```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline


df = pd.read_csv("/WindEnergy.csv")
#df.head()
df.columns
df=df.rename(columns={"Date/Time":"DateTime"})



df = df[['DateTime','LV ActivePower (kW)','Wind Speed (m/s)','Wind Direction (°)','Theoret
df.head()
```

	DateTime	LV ActivePower (kW)	Wind Speed (m/s)	Wind Direction (°)	Theoretical_Power_Curve (KWh)
0	01 01 2018 00:00	380.047791	5.311336	259.994904	416.328908
1	01 01 2018 00:10	453.769196	5.672167	268.641113	519.917511
2	01 01 2018 00:20	306.376587	5.216037	272.564789	390.900016

```
df.index = df['DateTime']
df.head()
```

	DateTime	LV ActivePower (kW)	Wind Speed (m/s)	Wind Direction (°)	Theoretical_Power_Curve (KWh)
DateTime					
01 01 2018 00:00	01 01 2018 00:00	380.047791	5.311336	259.994904	416.328908
01 01 2018 00:10	01 01 2018 00:10	453.769196	5.672167	268.641113	519.917511

```
import datetime
df['DateTime'] = pd.to_datetime(df.DateTime,format='%d %m %Y %H:%M')


df.dtypes
```

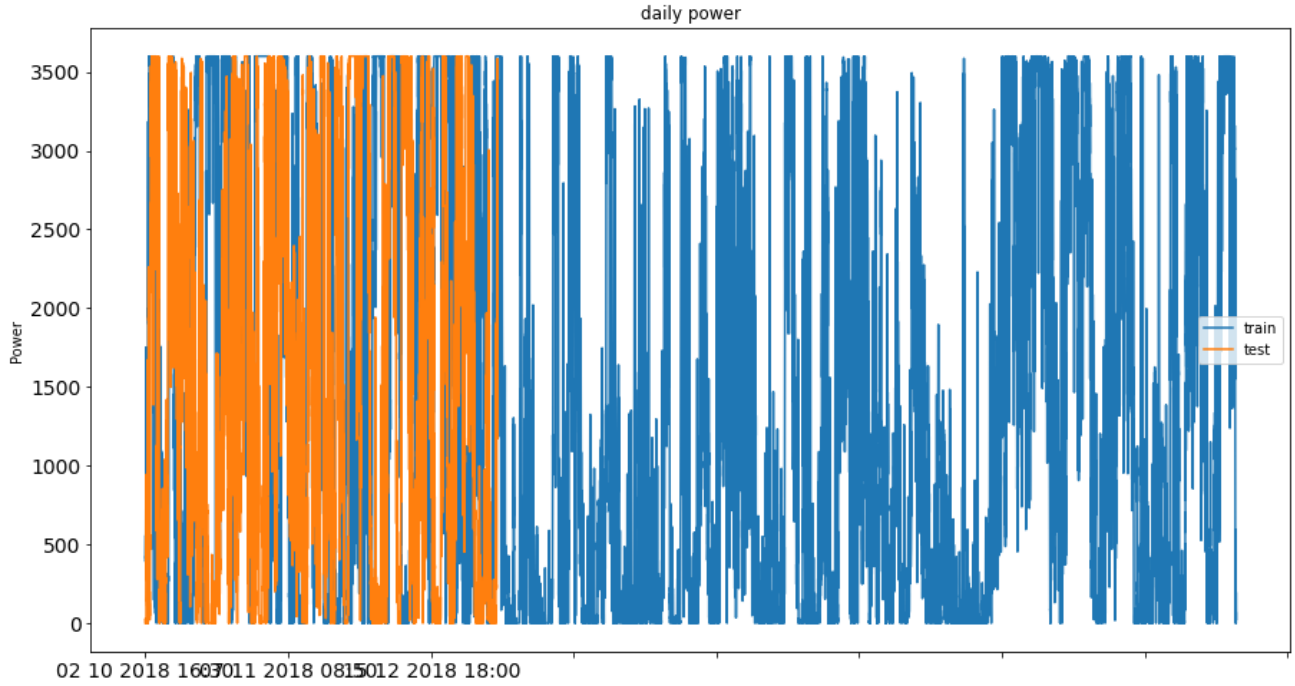
```
DateTime                          datetime64[ns]
LV ActivePower (kW)                      float64
Wind Speed (m/s)                         float64
Wind Direction (°)                       float64
Theoretical_Power_Curve (KWh)            float64
dtype: object
```

```
df.head()
```

DateTime	DateTime	LV ActivePower (kW)	Wind Speed (m/s)	Wind Direction (°)	Theoretical_Power_Curve (KWh)
01 01 2018 00:00	2018-01-01 00:00:00	380.047791	5.311336	259.994904	416.328908
01 01 2018 00:10	2018-01-01 00:10:00	453.769196	5.672167	268.641113	519.917511

```
df['year'] = pd.DatetimeIndex(df['DateTime']).year
df['month'] = pd.DatetimeIndex(df['DateTime']).month
df['date'] = pd.DatetimeIndex(df['DateTime']).date
df['hour'] = pd.DatetimeIndex(df['DateTime']).hour
