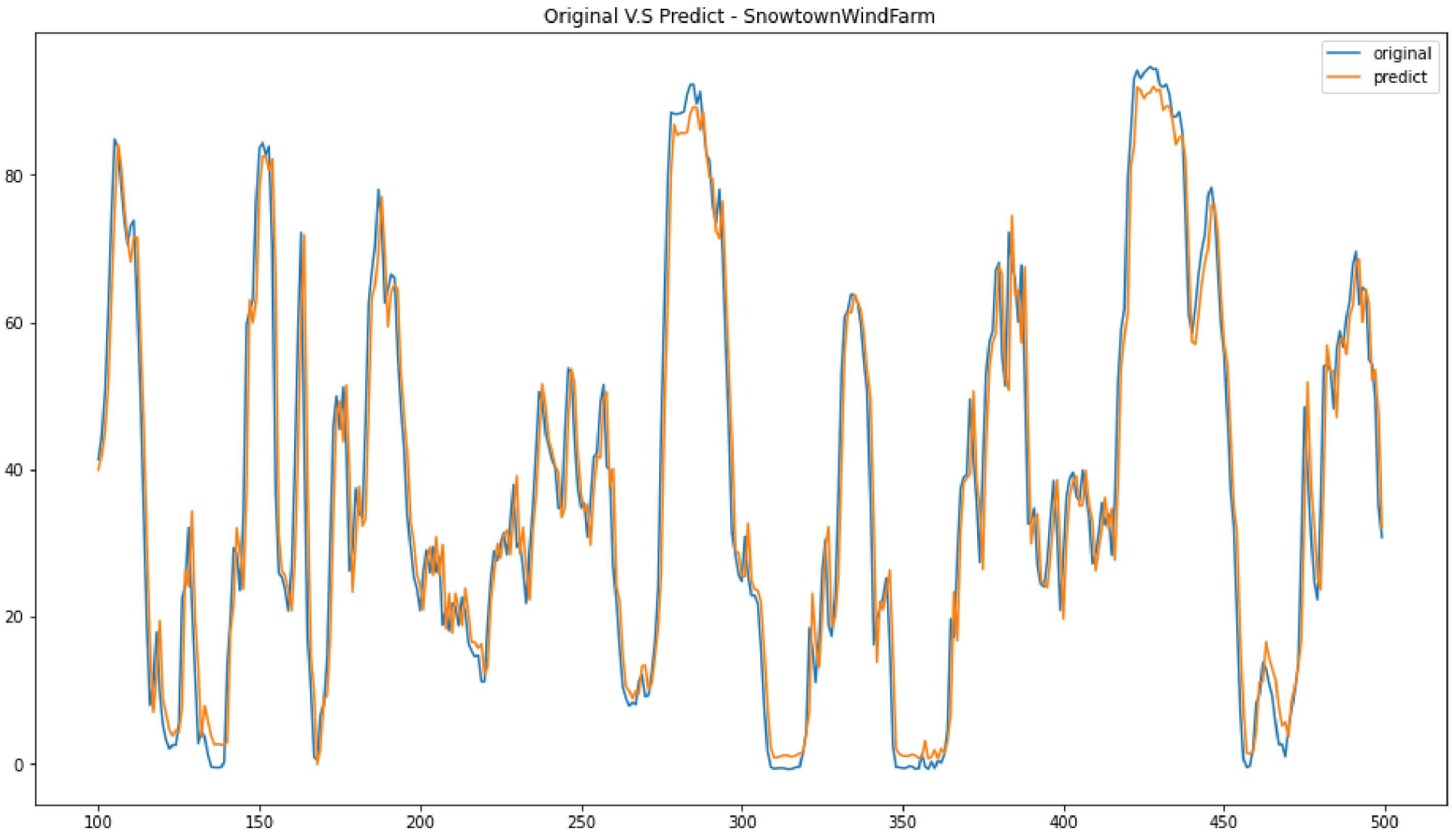
In [62]:
```
best_arma = ARMA(wind_farm['Snowtown'],order=(4,0)).fit()
print(best_arma.summary())
```

```
/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-packages/statsmodels/tsa/arima_model.py:472: FutureWarnin
g:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:

import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)

  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
                              ARMA Model Results
==============================================================================
Dep. Variable:               Snowtown   No. Observations:                17520
Model:                     ARMA(4, 0)   Log Likelihood              -63321.133
Method:                        css-mle   S.D. of innovations              8.982
Date:                Mon, 04 Sep 2023   AIC                         126654.266
Time:                        01:46:13   BIC                         126700.893
Sample:                             0   HQIC                        126669.619

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const            36.6608      1.253     29.253      0.000      34.204      39.117
ar.L1.Snowtown    1.1528      0.008    152.678      0.000       1.138       1.168
ar.L2.Snowtown   -0.2644      0.012    -22.983      0.000      -0.287      -0.242
ar.L3.Snowtown    0.0911      0.012      7.915      0.000       0.069       0.114
ar.L4.Snowtown   -0.0335      0.008     -4.437      0.000      -0.048      -0.019
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.0722           -0.0000j            1.0722           -0.0000
AR.2            2.8779           -0.0000j            2.8779           -0.0000
AR.3           -0.6158           -3.0486j            3.1102           -0.2817
AR.4           -0.6158           +3.0486j            3.1102            0.2817
-----------------------------------------------------------------------------
```

