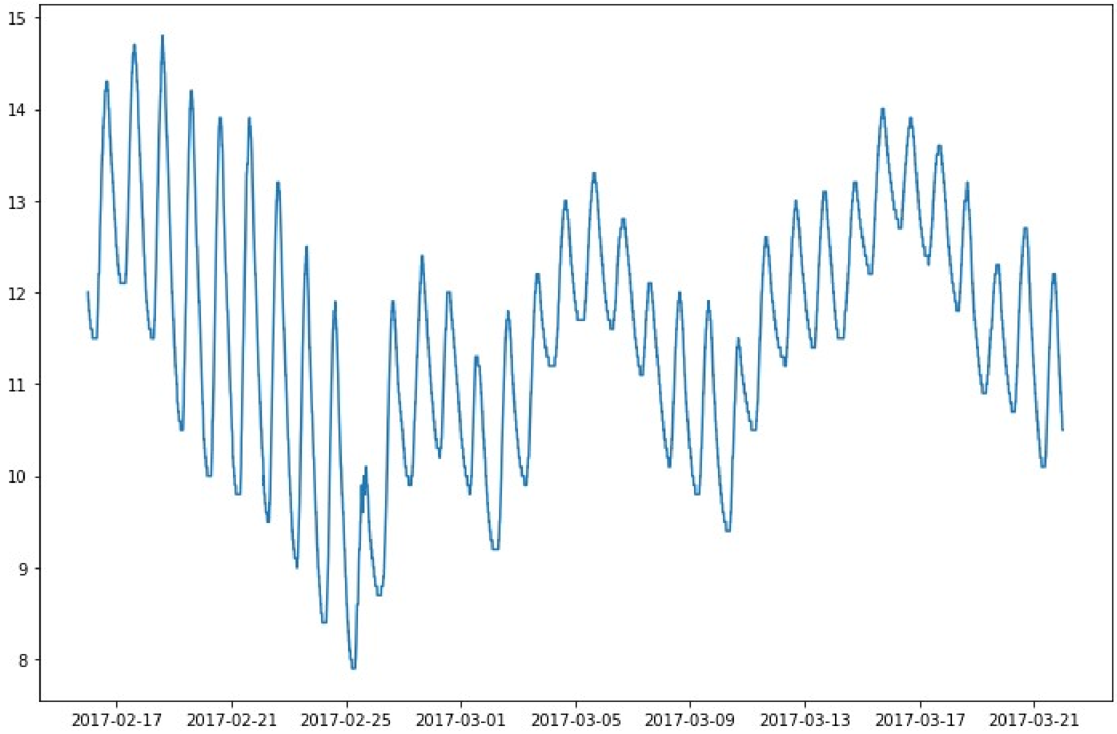
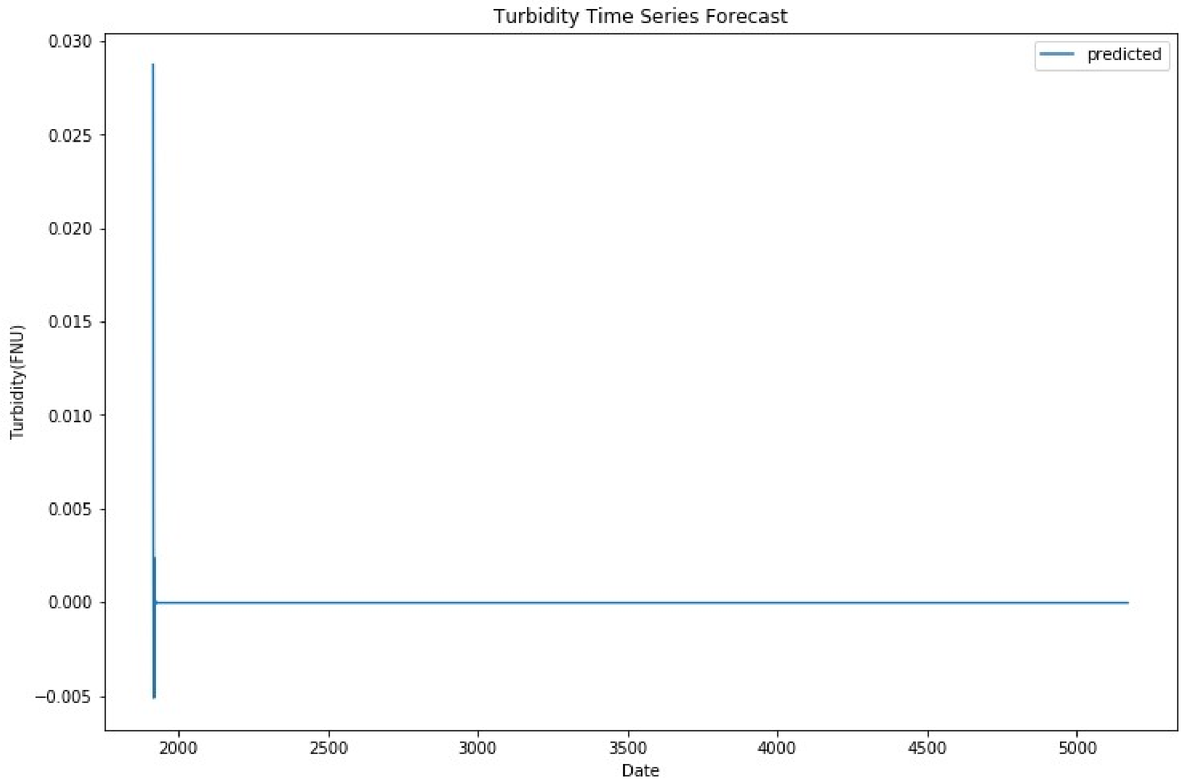
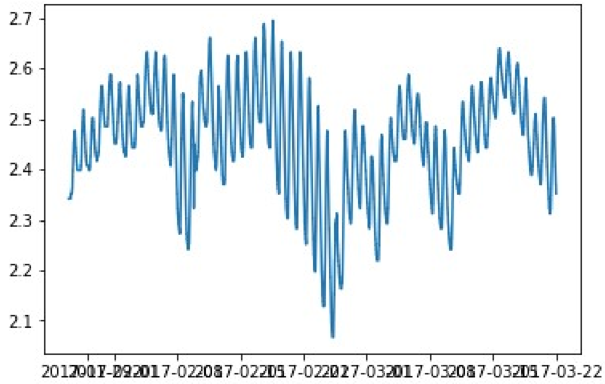
```
    model = ARIMA(ts, order=order) # (ARMA) = (p,d,q)

  File "C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py", line
996, in __new__
    mod.__init__(endog, order, exog, dates, freq, missing)

  File "C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py", line
1014, in __init__
    raise ValueError("d > 2 is not supported")

ValueError: d > 2 is not supported


In [100]:

In [100]: arima_model(do_train,(2,2,1))
     ...:
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:225:
ValueWarning: A date index has been provided, but it has no associated frequency
information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:225:
ValueWarning: A date index has been provided, but it has no associated frequency
information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:531:
ValueWarning: No supported index is available. Prediction results will be given with an
integer index beginning at `start`.
  ValueWarning)
```





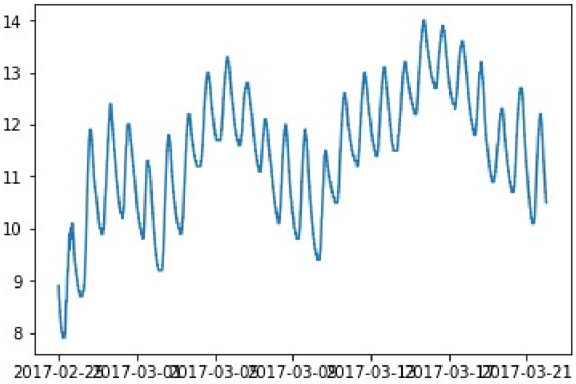
```
In [101]: plt.plot(do_logScale)
Out[101]: [<matplotlib.lines.Line2D at 0x16fb4e9e630>]
```



```
In [102]: def input_data():
     ...:     def parser(x):
     ...:         return datetime.strptime(x,'%Y-%m-%d %H:%M')
     ...:     dataset = pd.read_csv('Data8.csv',header=0, delimiter=',',index_col=0, 
parse_dates=[0], date_parser=parser)
     ...:     dataset = dataset.fillna(method ='pad')
     ...:     do = dataset.filter(['DO(mg/L)'], axis=1)
     ...:     train_size,test_size = 1920, 3252
     ...:     do_train,do_test = tts(do,test_size = test_size, random_state=0, 
shuffle=False)
     ...:     #dataset.fillna(method ='bfill')

In [103]: input_data()
Traceback (most recent call last):
  File "<ipython-input-103-ed564c6cbd08>", line 1, in <module>
    input_data()

  File "<ipython-input-102-8cf998e3fabf>", line 8, in input_data
    do_train,do_test = tts(do,test_size = test_size, random_state=0, shuffle=False)

  File "C:\Users\admin\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py",
line 2194, in train_test_split
    train_size)

  File "C:\Users\admin\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py",
line 1829, in _validate_shuffle_split
    'samples %d' % (test_size, n_samples))

ValueError: test_size=3252 should be smaller than the number of samples 2396


In [104]:

In [104]:

Removing all variables...