df['minute'] = pd.DatetimeIndex(df['DateTime']).minute


x_train=df.loc['01 01 2018 00:00':'28 09 2018 21:20']  #till 9th month we are going to tra
x_test=df.loc['02 10 2018 16:30':'31 12 2018 23:50']   #last 3 months are used as test set


ts_train = x_train['Theoretical_Power_Curve (KWh)']
ts_test = x_test['Theoretical_Power_Curve (KWh)']
ts = pd.DataFrame(df["Theoretical_Power_Curve (KWh)"])


x_train = x_train.loc[:,"Theoretical_Power_Curve (KWh)"]
x_train.plot(figsize=(15,8), title= 'daily power', fontsize=14, label='train')
x_test = x_test.loc[:,"Theoretical_Power_Curve (KWh)"]
x_test.plot(figsize=(15,8), title= 'daily power', fontsize=14, label='test')
plt.xlabel("Datetime")
plt.ylabel("Power")
plt.legend(loc='best')
plt.show()
```

```
    print ('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Numb

    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print (dfoutput)



x_train=df.loc['01 01 2018 00:00':'28 09 2018 21:20']  #till 9th month we are going to tra
x_test=df.loc['02 10 2018 16:30':'31 12 2018 23:50']   #last 3 months are used as test set

ts_train = x_train['Theoretical_Power_Curve (KWh)']
ts_test = x_test['Theoretical_Power_Curve (KWh)']
ts = pd.DataFrame(df["Theoretical_Power_Curve (KWh)"])


from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20,10
test_stationarity(x_train['Theoretical_Power_Curve (KWh)'])
```




```
ts = pd.DataFrame(df["Theoretical_Power_Curve (KWh)"])
ts.head()
#type(ts)
ts.dtypes
```

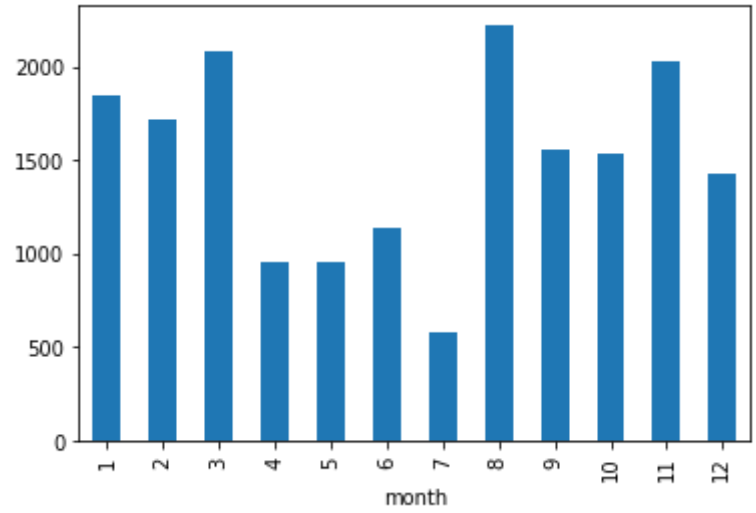
```
Theoretical_Power_Curve (KWh)    float64
dtype: object
```

```
df.columns
```

```
Index(['DateTime', 'LV ActivePower (kW)', 'Wind Speed (m/s)',
       'Wind Direction (°)', 'Theoretical_Power_Curve (KWh)', 'year', 'month',
       'date', 'hour', 'minute'],
      dtype='object')
