Pollution SARIMAX TimeSeries Forecasting

May 28, 2019

```
In [1]: !pip install --user xlrd
```

```
Requirement already satisfied: xlrd in /home/omar/.local/lib/python3.5/site-packages (1.2.0)
You are using pip version 19.0.3, however version 19.1.1 is available.You should consider upgrad
```

```
In [2]: !ls
```

```
ARIMA TimeSeries Forecasting-Copy2.ipynb
ARIMA TimeSeries Forecasting-Copy3.ipynb
ARIMA TimeSeries Forecasting.ipynb
ARIMA TimeSeries Forecasting.pdf
cleaned_data
eda and time series prediction.ipynb
Historical Pollen Index
Historical Pollution Index
Historical Symptom Logs
__MACOSX
method_1_ARIMA_timeseries_prediction_sample.ipynb
pollen.csv
pollen_history.csv
SARIMAX Pollen TimeSeries Forecasting.ipynb
SARIMAX Pollution TimeSeries Forecasting.ipynb
symptom_cause_2019051616-03_001.csv
weatherStored_2019051615-59_001.csv
```

```
In [3]: !ls cleaned_data/
```

```
Pollen_full_data_Eve.xlsx     Symptom Cause Eve.xlsx
Pollution full data Eve.xlsx  Symptom log full data Eve.xlsx
~$Symptom Cause Eve.xlsx
```

```
In [4]: import pandas as pd
        from datetime import datetime

        import plotly
```

```
plotly.offline.init_notebook_mode()

import plotly.plotly as py
import plotly.graph_objs as go

from statsmodels.tsa.stattools import adfuller
from numpy import log

import itertools
import numpy as np
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")

plt.style.use('fivethirtyeight')
import pandas as pd
import statsmodels.api as sm
import matplotlib
matplotlib.rcParams['axes.labelsize'] = 14
matplotlib.rcParams['xtick.labelsize'] = 12
matplotlib.rcParams['ytick.labelsize'] = 12
matplotlib.rcParams['text.color'] = 'k'
```

In [5]:
```
data = pd.read_excel('cleaned_data/Pollution full data Eve.xlsx')
```

In [6]:
```
list(data)
```

Out[6]:
```
['city',
 'date_time',
 'date_time.1',
 'pollutantGlobalIndex',
 'so2Index',
 'no2Index',
 'o3Index',
 'coIndex',
 'pm25Index',
 'pm10Index']
```

