```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

!pip install pmdarima
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.4-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl
(2.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 10.2 MB/s eta
0:00:00
ent already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-
packages (from pmdarima) (1.3.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.9)
Requirement already satisfied: numpy>=1.21.2 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (1.25.2)
Requirement already satisfied: pandas>=0.19 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.3)
Requirement already satisfied: scikit-learn>=0.22 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (1.11.4)
Requirement already satisfied: statsmodels>=0.13.2 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.1)
Requirement already satisfied: urllib3 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (2.0.7)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (67.7.2)
Requirement already satisfied: packaging>=17.1 in
/usr/local/lib/python3.10/dist-packages (from pmdarima) (24.0)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima)
(2023.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22-
>pmdarima) (3.4.0)
Requirement already satisfied: patsy>=0.5.4 in
/usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2-
>pmdarima) (0.5.6)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-
packages (from patsy>=0.5.4->statsmodels>=0.13.2->pmdarima) (1.16.0)
```

```
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.4
```

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima.model import ARIMA
from pmdarima.arima import auto_arima
from sklearn.metrics import mean_squared_error
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

df_all = pd.read_csv("/continuous dataset.csv", parse_dates =
['datetime'], index_col = ['datetime'])

print("There are %0.0f" %df_all.shape[0] + " repeated measures and
%0.0f" %df_all.shape[1] +" variables in the dataset" )

df_all.head()
```

```
There are 48048 repeated measures and 16 variables in the dataset
```

```
{"summary":"{\n  \"name\": \"df_all\",\n  \"rows\": 48048,\n
\"fields\": [\n    {\n      \"column\": \"datetime\",\n
\"properties\": {\n        \"dtype\": \"date\",\n        \"min\":
\"2015-01-03 01:00:00\",\n        \"max\": \"2020-06-27 00:00:00\",\n
\"num_unique_values\": 48048,\n        \"samples\": [\n
\"2020-02-23 21:00:00\",\n          \"2016-09-27 09:00:00\",\n
\"2016-12-20 03:00:00\"\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"nat_demand\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 192.06889632625837,\n
\"min\": 85.19250000000002,\n        \"max\": 1754.882,\n
\"num_unique_values\": 47909,\n        \"samples\": [\n
1545.4529,\n          1579.9799,\n          1339.4052\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"T2M_toc\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
1.6754622417699032,\n        \"min\": 22.953454589843773,\n
\"max\": 35.03957519531252,\n        \"num_unique_values\": 42237,\n
\"samples\": [\n          29.281640625000023,\n
24.036706542968773,\n          26.281854248046898\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"QV2M_toc\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.0016071404585751856,\n        \"min\": 0.0120538585,\n
\"max\": 0.022690402,\n        \"num_unique_values\": 47248,\n
```

```
\"samples\": [\n          0.019157747,\n          0.019322973,\n
0.020233907\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"TQL_toc\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0.06558925069650688,\n        \"min\": 0.0,\n
\"max\": 0.52124023,\n        \"num_unique_values\": 13050,\n
\"samples\": [\n          0.022590637,\n          0.004293442,\n
0.010044098\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"W2M_toc\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 7.295502250424622,\n        \"min\":
0.0089788897306018,\n        \"max\": 39.22972628709697,\n
\"num_unique_values\": 48023,\n        \"samples\": [\n
20.093922439238703,\n          22.15020528659455 2,\n
11.640334714815946\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"T2M_san\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 3.018128790565396,\n        \"min\":
19.765222167968773,\n        \"max\": 39.06343994140627,\n
\"num_unique_values\": 44190,\n        \"samples\": [\n
31.645532226562523,\n          24.889215087890648,\n
29.191674804687523\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"QV2M_san\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.0018885343949557574,\n        \"min\":
0.010246815,\n        \"max\": 0.02216483,\n
\"num_unique_values\": 47316,\n        \"samples\": [\n
0.017702429,\n          0.018721098,\n          0.018792616\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"TQL_san\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0.08629285525954415,\n        \"min\": 8.895993e-
06,\n        \"max\": 0.48498535,\n        \"num_unique_values\":
13179,\n        \"samples\": [\n          0.0071487427,\n
0.0039424896,\n          0.21185303\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"W2M_san\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
4.103711372531107,\n        \"min\": 0.0603936728995126,\n
\"max\": 24.483937231847477,\n        \"num_unique_values\": 48026,\n
\"samples\": [\n          9.523158134136928,\n
3.611995239368736,\n          0.9835783159591852\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"T2M_dav\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
2.4140191912473736,\n        \"min\": 19.933740234375023,\n
\"max\": 34.21621093750002,\n        \"num_unique_values\": 43088,\n
\"samples\": [\n          30.771081542968773,\n
25.179650878906276,\n          22.677697753906276\n        ],\n
```

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"QV2M_dav\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
0.0015843232644833793,\n        \"min\": 0.009655291,\n
\"max\": 0.021066189,\n        \"num_unique_values\": 47262,\n
\"samples\": [\n          0.01738023,\n          0.017588386,\n
0.017571086\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"TQL_dav\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0.08789897442407868,\n        \"min\": 3.2305717e-
05,\n        \"max\": 0.4777832,\n        \"num_unique_values\":
9607,\n        \"samples\": [\n          0.0892334,\n
0.03968811,\n          0.022514343\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"W2M_dav\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
1.7105216506956396,\n        \"min\": 0.0154974073332611,\n
\"max\": 10.288901666240111,\n        \"num_unique_values\": 48022,\n
\"samples\": [\n          3.345817745136007,\n
2.036861394489706,\n          2.066743888745352\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Holiday_ID\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3,\n        \"min\": 0,\n        \"max\": 22,\n
\"num_unique_values\": 23,\n        \"samples\": [\n          16,\n
10,\n          0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"holiday\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"school\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df_all"}
```

```
df = df_all['nat_demand'].resample('W').sum()
df = df[1:273]
df
```

```
datetime
2015-01-11    181919.6224
2015-01-18    188082.3152
2015-01-25    179448.7184
2015-02-01    184393.4256
2015-02-08    187290.1846
                  ...
```