```

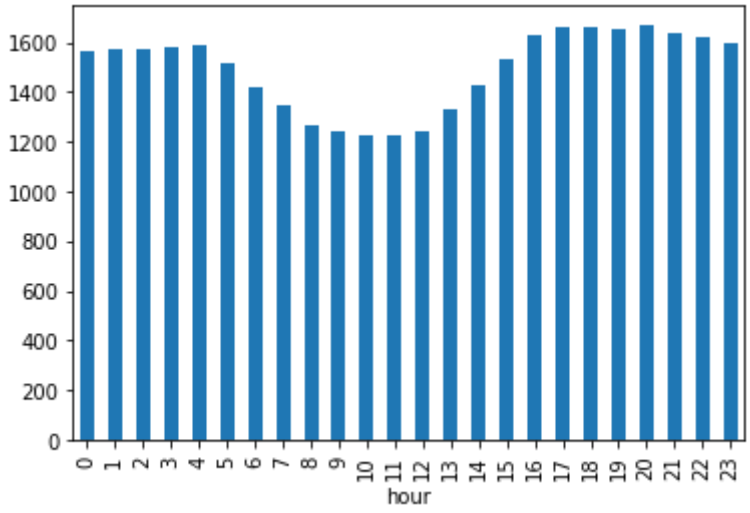
```
df.groupby('month')['Theoretical_Power_Curve (KWh)'].mean().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3dcf8c4780>
```



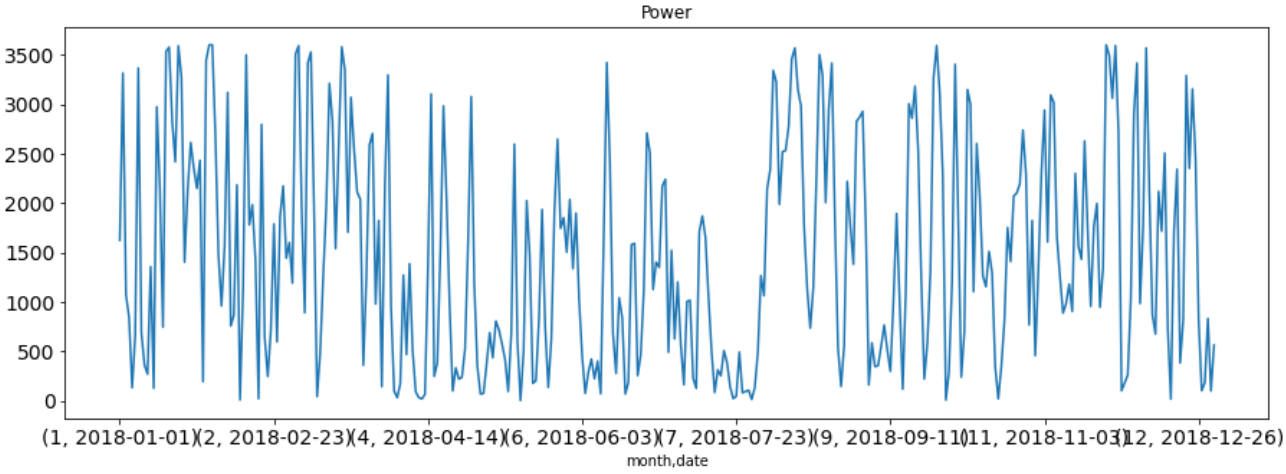

```
df.groupby(['hour'])['Theoretical_Power_Curve (KWh)'].mean().plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f3dd14ea470>

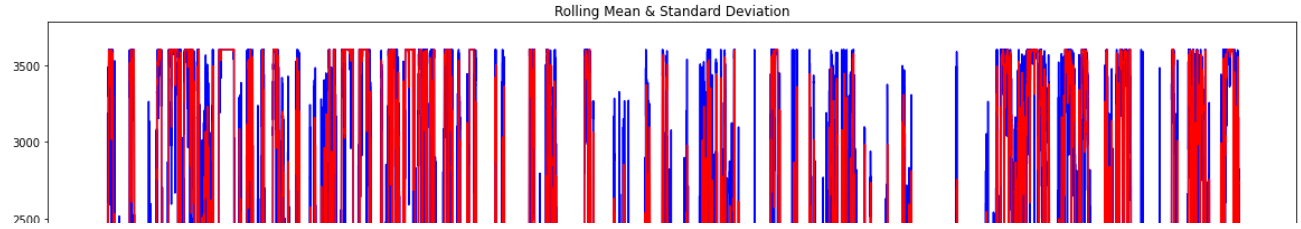



```
temp=df.groupby(['month', 'date'])['Theoretical_Power_Curve (KWh)'].mean()
temp.plot(figsize=(15,5), title= 'Power', fontsize=14)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f3dce76af60>




```
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):
        #Determing rolling statistics
    #rolmean = pd.rolling_mean(ts, window=24)
    rolmean = timeseries.rolling(12).mean()
    #rolstd = timeseries.rolling_std(ts, window=24)
    rolstd = timeseries.rolling(12).std()
        #Plot rolling statistics:
    orig = plt.plot(timeseries, color='blue',label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)
        #Perform Dickey-Fuller test:
```




```
x_train_log = np.log(x_train['Theoretical_Power_Curve (KWh)'])
x_test_log = np.log(x_test['Theoretical_Power_Curve (KWh)'])
#moving_avg = pd.rolling_mean(x_train_log, 24)
rolmean_log = x_train_log.rolling(12).mean()  #12 indicates rolling average for 12 months
```

```
train_log_moving_avg_diff = x_train_log - rolmean_log
```

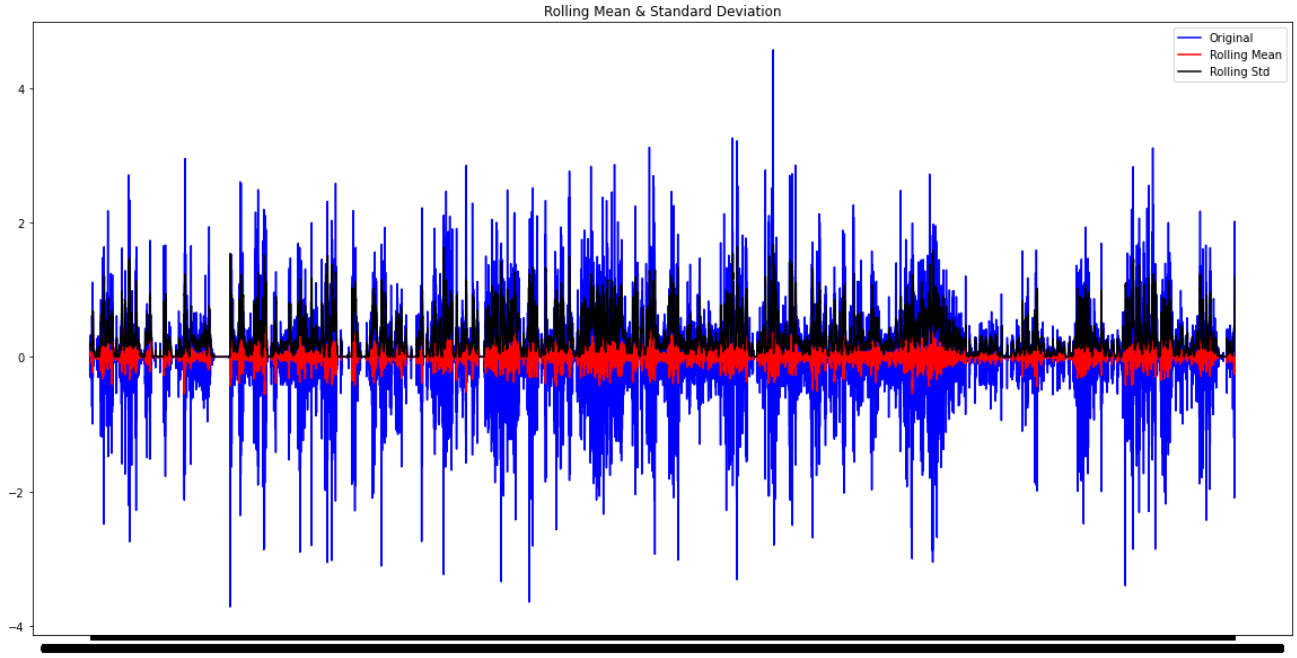

```
x_train_log_diff = x_train_log - x_train_log.shift(1)
x_train_log_diff = x_train_log_diff.replace([np.inf, -np.inf], np.nan)
x_train_log_diff = x_train_log_diff.dropna()

x_train_log = x_train_log.replace([np.inf, -np.inf], np.nan)
x_train_log = x_train_log.dropna()

x_train_log_diff = x_train_log_diff.replace([np.inf, -np.inf], np.nan)