```

```
In [104]: dataset = pd.read_csv('Data8.csv',header=0, delimiter=',',index_col=0,
parse_dates=[0], date_parser=parser)
Traceback (most recent call last):

  File "<ipython-input-104-9d37fc20a145>", line 1, in <module>
    dataset = pd.read_csv('Data8.csv',header=0, delimiter=',',index_col=0,
parse_dates=[0], date_parser=parser)

NameError: name 'pd' is not defined


In [105]:

In [105]: import numpy as np
     ...: import matplotlib.pyplot as mp
     ...: import pandas as pd
     ...: from pandas import datetime
     ...: from sklearn.preprocessing import Imputer
     ...: from pandas.plotting import autocorrelation_plot
     ...: from matplotlib import pyplot
     ...: from statsmodels.tsa.arima_model import ARIMA
     ...: from sklearn.model_selection import train_test_split as tts
     ...: from statsmodels.tsa.stattools import adfuller
     ...: from statsmodels.tsa.stattools import acf, pacf
     ...: from statsmodels.tsa.seasonal import seasonal_decompose
     ...: import matplotlib.pylab as plt #for visualization

In [106]: def parser(x):
     ...:     return datetime.strptime(x,'%Y-%m-%d %H:%M')
     ...:
```

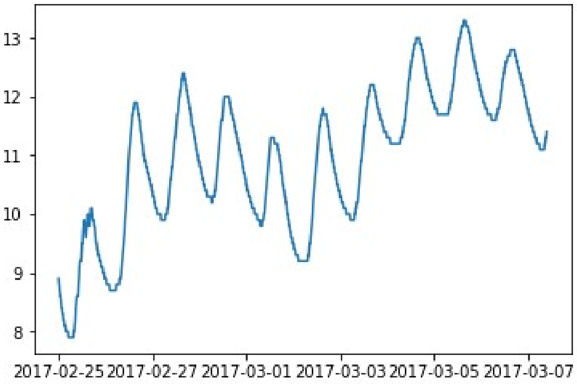
```
     ...: dataset = pd.read_csv('Data8.csv',header=0, delimiter=',',index_col=0,
parse_dates=[0], date_parser=parser)