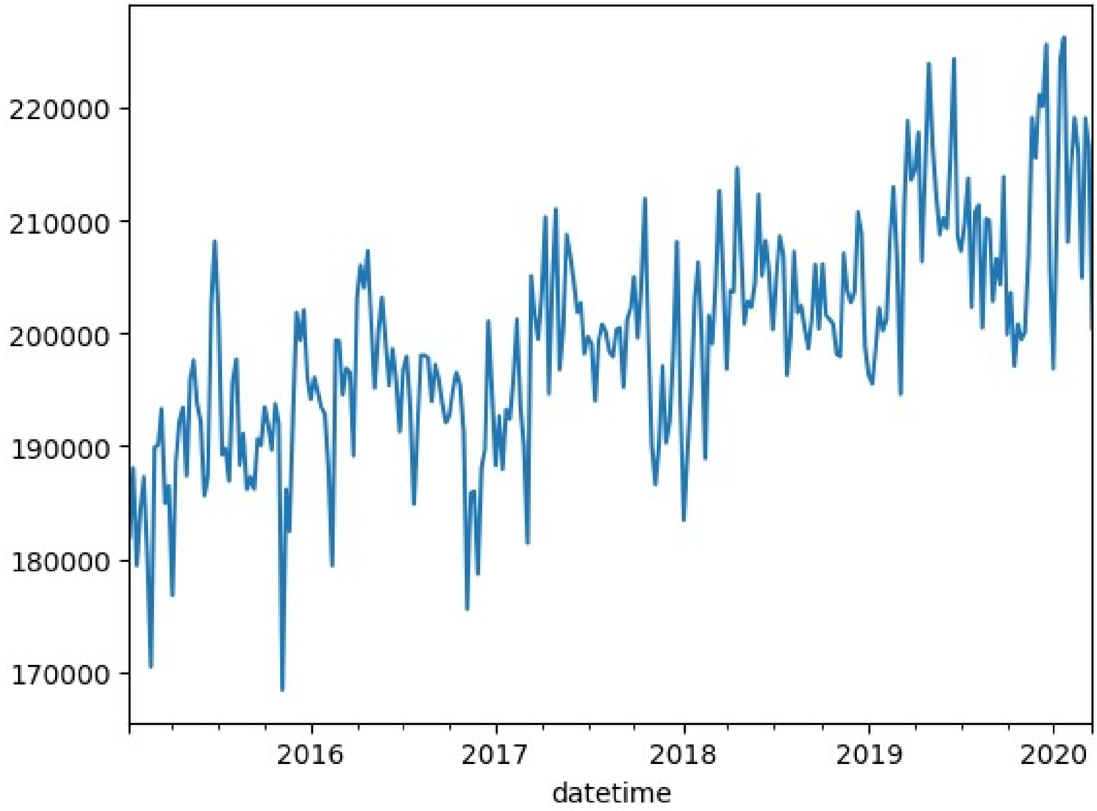
```
2020-02-23    216005.1882
2020-03-01    204924.9816
2020-03-08    219065.4724
2020-03-15    216436.7037
2020-03-22    200434.6963
Freq: W-SUN, Name: nat_demand, Length: 272, dtype: float64
```

```
from statistics import mean

print("Max: ",max(df_all["nat_demand"]))
print("Mean: ",mean(df_all["nat_demand"]))
print("Min: ",min(df_all["nat_demand"]))
```

```
Max:  1754.882
Mean:  1182.8686472323864
Min:  85.19250000000002
```

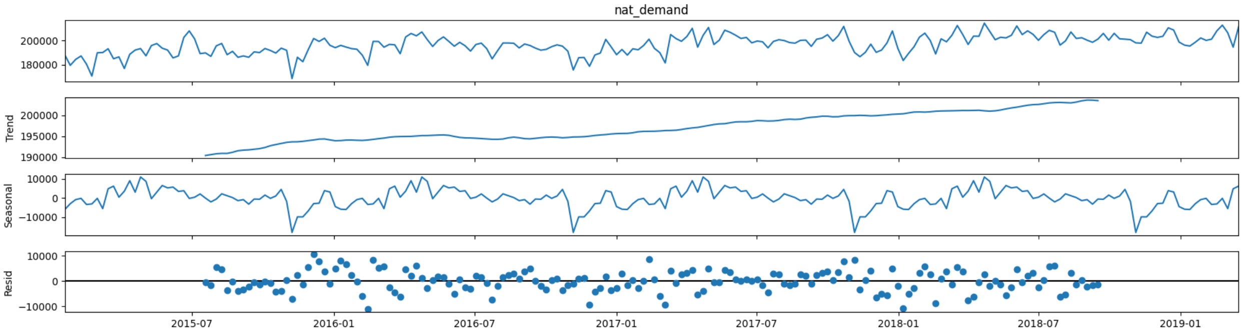
```
df.plot()
```

```
<Axes: xlabel='datetime'>
```



```
df_train = df[1:219]
df_test = df[219:273]
```