In [7]:
```
data.head()
```

Out[7]:
```
     city           date_time         date_time.1  pollutantGlobalIndex  \
0  Austin 2017-09-06 14:00:00 2017-09-06 14:00:00                     3
1  Austin 2017-09-06 15:00:00 2017-09-06 15:00:00                     3
2  Austin 2017-09-06 16:00:00 2017-09-06 16:00:00                     3
3  Austin 2017-09-06 17:00:00 2017-09-06 17:00:00                     3
4  Austin 2017-09-06 18:00:00 2017-09-06 18:00:00                     3

   so2Index  no2Index  o3Index  coIndex  pm25Index  pm10Index
```

```
0          1          1         3       NaN           3          NaN
1          1          1         3       NaN           3          NaN
2          1          1         3       NaN           3          NaN
3          1          1         3       NaN           2          NaN
4          1          1         3       NaN           2          NaN
```

```
In [8]: data = data[['date_time', 'pollutantGlobalIndex']]
```

```
In [9]: data.head()
```

```
Out[9]:             date_time  pollutantGlobalIndex
        0 2017-09-06 14:00:00                     3
        1 2017-09-06 15:00:00                     3
        2 2017-09-06 16:00:00                     3
        3 2017-09-06 17:00:00                     3
        4 2017-09-06 18:00:00                     3
```

```
In [11]: data.dropna()
```

```
Out[11]:                 date_time  pollutantGlobalIndex
         0      2017-09-06 14:00:00                     3
         1      2017-09-06 15:00:00                     3
         2      2017-09-06 16:00:00                     3
         3      2017-09-06 17:00:00                     3
         4      2017-09-06 18:00:00                     3
         5      2017-09-06 19:00:00                     3
         6      2017-09-06 20:00:00                     3
         7      2017-09-06 21:00:00                     3
         8      2017-09-06 22:00:00                     3
         9      2017-09-06 23:00:00                     3
         10     2017-09-07 00:00:00                     3
         11     2017-09-07 01:00:00                     3
         12     2017-09-07 02:00:00                     4
         13     2017-09-07 03:00:00                     4
         14     2017-09-07 04:00:00                     4
         15     2017-09-07 05:00:00                     4
         16     2017-09-07 06:00:00                     4
         17     2017-09-07 07:00:00                     4
         18     2017-09-07 08:00:00                     4
         19     2017-09-07 09:00:00                     4
         20     2017-09-07 10:00:00                     3
         21     2017-09-07 11:00:00                     3
         22     2017-09-07 12:00:00                     3
         23     2017-09-07 13:00:00                     3
         24     2017-09-07 14:00:00                     3
         25     2017-09-07 15:00:00                     3
         26     2017-09-07 16:00:00                     3
         27     2017-09-07 17:00:00                     2
         28     2017-09-07 18:00:00                     2
```

```
29    2017-09-07 19:00:00                   2
...                   ...                 ...
26244 2019-05-20 07:00:00                   6
26245 2019-05-20 08:00:00                   6
26246 2019-05-20 09:00:00                   6
26247 2019-05-20 10:00:00                   6
26248 2019-05-20 11:00:00                   6
26249 2019-05-20 12:00:00                   7
26250 2019-05-20 13:00:00                   7
26251 2019-05-20 14:00:00                   7
26252 2019-05-20 15:00:00                   7
26253 2019-05-20 16:00:00                   8
26254 2019-05-20 17:00:00                   9
26255 2019-05-20 18:00:00                   8
26256 2019-05-20 19:00:00                   8
26257 2019-05-20 20:00:00                   7
26258 2019-05-20 21:00:00                   7
26259 2019-05-20 22:00:00                   7
26260 2019-05-20 23:00:00                   7
26261 2019-05-21 00:00:00                   6
26262 2019-05-21 01:00:00                   4
26263 2019-05-21 02:00:00                   4
26264 2019-05-21 03:00:00                   4
26265 2019-05-21 04:00:00                   4
26266 2019-05-21 05:00:00                   6
26267 2019-05-21 06:00:00                   4
26268 2019-05-21 07:00:00                   7
26269 2019-05-21 08:00:00                   6
26270 2019-05-21 09:00:00                   7
26271 2019-05-21 10:00:00                   7
26272 2019-05-21 11:00:00                   7
26273 2019-05-21 12:00:00                   7

[26274 rows x 2 columns]
```

print(data.index.min() + " " + data.index.max())
a = data['date_time'].astype(str).str.split(" ", n = 1, expand = True)
data['date'] = a[0]

```
In [12]: data.head()
```

```
Out[12]:             date_time  pollutantGlobalIndex
         0 2017-09-06 14:00:00                     3
         1 2017-09-06 15:00:00                     3
         2 2017-09-06 16:00:00                     3
         3 2017-09-06 17:00:00                     3
         4 2017-09-06 18:00:00                     3
```

```
In [14]: data = data.groupby('date_time')['pollutantGlobalIndex'].sum().reset_index()
```

```
In [15]: data.head(20)
```

```
Out[15]:               date_time  pollutantGlobalIndex
         0   2017-09-06 14:00:00                     3
         1   2017-09-06 15:00:00                     3
         2   2017-09-06 16:00:00                     3
         3   2017-09-06 17:00:00                     3
         4   2017-09-06 18:00:00                     3
         5   2017-09-06 19:00:00                     3
         6   2017-09-06 20:00:00                     3
         7   2017-09-06 21:00:00                     3
         8   2017-09-06 22:00:00                     3
         9   2017-09-06 23:00:00                     3
         10  2017-09-07 00:00:00                     3
         11  2017-09-07 01:00:00                     3
         12  2017-09-07 02:00:00                     4
         13  2017-09-07 03:00:00                     4
         14  2017-09-07 04:00:00                     4
         15  2017-09-07 05:00:00                     4
         16  2017-09-07 06:00:00                     4
         17  2017-09-07 07:00:00                     4
         18  2017-09-07 08:00:00                     4
         19  2017-09-07 09:00:00                     4
```