x_train_log_diff = x_train_log_diff.dropna()
```

```
dtype: float64
```

```
x_train_log_diff.dropna(inplace = True)
x_train_log_diff
```

```
DateTime
01 01 2018 00:10    0.222195
01 01 2018 00:20   -0.285218
01 01 2018 00:30    0.277902
01 01 2018 00:40   -0.048479
01 01 2018 00:50    0.015605
                      ...
28 09 2018 20:20   -0.153451
28 09 2018 20:50    2.015180
28 09 2018 21:00   -0.364334
28 09 2018 21:10   -1.955262
28 09 2018 21:20    1.185747
Name: Theoretical_Power_Curve (KWh), Length: 30983, dtype: float64
```

```
x_train_log_diff.dropna(inplace = True)
test_stationarity(x_train_log_diff.dropna())
```

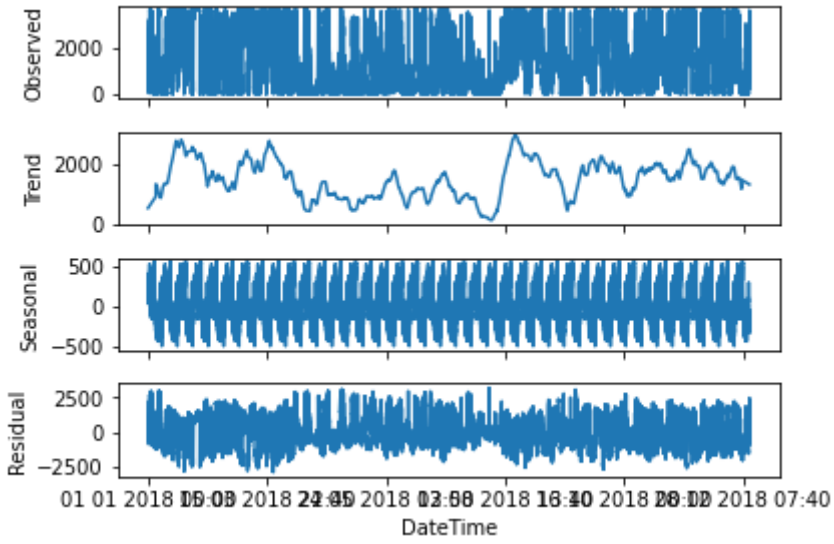



```
Results of Dickey-Fuller Test:
Test Statistic                  -32.458206
p-value                           0.000000
#Lags Used                       42.000000
Number of Observations Used    30940.000000
Critical Value (1%)               -3.430561
Critical Value (5%)               -2.861633
Critical Value (10%)              -2.566820
dtype: float64
```

```
ts.head()
```

DateTime	Theoretical_Power_Curve (KWh)
01 01 2018 00:00	416.328908
01 01 2018 00:10	519.917511
01 01 2018 00:20	390.900016
01 01 2018 00:30	516.127569
01 01 2018 00:40	491.702972

```
import statsmodels.api as sm

decomposition = sm.tsa.seasonal_decompose(ts['Theoretical_Power_Curve (KWh)'], model='addi
                          extrapolate_trend='freq', freq=1300) #additive or multiplicati
fig = decomposition.plot()
plt.show()
```




```
!pip install pyramid
```

```
Requirement already satisfied: pyramid in /usr/local/lib/python3.6/dist-packages (1.1
Requirement already satisfied: zope.interface>=3.8.0 in /usr/local/lib/python3.6/dist