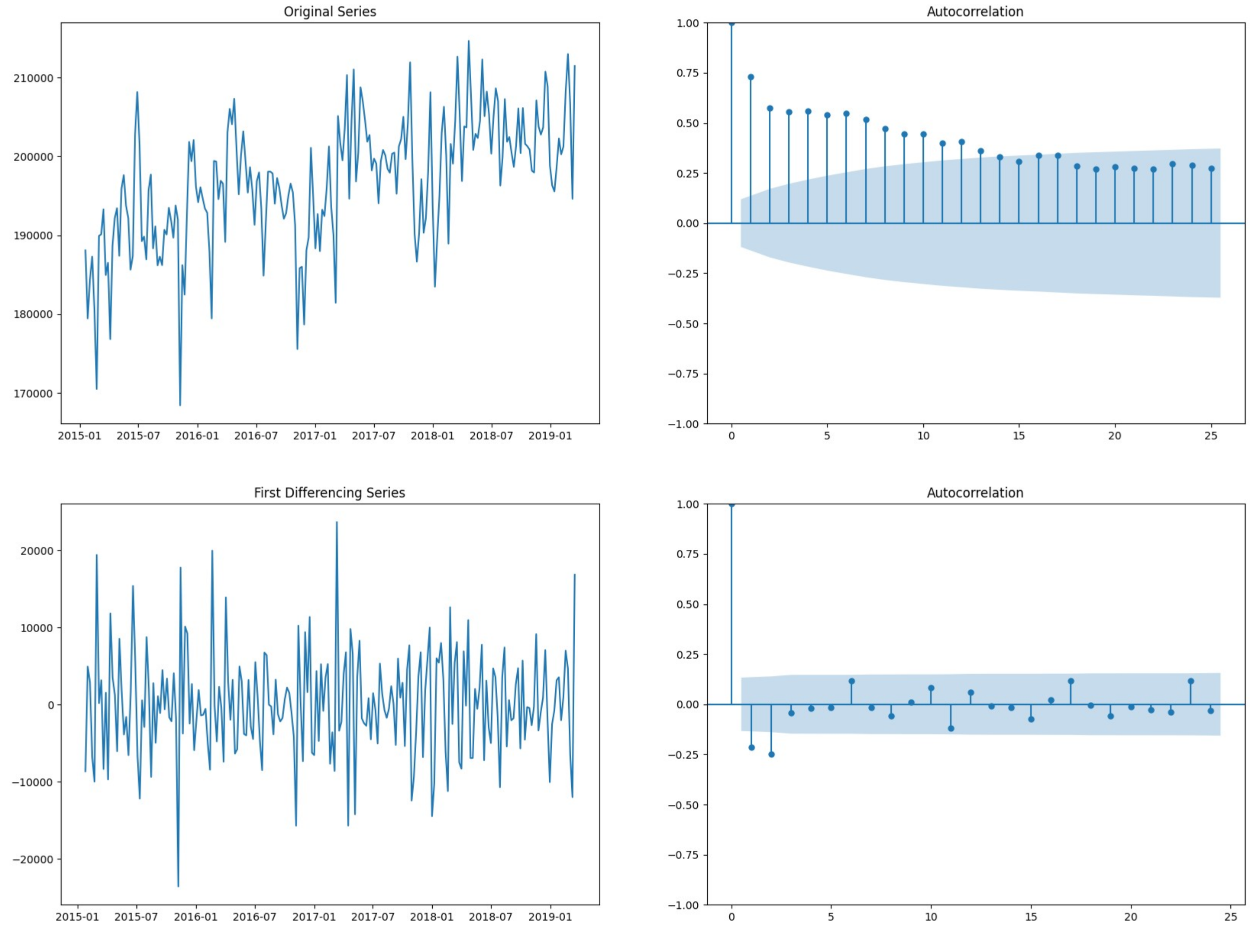
```
decomp = sm.tsa.seasonal_decompose(df_train,model = 'additive')
fig = decomp.plot()
fig.set_figwidth(20)
```



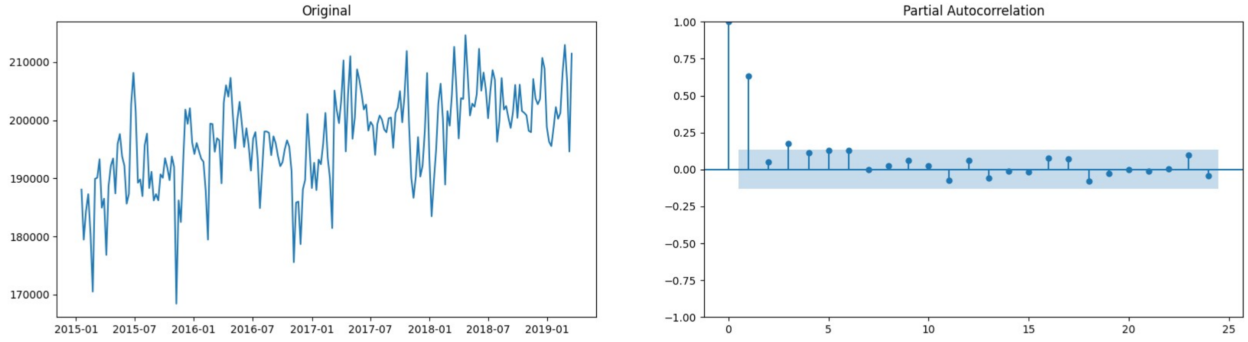
```
adf = adfuller(df_train)
print('adfuller test P-Value: ', adf[1])
```

```
adfuller test P-Value:  0.020741302189223345
```

```
fig, axes = plt.subplots(2, 2, figsize = (20,15))
axes[0,0].plot(df_train)
axes[0,0].set_title('Original Series')
plot_acf(df, ax=axes[0,1])
axes[1,0].plot(df_train.diff())
axes[1,0].set_title('First Differencing Series')
plot_acf(df_train.diff().dropna(), ax=axes[1,1])
plt.show()
```