data['2017-09-07'].max()

```
In [18]: type(data)
```

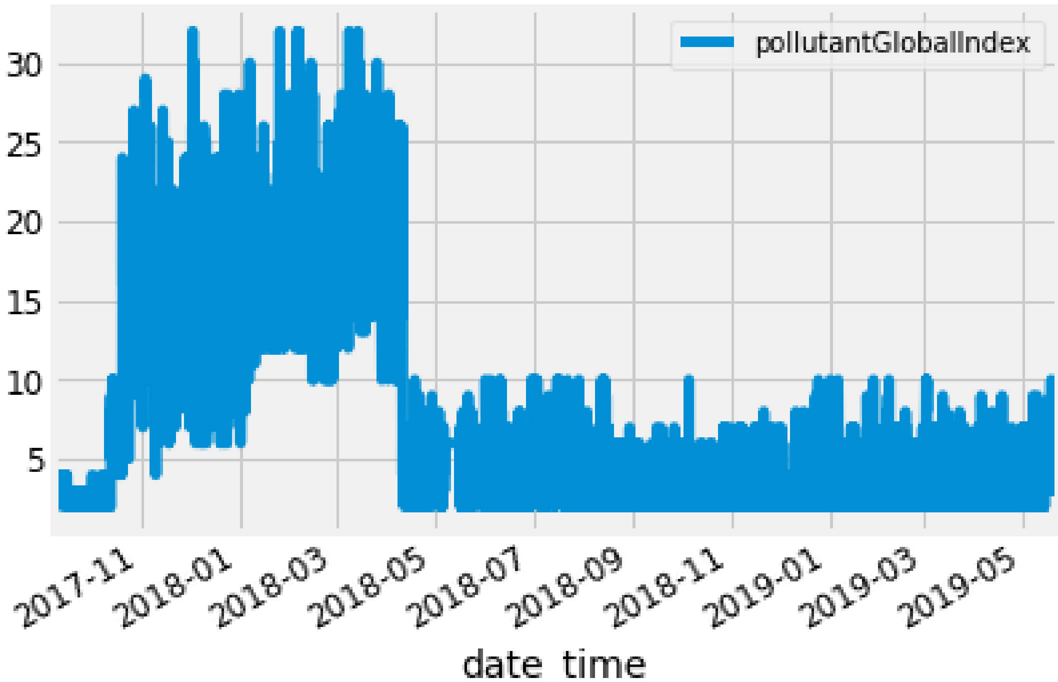
```
Out[18]: pandas.core.frame.DataFrame
```

```
In [19]: data.set_index('date_time', inplace=True)
```

```
In [20]: data.index = pd.to_datetime(data.index)
```

```
In [21]: data.plot()
```

```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1a9ff1a2b0>
```



We will use a "grid search" to iteratively explore different combinations of parameters. For each combination of parameters, we fit a new seasonal ARIMA model with the SARIMAX() function from the statsmodels module and assess its overall quality. Once we have explored the entire landscape of parameters, our optimal set of parameters will be the one that yields the best performance for our criteria of interest. Let's begin by generating the various combination of parameters that we wish to assess:

```
In [22]: # Define the p, d and q parameters to take any value between 0 and 2
         p = d = q = range(0, 2)

         # Generate all different combinations of p, q and q triplets
         pdq = list(itertools.product(p, d, q))

         # Generate all different combinations of seasonal p, q and q triplets
         seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

         print('Examples of parameter combinations for Seasonal ARIMA...')
         print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
         print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
         print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
         print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
```

```
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

```
In [23]: for param in pdq:
             for param_seasonal in seasonal_pdq:
                 try:
                     mod = sm.tsa.statespace.SARIMAX(y,
                                                     order=param,
                                                     seasonal_order=param_seasonal,
                                                     enforce_stationarity=False,
                                                     enforce_invertibility=False)

                     results = mod.fit()

                     print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
                 except:
                     continue
```

```
In [24]: y = data
```

The code below iterates through combinations of parameters and uses the SARIMAX function from statsmodels to fit the corresponding Seasonal ARIMA model. Here, the order argument specifies the (p, d, q) parameters, while the seasonal_order argument specifies the (P, D, Q, S) seasonal component of the Seasonal ARIMA model. After fitting each SARIMAX()model, the code prints out its respective AIC score.

The lesser the AIC score, the better model architecture it is.

Akaike's Information Criterion (AIC): Formally, AIC is defined as 2log+2 where is the maximized log likelihood and is the number of parameters in the model. For the normal regression problem, AIC is an estimate of the Kullback-Leibler discrepancy between a true model and a candidate model.

```
In [25]: for param in pdq:
             for param_seasonal in seasonal_pdq:

                 mod = sm.tsa.statespace.SARIMAX(y,
                                                 order=param,
                                                 seasonal_order=param_seasonal,
                                                 enforce_stationarity=False,
                                                 enforce_invertibility=False)

                 results = mod.fit()

                 print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:106687.52803499259
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:95806.76451832935
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:77719.46114448212
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:72595.83401805462
```