Requirement already satisfied: plaster-pastedeploy in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: plaster in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: hupper>=1.5 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: webob>=1.8.3 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: translationstring>=0.4 in /usr/local/lib/python3.6/dis
Requirement already satisfied: venusian>=1.0 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: zope.deprecation>=3.5.0 in /usr/local/lib/python3.6/di
Requirement already satisfied: PasteDeploy>=2.0 in /usr/local/lib/python3.6/dist-pack
```

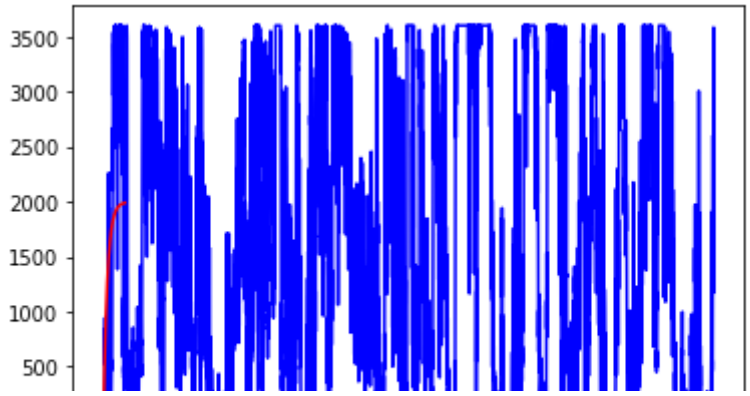
```
!pip install pmdarima
```

```
Requirement already satisfied: pmdarima in /usr/local/lib/python3.6/dist-packages (1
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: urllib3 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: statsmodels>=0.11 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: Cython<0.29.18,>=0.29 in /usr/local/lib/python3.6/dist
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dis
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (fr
```

```
import pmdarima as pm
stepwise_model = pm.auto_arima(ts_train,start_p=0,start_q=0,max_p=10,max_q=5,m=10,start_P=
                               seasonal=True,d=1,D=1,trace=True,error_action='ignore',suppres
```

```
 plt.plot(ts_test, c="blue")
plt.plot(forecasts1, c= "red")
```

```
[<matplotlib.lines.Line2D at 0x7f36b882ba20>]
```



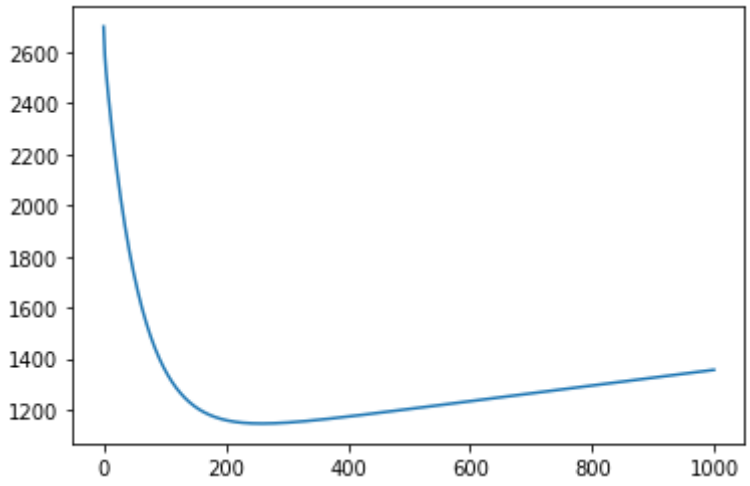

```
import pmdarima as pm
stepwise_model = pm.auto_arima(ts,start_p=0,start_q=0,max_p=10,max_q=10,m=1,h=1,stationary
                               seasonal=True,d=1,D=1,trace=True,error_action='ignore',suppres


forecasts = stepwise_model.predict(n_periods= 450)
#plt.plot(forecasts)
forecasts
```

```