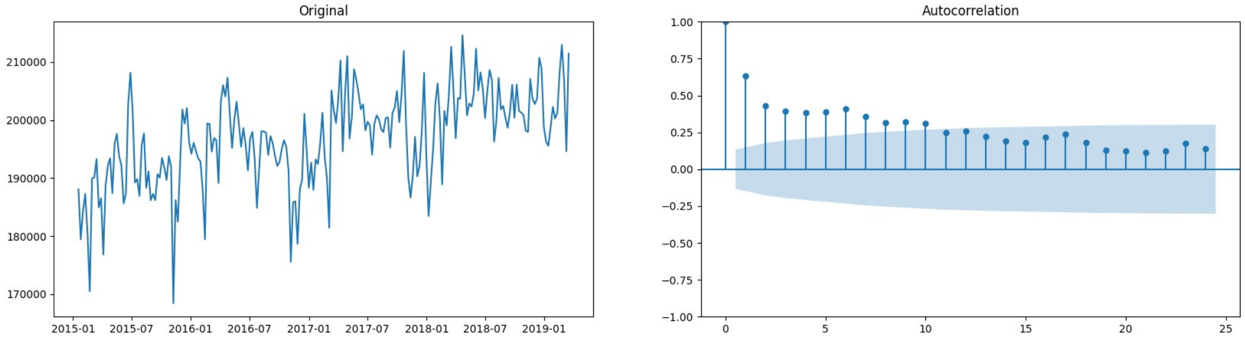


```
fig, axes = plt.subplots(1, 2, figsize = (20,5))
axes[0].plot(df_train); axes[0].set_title('Original')
#axes[1].set(ylim=(0,5))
plot_pacf(df_train, ax=axes[1])

plt.show()
```



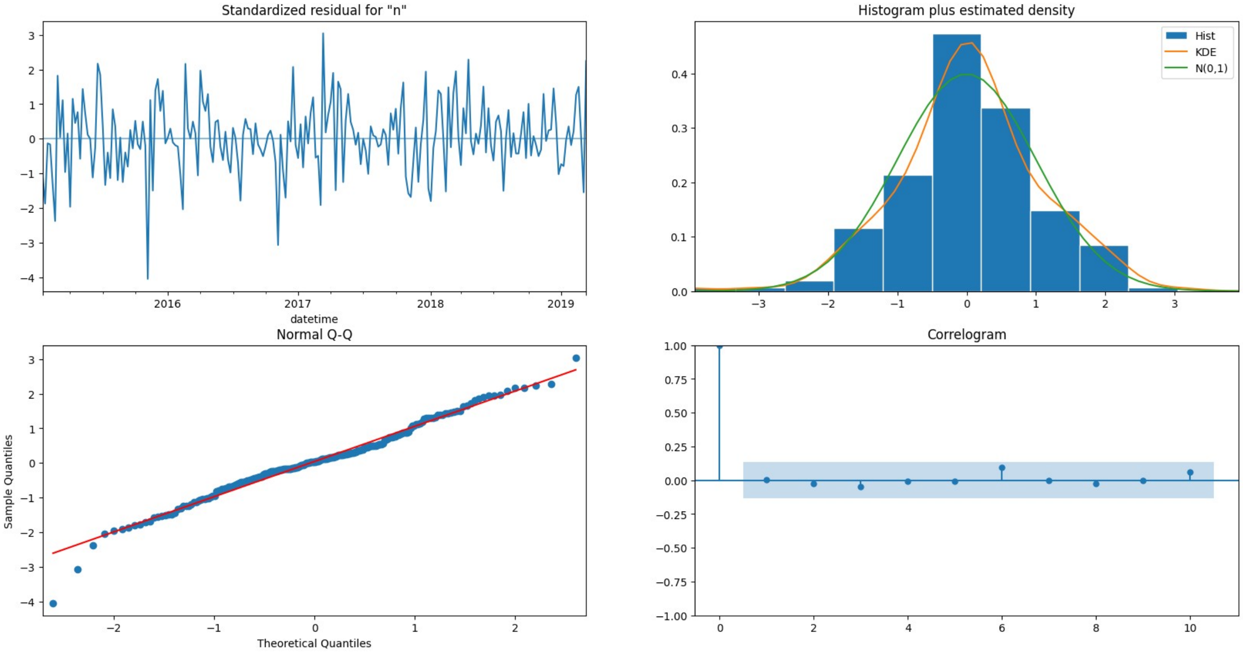
```
fig, axes = plt.subplots(1, 2, figsize = (20,5))
axes[0].plot(df_train); axes[0].set_title('Original')
plot_acf(df_train, ax=axes[1])
```

```
plt.show()
```



```
model = ARIMA(df_train,exog = None, order = (1,0,2)).fit()
model.summary()

<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results

======================================================================
========
Dep. Variable:             nat_demand   No. Observations:
218
Model:                 ARIMA(1, 0, 2)   Log Likelihood                -
2200.850
Date:                Fri, 29 Mar 2024   AIC
4411.700
Time:                        10:22:56   BIC
4428.622
Sample:                    01-18-2015   HQIC
4418.535
                         - 03-17-2019

Covariance Type:                  opg

======================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
const       1.968e+05   3712.062     53.026      0.000     1.9e+05
2.04e+05
ar.L1          0.9756      0.022     44.091      0.000       0.932
1.019
ma.L1         -0.4420      0.067     -6.631      0.000      -0.573
-0.311
```

```
ma.L2          -0.3202      0.069     -4.641      0.000      -0.455
-0.185
sigma2      3.315e+07      0.053   6.29e+08      0.000    3.32e+07
3.32e+07
======================================================================
=============
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
11.37
Prob(Q):                              0.98   Prob(JB):
0.00
Heteroskedasticity (H):               0.73   Skew:
-0.21
Prob(H) (two-sided):                  0.18   Kurtosis:
4.04
======================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
[2] Covariance matrix is singular or near-singular, with condition
number 1.58e+25. Standard errors may be unstable.
"""

model.plot_diagnostics(figsize=(20,10))
plt.show()
```