In [63]:
```
ar_preds = []
for i in range(100,500):
    ar_preds.append(best_ar.params['intercept']+best_ar.params['Snowtown.L1']*wind_farm['Snowtown'][i-1]+best_ar.params['Snowt

print('AR Model NMBE: ',NMBE(wind_farm['Snowtown'][100:500].values,np.array(ar_preds)))
print('AR Model NMAE: ',NMAE(wind_farm['Snowtown'][100:500].values,np.array(ar_preds)))
```

```
AR Model NMBE:  -0.0002013561053139933
AR Model NMAE:  0.13404619691332592
```

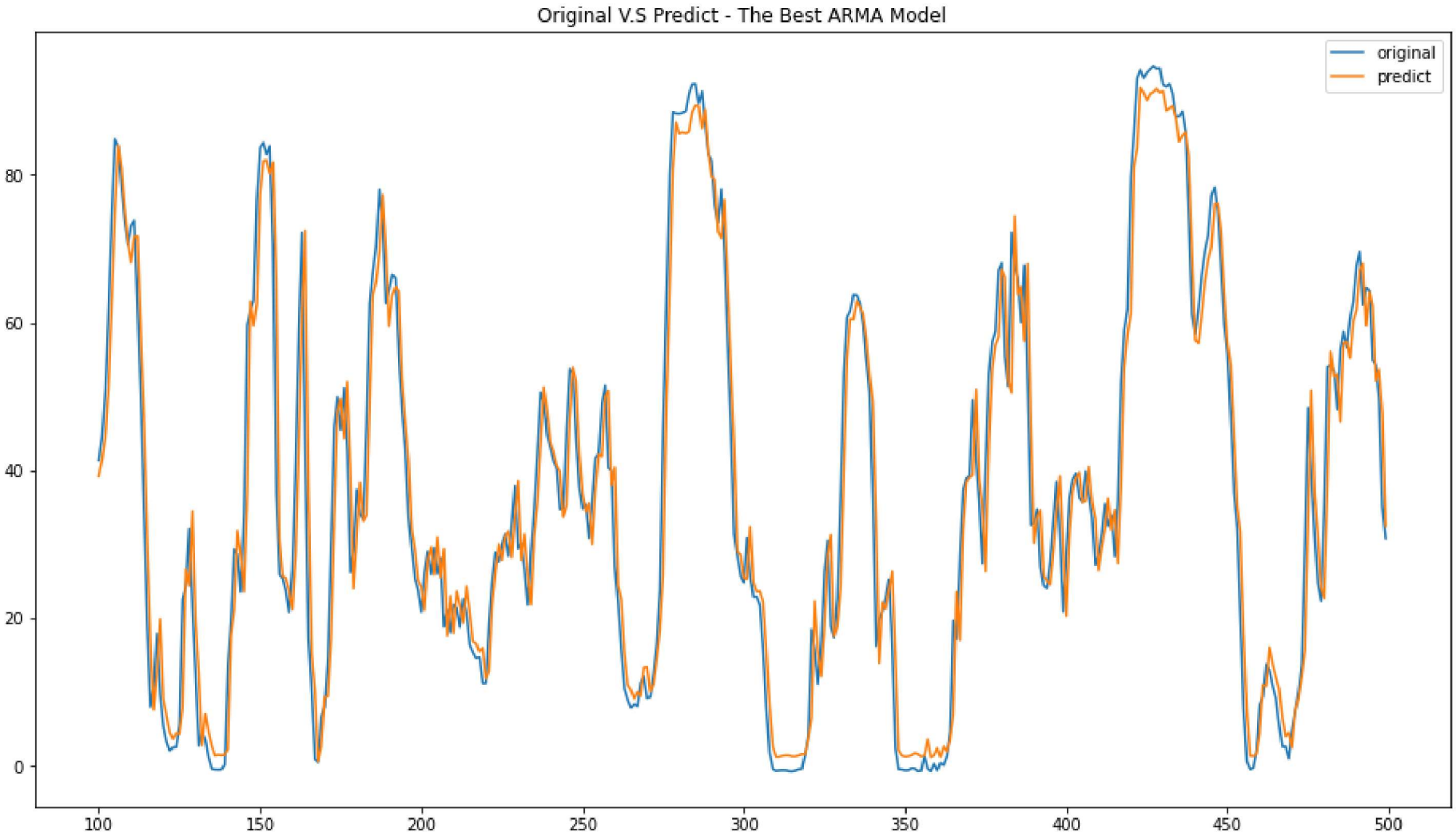
In [64]:
```
plt.figure(figsize=(16,9))
plt.plot(range(100,500),wind_farm['Snowtown'][100:500].values,label='original')
plt.plot(range(100,500),ar_preds,label='predict')
plt.title('Original V.S Predict - The Best AR Model')
plt.legend()
plt.show()
```