array([2699.11522287, 2632.1769497 , 2586.72049427, 2552.22730231,
       2522.95496392, 2496.26007868, 2470.95322687, 2446.49911965,
       2422.65483465, 2399.30801908, 2376.40465853, 2353.91683967,
       2331.82836268, 2310.12831942, 2288.80822652, 2267.86074486,
       2247.27910709, 2227.05686104, 2207.18775427, 2187.66568145,
       2168.48465992, 2149.63881782, 2131.12238776, 2112.92970315,
       2095.05519552, 2077.4933925 , 2060.23891597, 2043.28648038,
       2026.6308911 , 2010.26704284, 1994.18991814, 1978.39458581,
       1962.87619951, 1947.6299962 , 1932.65129479, 1917.93549468,
       1903.47807441, 1889.27459027, 1875.32067498, 1861.61203635,
       1848.14445604, 1834.91378823, 1821.91595843, 1809.14696221,
       1796.60286399, 1784.27979591, 1772.17395661, 1760.28161011,
       1748.5990847 , 1737.12277178, 1725.84912485, 1714.77465838,
       1703.8959468 , 1693.20962345, 1682.71237956, 1672.40096328,
       1662.27217867, 1652.32288479, 1642.54999467, 1632.95047447,
       1623.5213425 , 1614.25966838, 1605.1625721 , 1596.22722319,
       1587.45083987, 1578.83068818, 1570.3640812 , 1562.04837822,
       1553.88098393, 1545.85934767, 1537.98096264, 1530.24336516,
       1522.6441339 , 1515.18088921, 1507.85129232, 1500.65304473,
       1493.58388744, 1486.64160031, 1479.82400137, 1473.12894621,
       1466.55432726, 1460.09807323, 1453.75814842, 1447.53255218,
       1441.41931823, 1435.41651413, 1429.52224067, 1423.7346313 ,
       1418.05185157, 1412.47209859, 1406.99360046, 1401.61461577,
       1396.33343305, 1391.14837027, 1386.05777432, 1381.06002052,
       1376.15351216, 1371.33667995, 1366.60798161, 1361.96590138,
       1357.40894958, 1352.93566214, 1348.54460017, 1344.23434952,
       1340.00352037, 1335.85074681, 1331.7746864 , 1327.77401981,