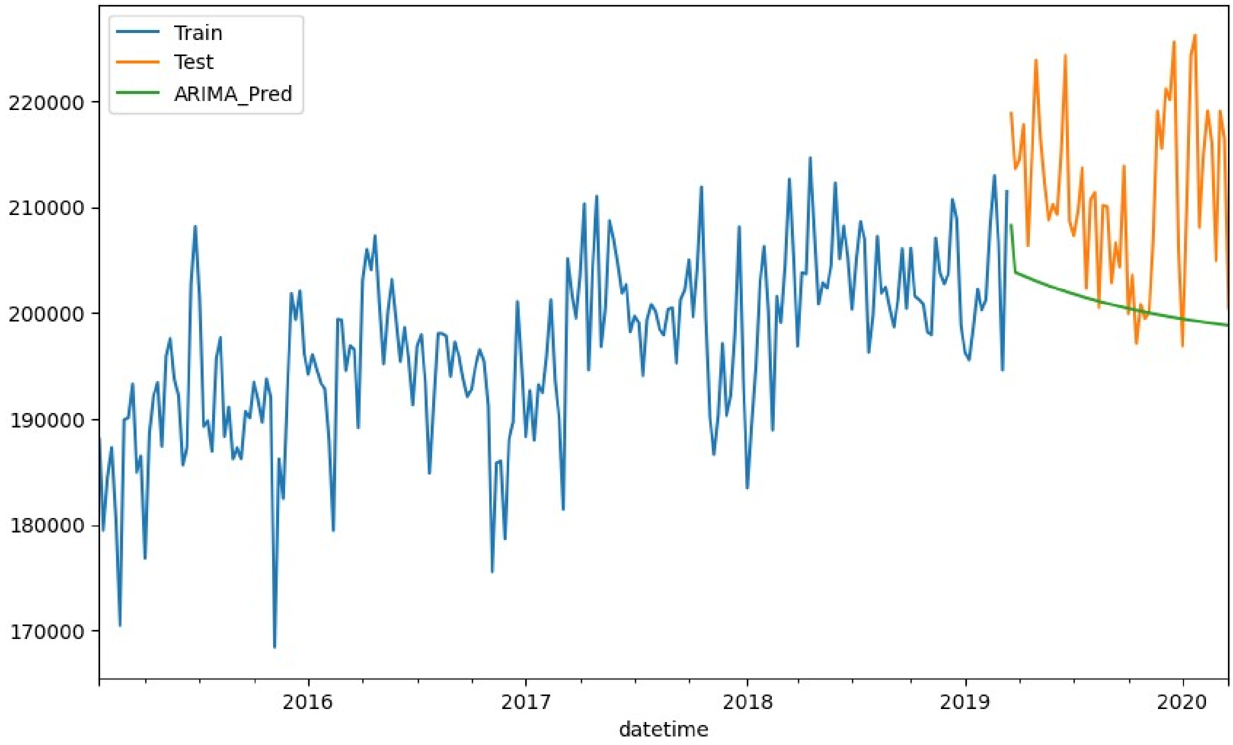


```
pred = model.predict(start = len(df_train),end = len(df)-2)
print("The Root Mean Squared Error is: "+
str(np.sqrt(mean_squared_error(df_test,pred))))
```