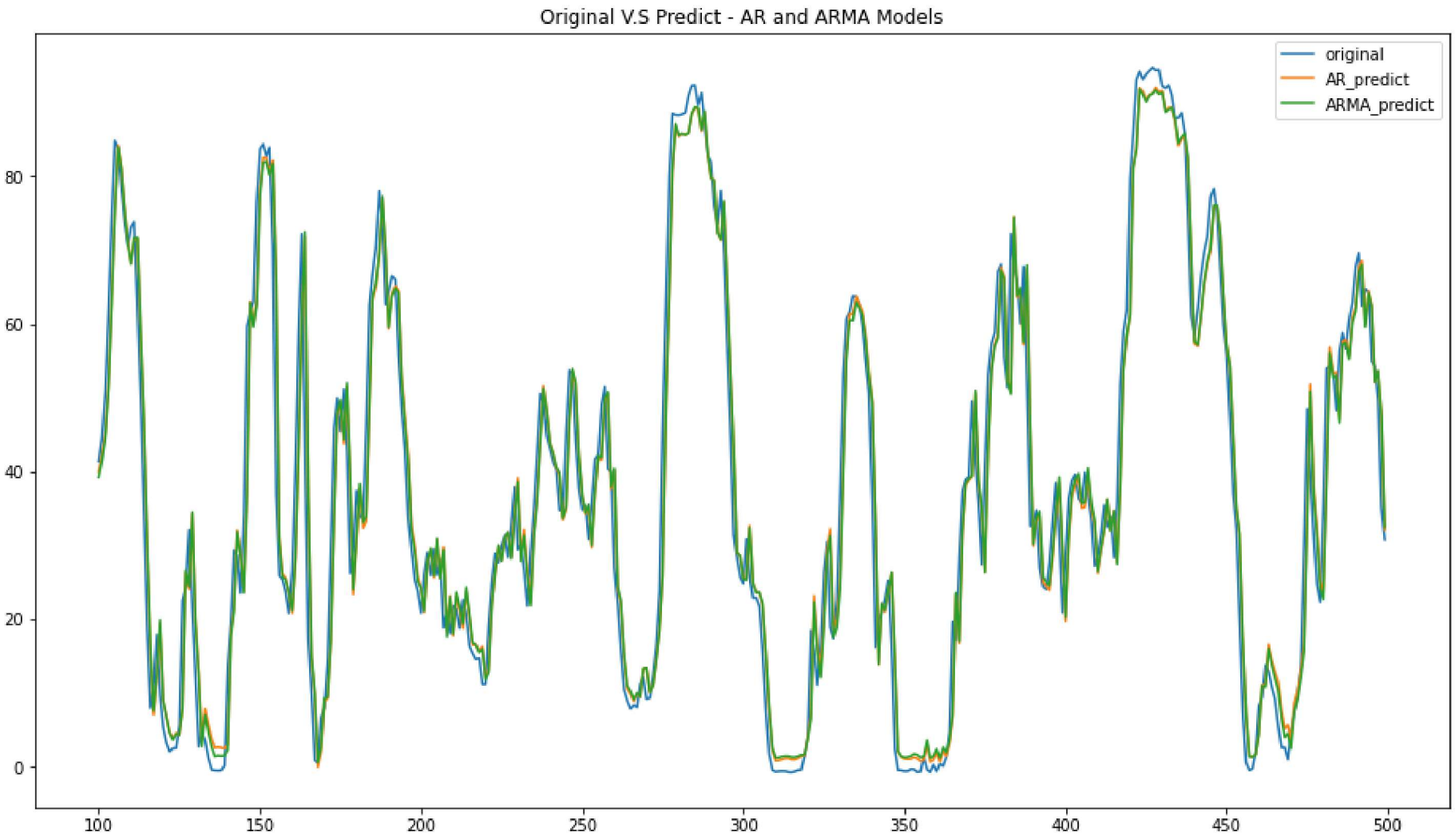
In [65]:
```
arma_preds = best_arma.predict(100,500)

print('ARMA Model NMBE: ',NMBE(wind_farm['Snowtown'][100:500].values,arma_preds.values))
print('ARMA Model NMAE: ',NMAE(wind_farm['Snowtown'][100:500].values,arma_preds.values))
```

```
ARMA Model NMBE:  0.0009135746653356808
ARMA Model NMAE:  0.13451608620153263
```

In [70]:
```
plt.figure(figsize=(16,9))
plt.plot(range(100,500),wind_farm['Snowtown'][100:500].values,label='original')
plt.plot(range(100,500),arma_preds[:-1],label='predict')
plt.title('Original V.S Predict - The Best ARMA Model')
plt.legend()
plt.show()
```



In [71]:
```
plt.figure(figsize=(16,9))
plt.plot(range(100,500),wind_farm['Snowtown'][100:500].values,label='original')
plt.plot(range(100,500),ar_preds,label='AR_predict')
plt.plot(range(100,500),arma_preds[:-1],label='ARMA_predict')
plt.title('Original V.S Predict - AR and ARMA Models')
plt.legend()
plt.show()
```



Is there a difference in the number of parameters to estimate in the two models?

According to the above results, the optimal AR model and the optimal ARMA model are very similar, and the performance of the two indicators of NMBE and NMAE is also very close.

This is because the lags parameter of the optimal AR model is [1,2,3,4,5,45], and the (p,q) parameter of the optimal ARMA model is (4, 0), which means, The optimal ARMA model is actually an AR model, and the lags parameter is [1,2,3,4].

It can be seen that the parameters of the two models themselves are very close, which is why the performance of the two is almost the same.