     ...: dataset = dataset.fillna(method ='pad')
     ...: do = dataset.filter(['DO(mg/L)'], axis=1)

In [107]: datset.size
Traceback (most recent call last):

  File "<ipython-input-107-984f43d6dab5>", line 1, in <module>
    datset.size

NameError: name 'datset' is not defined


In [108]:

In [108]: dataset.size
Out[108]: 7188

In [109]: do.size
Out[109]: 2396

In [110]: plt.plot(do)
Out[110]: [<matplotlib.lines.Line2D at 0x16fba23b278>]
```



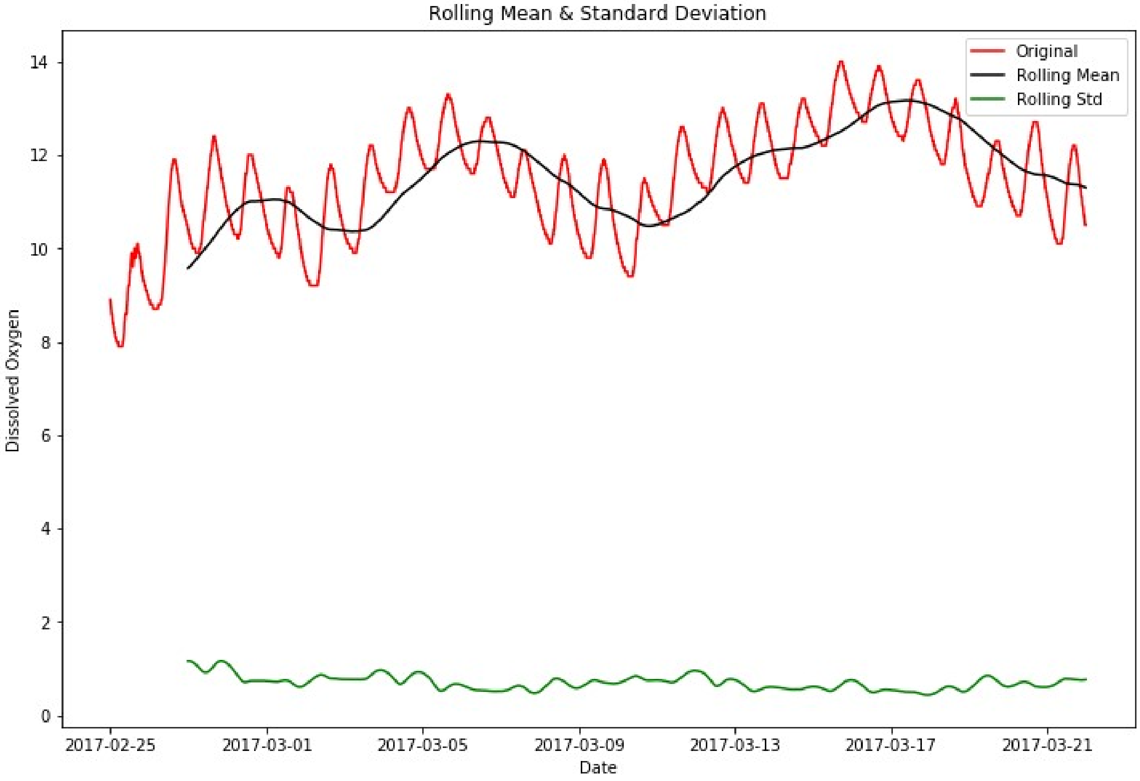
```
In [111]: input_data()
Traceback (most recent call last):

  File "<ipython-input-111-ed564c6cbd08>", line 1, in <module>
    input_data()

NameError: name 'input_data' is not defined


In [112]:

In [112]: def input_data():
     ...:     def parser(x):
     ...:         return datetime.strptime(x,'%Y-%m-%d %H:%M')
     ...:     dataset = pd.read_csv('Data8.csv',header=0, delimiter=',',index_col=0,
parse_dates=[0], date_parser=parser)
     ...:     dataset = dataset.fillna(method ='pad')
     ...:     do = dataset.filter(['DO(mg/L)'], axis=1)
     ...:     train_size,test_size = 1000, 1396
     ...:     do_train,do_test = tts(do,test_size = test_size, random_state=0,
shuffle=False)
     ...:     #dataset.fillna(method ='bfill')

In [113]: def check_adfuller(att):
     ...:
     ...: #Perform Augmented Dickey-Fuller test:
     ...:     print('Results of Dickey Fuller Test:')
     ...:     print("--------For a stationary time series Test statistic is less than
critical values-----------")
     ...:     dftest = adfuller(att, autolag='AIC')
     ...:
     ...:     dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags
Used','Number of Observations Used'])
     ...:     for key,value in dftest[4].items():
     ...:         dfoutput['Critical Value (%s)'%key] = value
     ...:
     ...:     print(dfoutput)

In [114]: def check_mean_std(ts, name):
     ...:
     ...:     rolmean = ts.rolling(window=192).mean()
     ...:     rolstd = ts.rolling(window=192).std()
     ...:     plt.figure(figsize=(12,8))
     ...:     print(name)
     ...:     orig = plt.plot(ts, color='red',label='Original')
     ...:     mean = plt.plot(rolmean, color='black', label='Rolling Mean')
     ...:     std = plt.plot(rolstd, color='green', label = 'Rolling Std')
     ...:     plt.xlabel("Date")
     ...:     plt.ylabel("Dissolved Oxygen")
     ...:     plt.title('Rolling Mean & Standard Deviation')
     ...:     plt.legend()
     ...:     plt.show()

In [115]: def acf_pacf_plots(dataset):
     ...:     ts_diff = dataset - dataset.shift()
     ...:     ts_diff.dropna(inplace=True)
     ...:     lag_acf = acf(ts_diff, nlags=20)