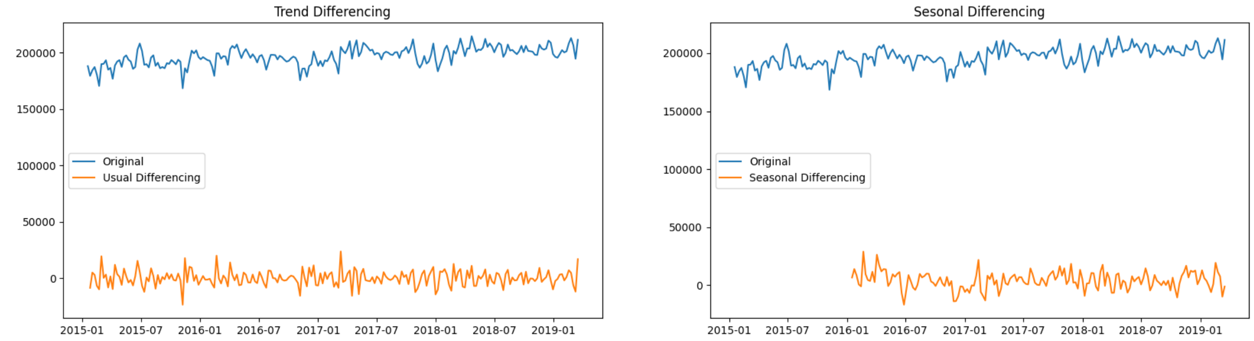
```
The Root Mean Squared Error is: 12745.584882387127
```

```
df_train.plot(legend = True,label = 'Train', figsize=(10,6))
df_test.plot(legend = True,label = 'Test')
pred.plot(legend = True,label = 'ARIMA_Pred')
```

```
<Axes: xlabel='datetime'>
```



```
fig, axes = plt.subplots(1, 2, figsize = (20,5))
axes[0].plot(df_train, label= 'Original')
axes[0].plot(df_train.diff(1), label= 'Usual Differencing')
axes[0].set_title('Trend Differencing')
axes[0].legend(loc='center left', fontsize=10)
axes[1].plot(df_train, label= 'Original')
axes[1].plot(df_train.diff(52), label= 'Seasonal Differencing')
axes[1].set_title('Sesonal Differencing')
axes[1].legend(loc='center left', fontsize=10)
plt.show()
```

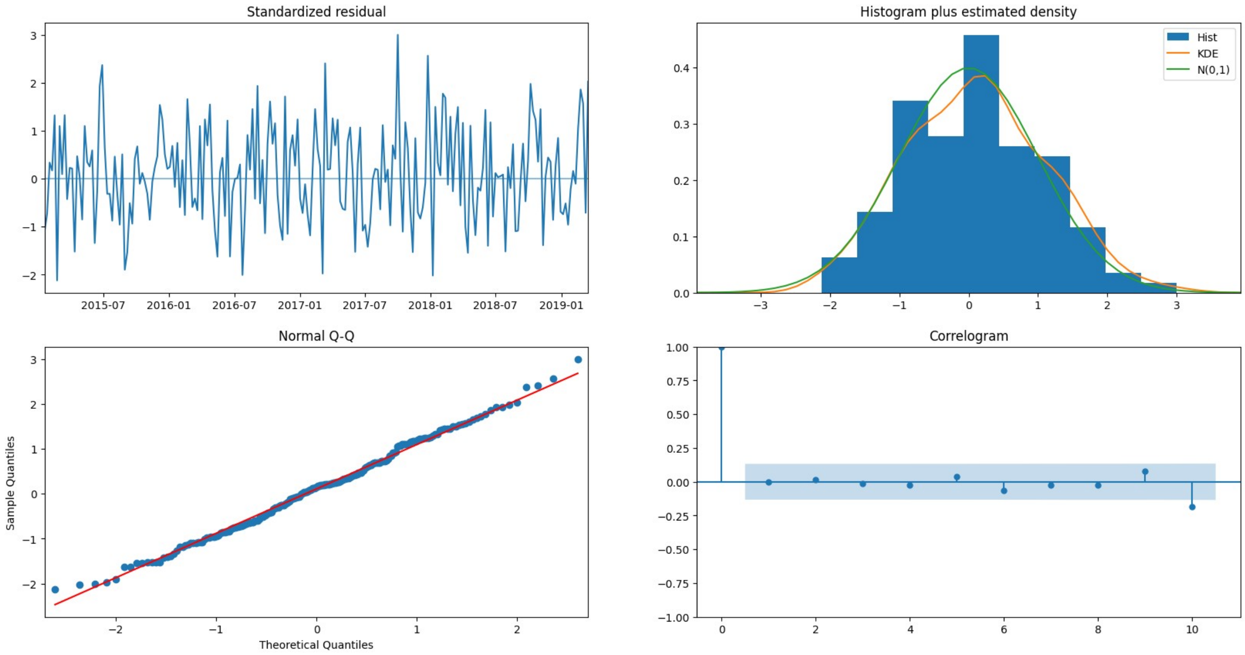


```
arima= auto_arima(df_train,trace=True, error_action='ignore', test =
'adf',
                         start_p=1,start_q=1,max_p=10,max_q=10,m=1,
D=0,

suppress_warnings=True,stepwise=True,seasonal=False)
arima.summary()

Performing stepwise search to minimize aic
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=4439.281, Time=0.59 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=5935.911, Time=0.05 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=inf, Time=0.07 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=5780.440, Time=0.16 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=4425.676, Time=0.81 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=inf, Time=0.31 sec
 ARIMA(3,0,1)(0,0,0)[0]             : AIC=4421.632, Time=1.15 sec
 ARIMA(3,0,0)(0,0,0)[0]             : AIC=inf, Time=0.71 sec
 ARIMA(4,0,1)(0,0,0)[0]             : AIC=4446.972, Time=1.11 sec
 ARIMA(3,0,2)(0,0,0)[0]             : AIC=inf, Time=1.18 sec
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=inf, Time=0.32 sec
 ARIMA(4,0,0)(0,0,0)[0]             : AIC=inf, Time=0.32 sec
 ARIMA(4,0,2)(0,0,0)[0]             : AIC=inf, Time=0.62 sec
 ARIMA(3,0,1)(0,0,0)[0] intercept   : AIC=4424.894, Time=0.51 sec

Best model:  ARIMA(3,0,1)(0,0,0)[0]
Total fit time: 7.970 seconds

<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results

======================================================================
========
Dep. Variable:                      y   No. Observations:
218
Model:               SARIMAX(3, 0, 1)   Log Likelihood                  -
2205.816
Date:                Fri, 29 Mar 2024   AIC
4421.632
```