```
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:77227.17792550023
ARIMA(0, 0, 0)x(1, 0, 1, 12)12 - AIC:72622.20617441263
ARIMA(0, 0, 0)x(1, 1, 0, 12)12 - AIC:73828.76874015582
ARIMA(0, 0, 0)x(1, 1, 1, 12)12 - AIC:72436.46745807579
ARIMA(0, 0, 1)x(0, 0, 0, 12)12 - AIC:90587.40701959425
ARIMA(0, 0, 1)x(0, 0, 1, 12)12 - AIC:82460.52020150796
ARIMA(0, 0, 1)x(0, 1, 0, 12)12 - AIC:70112.29216662335
ARIMA(0, 0, 1)x(0, 1, 1, 12)12 - AIC:64026.149056888775
ARIMA(0, 0, 1)x(1, 0, 0, 12)12 - AIC:69393.15984711159
ARIMA(0, 0, 1)x(1, 0, 1, 12)12 - AIC:64053.58289285139
ARIMA(0, 0, 1)x(1, 1, 0, 12)12 - AIC:66009.8563356791
ARIMA(0, 0, 1)x(1, 1, 1, 12)12 - AIC:63922.36949658996
ARIMA(0, 1, 0)x(0, 0, 0, 12)12 - AIC:56126.59866557239
ARIMA(0, 1, 0)x(0, 0, 1, 12)12 - AIC:56091.320396223324
ARIMA(0, 1, 0)x(0, 1, 0, 12)12 - AIC:65918.84910609439
ARIMA(0, 1, 0)x(0, 1, 1, 12)12 - AIC:56117.767094150986
ARIMA(0, 1, 0)x(1, 0, 0, 12)12 - AIC:56093.867617736585
ARIMA(0, 1, 0)x(1, 0, 1, 12)12 - AIC:56067.39256360466
ARIMA(0, 1, 0)x(1, 1, 0, 12)12 - AIC:61173.373579326944
ARIMA(0, 1, 0)x(1, 1, 1, 12)12 - AIC:56119.73748484236
ARIMA(0, 1, 1)x(0, 0, 0, 12)12 - AIC:55599.74150340479
ARIMA(0, 1, 1)x(0, 0, 1, 12)12 - AIC:55565.48645786028
ARIMA(0, 1, 1)x(0, 1, 0, 12)12 - AIC:65416.96440023401
ARIMA(0, 1, 1)x(0, 1, 1, 12)12 - AIC:55585.682663974265
ARIMA(0, 1, 1)x(1, 0, 0, 12)12 - AIC:55570.91907491916
ARIMA(0, 1, 1)x(1, 0, 1, 12)12 - AIC:55519.595768934116
ARIMA(0, 1, 1)x(1, 1, 0, 12)12 - AIC:60535.490496210725
ARIMA(0, 1, 1)x(1, 1, 1, 12)12 - AIC:55587.66450202813
ARIMA(1, 0, 0)x(0, 0, 0, 12)12 - AIC:56020.82034273728
ARIMA(1, 0, 0)x(0, 0, 1, 12)12 - AIC:55982.410360225986
ARIMA(1, 0, 0)x(0, 1, 0, 12)12 - AIC:64227.05133302646
ARIMA(1, 0, 0)x(0, 1, 1, 12)12 - AIC:55348.879253294566
ARIMA(1, 0, 0)x(1, 0, 0, 12)12 - AIC:55981.549733660904
ARIMA(1, 0, 0)x(1, 0, 1, 12)12 - AIC:55373.61039903519
ARIMA(1, 0, 0)x(1, 1, 0, 12)12 - AIC:59587.17551528601
ARIMA(1, 0, 0)x(1, 1, 1, 12)12 - AIC:55350.87204135413
ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:55535.51708667213
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:55499.60586893518
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:64150.15088835024
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:55102.36067543709
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:55501.85792065177
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:55124.72102552027
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:59444.75120010918
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:55104.36054931962
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:55641.03146139863
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:55606.35577679375
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:65442.34636216519
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:55628.64166917707
```

```
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:55606.02325671892
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:55565.64356661648
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:60600.26351396194
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:55630.6163411032
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:54951.932862377056
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:54913.5231834527
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:64231.060852482886
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:54945.63165784913
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:54915.94299261454
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:54889.8878570757
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:59595.613022084566
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:54947.548306623394
```

In [26]:

```
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7840      0.004    183.397      0.000       0.776       0.792
ma.L1         -1.0366      0.002   -452.991      0.000      -1.041      -1.032
ar.S.L12       0.0003      0.006      0.050      0.960      -0.011       0.012
ma.S.L12      -0.9980      0.001   -738.487      0.000      -1.001      -0.995
sigma2         2.4233      0.015    157.330      0.000       2.393       2.453
==============================================================================
```

In [27]:

```
results.plot_diagnostics(figsize=(15, 12))
plt.show()
```