     ...:     lag_pacf = pacf(ts_diff, nlags=20, method='ols')
     ...:
     ...:     # ACF
     ...:     plt.figure(figsize=(22,10))
     ...:
     ...:     plt.subplot(121)
     ...:     plt.plot(lag_acf)
     ...:     plt.axhline(y=0,linestyle='--',color='gray')
     ...:     plt.axhline(y=-1.96/np.sqrt(len(ts_diff)),linestyle='--',color='gray')
     ...:     plt.axhline(y=1.96/np.sqrt(len(ts_diff)),linestyle='--',color='gray')
     ...:     plt.title('Autocorrelation Function')
     ...:
     ...:     # PACF
     ...:     plt.subplot(122)
     ...:     plt.plot(lag_pacf)
     ...:     plt.axhline(y=0,linestyle='--',color='gray')
```

```
   ...:     plt.axhline(y=-1.96/np.sqrt(len(ts_diff)),linestyle='--',color='gray')
   ...:     plt.axhline(y=1.96/np.sqrt(len(ts_diff)),linestyle='--',color='gray')
   ...:     plt.title('Partial Autocorrelation Function')
   ...:     plt.tight_layout()

In [116]: def arima_model(ts, order):
   ...:     # fit model
   ...:     model = ARIMA(ts, order=order) # (ARMA) = (p,d,q)
   ...:     model_fit = model.fit(disp=0)
   ...:
   ...:     # predict
   ...:     forecast = model_fit.predict(start=1919, end=5171)
   ...:
   ...:     # visualization
   ...:     plt.figure(figsize=(12,8))
   ...:     plt.plot(do_test,label = "original")