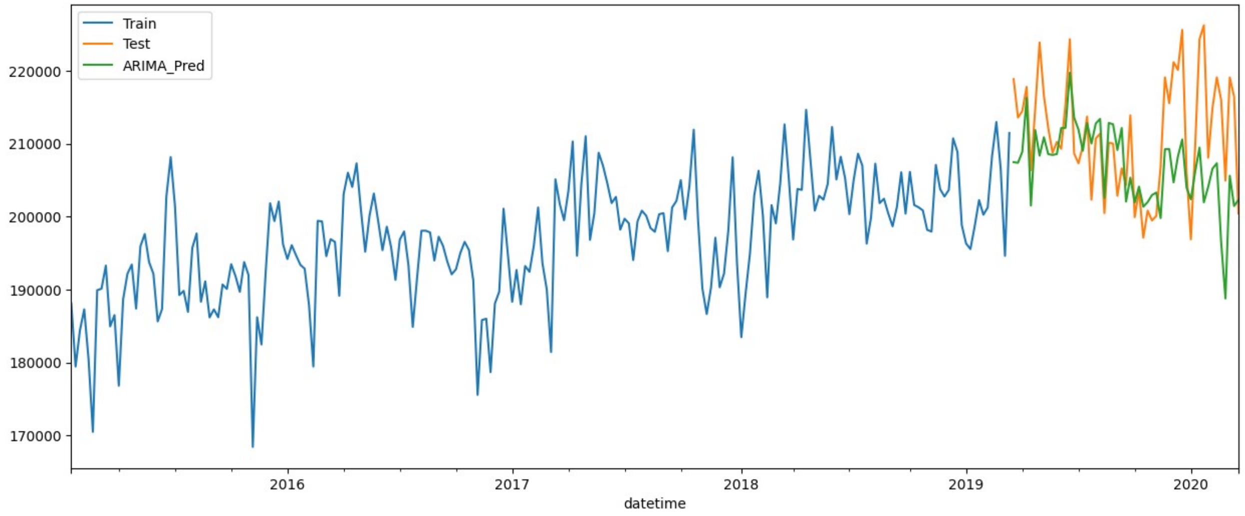
```
Time:                        10:26:13   BIC
4438.555
Sample:                    01-18-2015   HQIC
4428.467
                         - 03-17-2019

Covariance Type:                  opg

======================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
ar.L1          1.3656      0.087     15.718      0.000       1.195
1.536
ar.L2         -0.5351      0.123     -4.366      0.000      -0.775
-0.295
ar.L3          0.1695      0.090      1.879      0.060      -0.007
0.346
ma.L1         -0.8198      0.068    -11.981      0.000      -0.954
-0.686
sigma2      3.966e+07   6.78e-10   5.85e+16      0.000    3.97e+07
3.97e+07
======================================================================
=============
Ljung-Box (L1) (Q):                  0.07   Jarque-Bera (JB):
7.32
Prob(Q):                             0.79   Prob(JB):
0.03
Heteroskedasticity (H):              0.72   Skew:
-0.11
Prob(H) (two-sided):                 0.16   Kurtosis:
3.87
======================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
[2] Covariance matrix is singular or near-singular, with condition
number 4.82e+32. Standard errors may be unstable.
"""

df_exo = df_all.resample('W').sum().iloc[:,1:]
exo_train = df_exo[2:220]
exo_test = df_exo[220:273]

arimax= auto_arima(df_train,trace=True, X =
exo_train[['T2M_toc','TQL_toc','QV2M_san','QV2M_dav','holiday']]
```

```
                   , error_action='ignore', test = 'adf',
approximation=False
                   , start_p=0,start_q=0,max_p=10,max_q=10,m=1, D=0,

suppress_warnings=True,stepwise=True,seasonal=False)
arimax.summary()

Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=5464.301, Time=0.06 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=4282.008, Time=0.49 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=4387.906, Time=1.28 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=4269.989, Time=1.05 sec
 ARIMA(3,0,0)(0,0,0)[0]             : AIC=4261.630, Time=0.82 sec
 ARIMA(4,0,0)(0,0,0)[0]             : AIC=4258.813, Time=0.57 sec
 ARIMA(5,0,0)(0,0,0)[0]             : AIC=4255.799, Time=0.68 sec
 ARIMA(6,0,0)(0,0,0)[0]             : AIC=4257.615, Time=1.67 sec
 ARIMA(5,0,1)(0,0,0)[0]             : AIC=4252.409, Time=2.27 sec
 ARIMA(4,0,1)(0,0,0)[0]             : AIC=4250.952, Time=1.02 sec
 ARIMA(3,0,1)(0,0,0)[0]             : AIC=4249.808, Time=0.89 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=4247.665, Time=1.96 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=4253.358, Time=2.05 sec
 ARIMA(2,0,2)(0,0,0)[0]             : AIC=4248.961, Time=2.57 sec
 ARIMA(1,0,2)(0,0,0)[0]             : AIC=4247.552, Time=2.32 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=4375.582, Time=1.99 sec
 ARIMA(1,0,3)(0,0,0)[0]             : AIC=4249.519, Time=1.86 sec
 ARIMA(0,0,3)(0,0,0)[0]             : AIC=4355.916, Time=1.65 sec
 ARIMA(2,0,3)(0,0,0)[0]             : AIC=inf, Time=1.09 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=4249.596, Time=0.72 sec

Best model:  ARIMA(1,0,2)(0,0,0)[0]
Total fit time: 27.085 seconds

<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results

======================================================================
========
Dep. Variable:                      y   No. Observations:
218
Model:               SARIMAX(1, 0, 2)   Log Likelihood              -
2114.776
Date:                Fri, 29 Mar 2024   AIC
4247.552
Time:                        10:34:04   BIC
4278.013
Sample:                    01-18-2015   HQIC
4259.856
                         - 03-17-2019
```

```
Covariance Type:                  opg

======================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
T2M_toc       43.4804      2.212     19.653      0.000      39.144
47.817
TQL_toc     -337.0734     63.729     -5.289      0.000    -461.980
-212.167
QV2M_san    4.081e+04   5842.900      6.985      0.000    2.94e+04
5.23e+04
QV2M_dav   -4.255e+04   7221.493     -5.892      0.000   -5.67e+04   -
2.84e+04
holiday     -152.5739     11.533    -13.229      0.000    -175.179
-129.969
ar.L1          0.9939      0.011     89.352      0.000       0.972
1.016
ma.L1         -0.5359      0.075     -7.138      0.000      -0.683
-0.389
ma.L2         -0.2179      0.071     -3.048      0.002      -0.358
-0.078
sigma2      1.564e+07     16.978   9.21e+05      0.000    1.56e+07
1.56e+07
======================================================================
=============
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
1.71
Prob(Q):                              0.96   Prob(JB):
0.43
Heteroskedasticity (H):               1.27   Skew:
0.14
Prob(H) (two-sided):                  0.31   Kurtosis:
2.66
======================================================================
=============

Warnings:
[1] Covariance matrix calculated using the outer product of gradients
(complex-step).
[2] Covariance matrix is singular or near-singular, with condition
number 2.57e+21. Standard errors may be unstable.
"""

arimax.plot_diagnostics(figsize=(20,10))
plt.show()
```