       1323.84745039, 1319.99370378, 1316.21152756, 1312.49969083,
       1308.85698386, 1305.28221775, 1301.77422402, 1298.33185428,
       1294.95397993, 1291.63949175, 1288.38729958, 1285.19633206,
       1282.06553621, 1278.99387718, 1275.98033792, 1273.02391886,
       1270.12363762, 1267.27852873, 1264.48764332, 1261.75004883,
       1259.06482874, 1256.4310823 , 1253.84792425, 1251.31448455,
       1248.82990813, 1246.39335462, 1244.0039981 , 1241.66102687,
       1239.36364317, 1237.11106298, 1234.90251573, 1232.73724416,
       1230.61450397, 1228.5335637 , 1226.49370447, 1224.49421974,
       1222.53441514, 1220.61360824, 1218.73112834, 1216.88631629,
       1215.07852424, 1213.30711553, 1211.57146441, 1209.87095593,
       1208.20498567, 1206.57295964, 1204.97429406, 1203.40841518,
       1201.87475912, 1200.37277171, 1198.90190829, 1197.46163358,
       1196.05142152, 1194.67075506, 1193.31912608, 1191.99603517,
       1190.70099153, 1189.4335128 , 1188.1931249 , 1186.97936191,
       1185.79176593, 1184.62988693, 1183.49328262, 1182.38151832,
       1181.29416682, 1180.23080825, 1179.19102999, 1178.17442648,
       1177.18059917, 1176.20915633, 1175.25971299, 1174.3318908 ,
       1173.42531791, 1172.53962886, 1171.67446449, 1170.82947181,
       1170.00430389, 1169.19861978, 1168.4120844 , 1167.64436842,
       1166.89514817, 1166.16410556, 1165.45092796, 1164.75530812,
       1164.07694409, 1163.41553908, 1162.77080143, 1162.14244449,
       1161.53018654, 1160.93375071, 1160.35286487, 1159.78726159,
       1159.23667804, 1158.70085591, 1158.17954131, 1157.67248476,
       1157.17944102, 1156.70016911, 1156.23443217, 1155.78199742,
       1155.34263608, 1154.91612332, 1154.50223816, 1154.10076343,