   ...:     plt.figure(figsize=(12,8))
   ...:     plt.plot(forecast,label = "predicted")
   ...:     plt.title("Turbidity Time Series Forecast")
   ...:     plt.xlabel("Date")
   ...:     plt.ylabel("Turbidity(FNU)")
   ...:     plt.legend()
   ...:     plt.show()

In [117]: def moving_average():
   ...:     do_logScale = np.log(do)
   ...:     plt.plot(do_logScale)
   ...:
   ...:     do_ma = do.rolling(window=192).mean() #window size 12 denotes 12 months,
giving rolling mean at yearly level
   ...:     #sc_movingSTD = sc_logScale.rolling(window=192).std()
   ...:
   ...:
   ...: #   plt.plot(sc_logScale)
   ...: #plt.plot(sc_moving_Average, color='blue')
   ...:
   ...:
   ...:     plt.figure(figsize=(12,8))
   ...:     plt.plot(do, color = "red",label = "Original")
   ...:     plt.plot(do_ma, color='black', label = "DO moving_avg_mean")
   ...:     plt.title("Dissolved Oxygen (mg/L) of Potomac River")
   ...:     plt.xlabel("Date")
   ...:     plt.ylabel("DO(mg/L)")
   ...:     plt.legend()
   ...:     plt.show()
   ...:
   ...:     do_ma_diff = do - do_ma
   ...:     #sc_LogScaleMinusMovingAverage.head(100)
   ...:
   ...:     do_ma_diff.dropna(inplace=True)
   ...:     #print(sc_rolmean,sc_rolstd)
   ...:
   ...:     check_adfuller(do_ma_diff['DO(mg/L)'])
   ...:     check_mean_std(do_ma_diff, 'Dissolved Oxygen(mg/L)')