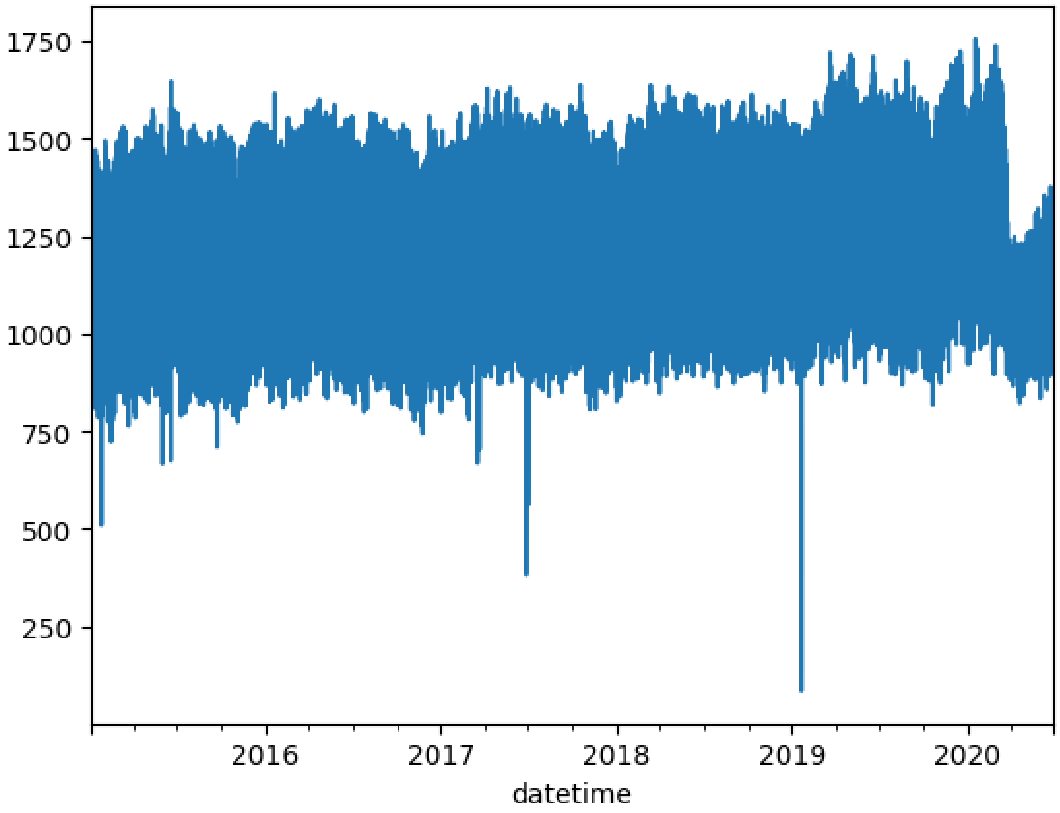
```
pred_x = arimax.predict(n_periods = len(df_test),X =
exo_test[['T2M_toc','TQL_toc','QV2M_san','QV2M_dav','holiday']] )
df_train.plot(legend = True,label = 'Train', figsize=(15,6))
df_test.plot(legend = True,label = 'Test')
pred_x.plot(legend = True,label = 'ARIMA_Pred')

<Axes: xlabel='datetime'>
```



```
print("The Root Mean Squared Error is: "+
str(np.sqrt(mean_squared_error(df_test,pred_x))))

The Root Mean Squared Error is: 8515.136350323046
```

```
df_all["nat_demand"].plot()
```

```
<Axes: xlabel='datetime'>
```


datetime