       1153.71148568, 1153.33419516, 1152.96868569, 1152.61475467,
       1152.27220296, 1151.94083486, 1151.62045802, 1151.31088343,
       1151.01192529, 1150.72340102, 1150.44513118, 1150.1769394 ,
       1149.91865236, 1149.67009972, 1149.43111405, 1149.20153083,
       1148.98118835, 1148.76992767, 1148.56759262, 1148.37402967,
       1148.18908798, 1148.01261925, 1147.84447779, 1147.68452036,
       1147.53260622, 1147.38859703, 1147.25235685, 1147.12375206,
       1147.00265134, 1146.88892563, 1146.78244811, 1146.6830941 ,
```

```
1146.5907411 , 1146.50526871, 1146.42655858, 1146.35449442,
1146.28896193, 1146.22984877, 1146.17704454, 1146.13044074,
1146.08993073, 1146.05540971, 1146.02677467, 1146.0039244 ,
1145.98675939, 1145.97518187, 1145.96909576, 1145.96840659,
1145.97302156, 1145.98284945, 1145.99780059, 1146.01778688,
1146.04272172, 1146.07252001, 1146.10709809, 1146.14637378,
1146.19026626, 1146.23869615, 1146.29158541, 1146.34885735,
1146.41043658, 1146.47624904, 1146.54622191, 1146.62028364,
1146.69836392, 1146.78039363, 1146.86630484, 1146.9560308 ,
1147.04950591, 1147.14666568, 1147.24744675, 1147.35178683,
1147.45962472, 1147.57090025, 1147.68555431, 1147.80352879,
1147.92476659, 1148.04921158, 1148.1768086 , 1148.30750344,
1148.44124283, 1148.57797441, 1148.71764671, 1148.86020916,
1149.00561205, 1149.15380654, 1149.3047446 , 1149.45837906,
1149.61466355, 1149.77355248, 1149.93500106, 1150.09896528,
```

```
forecasts = stepwise_model.predict(n_periods= 1000)
plt.plot(forecasts)
```

```
[<matplotlib.lines.Line2D at 0x7f886491d240>]
```




```
1168.82465955, 1169.09771087, 1169.37137629, 1169.64561856,
```

```
forecasts = stepwise_model.predict(n_periods= 10000)
plt.plot(forecasts)
```

```
[<matplotlib.lines.Line2D at 0x7f886cc2f198>]
```