```

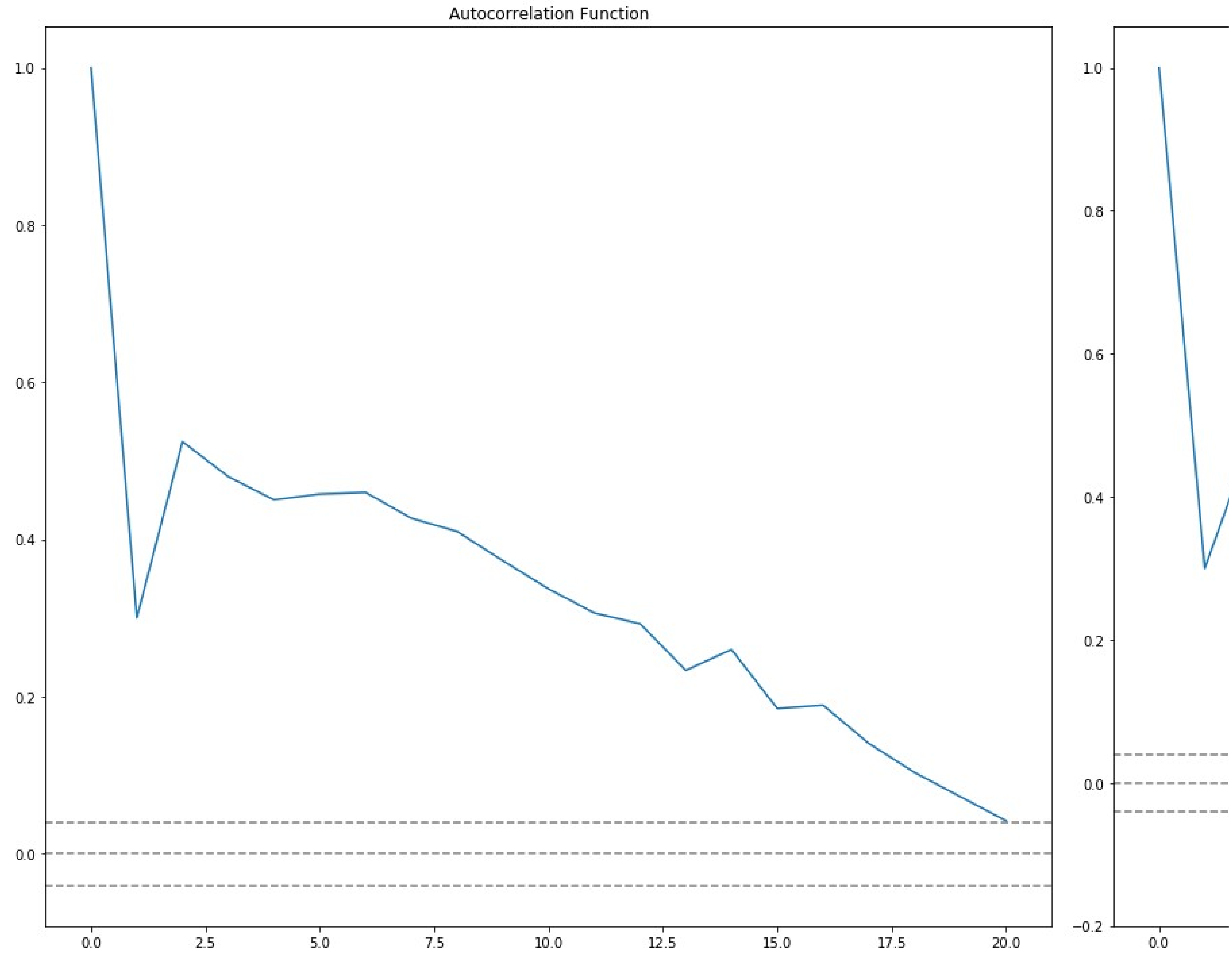
```
In [118]: input_data()

In [119]: train_size,test_size = 1000, 1396
     ...: do_train,do_test = tts(do,test_size = test_size, random_state=0, shuffle=False)

In [120]: plt.plot(do_train)
Out[120]: [<matplotlib.lines.Line2D at 0x16fb5291e10>]
```



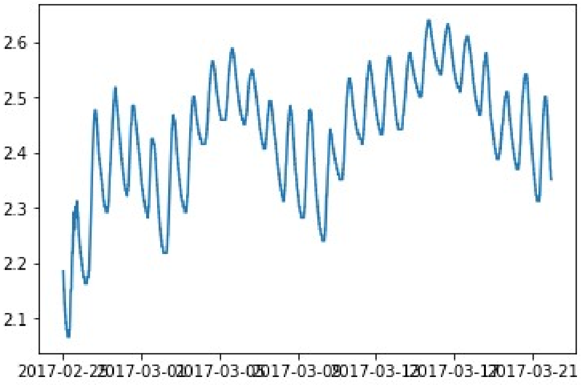
```
In [121]: check_adfuller(dataset['DO(mg/L)'])
     ...:
Results of Dickey Fuller Test:
--------For a stationary time series Test statistic is less than critical
values-----------
Test Statistic                  -4.000545
p-value                          0.001408
#Lags Used                      22.000000
Number of Observations Used   2373.000000
Critical Value (1%)             -3.433109
Critical Value (5%)             -2.862759
Critical Value (10%)            -2.567419
dtype: float64

In [122]: check_mean_std(dataset['DO(mg/L)'],'\n\nDissolved Oxygen')
     ...:


Dissolved Oxygen
```



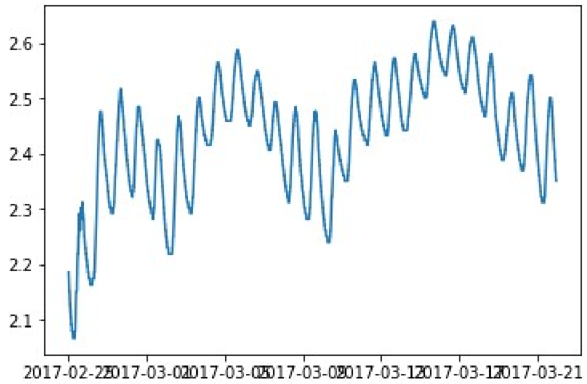
```
In [123]: acf_pacf_plots(do)
     ...:
```

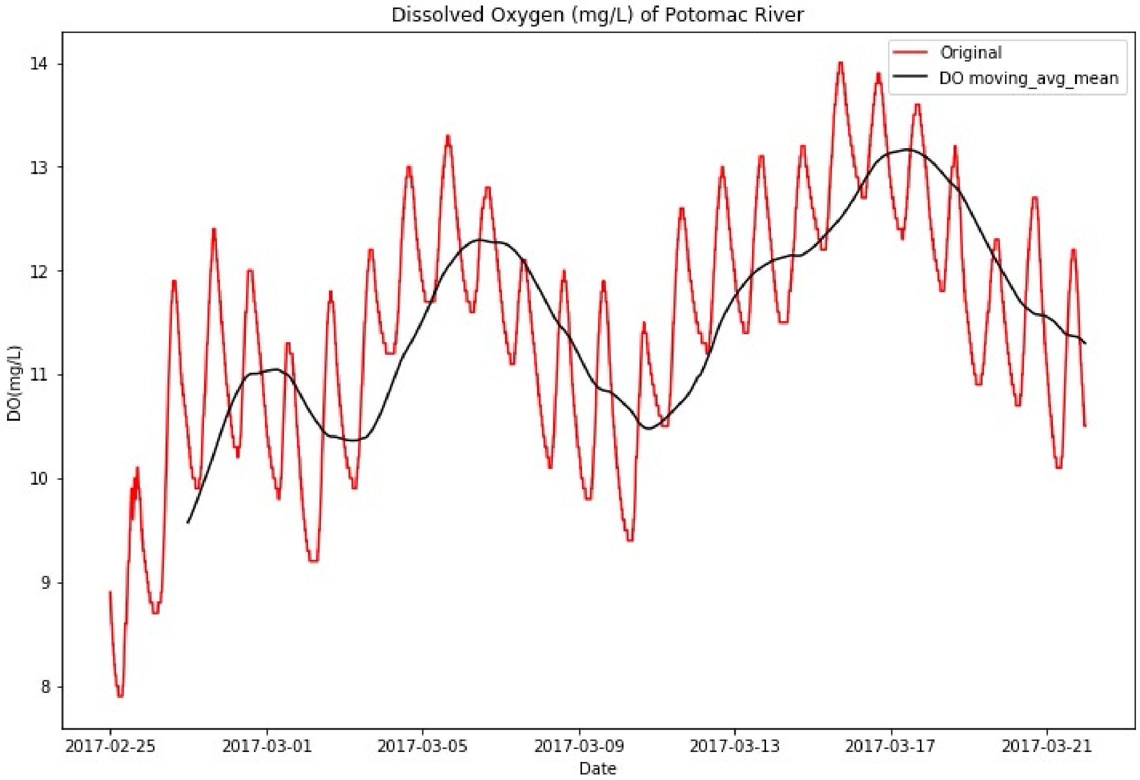


```
In [124]: do_logScale = np.log(do)
     ...: plt.plot(do_logScale)
Out[124]: [<matplotlib.lines.Line2D at 0x16fb5350a20>]
```

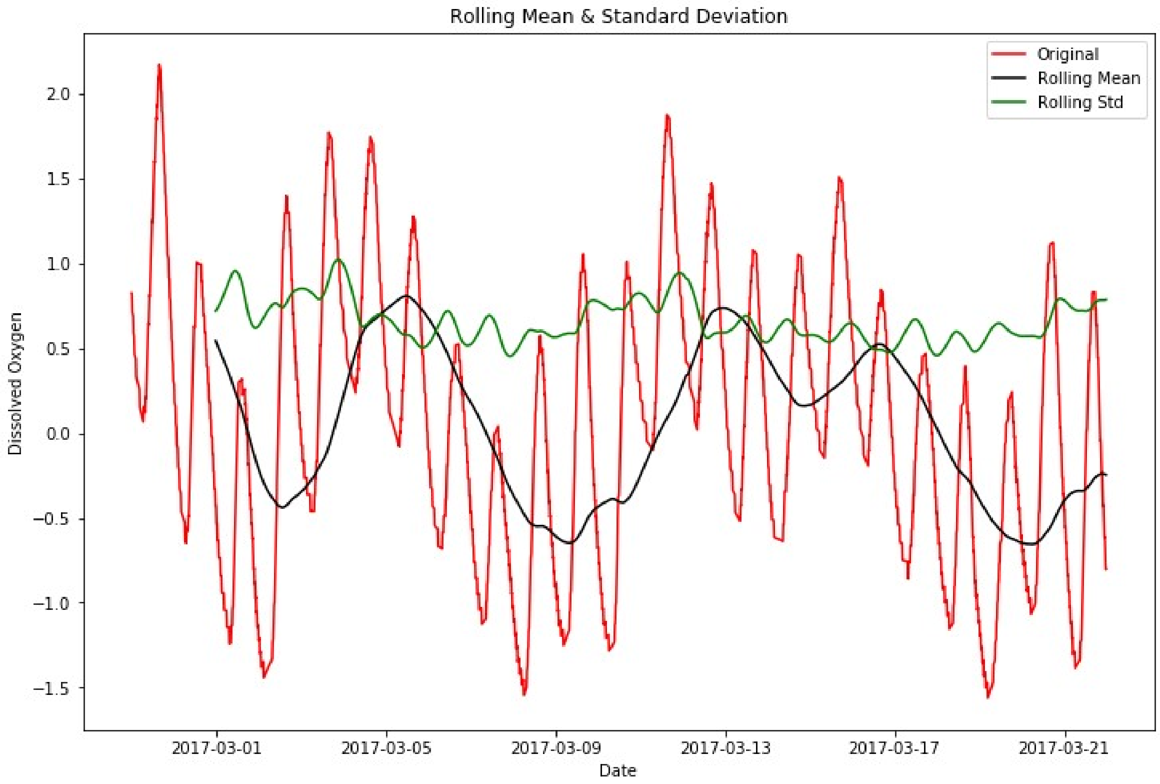


```
In [125]: moving_average()
     ...:
```



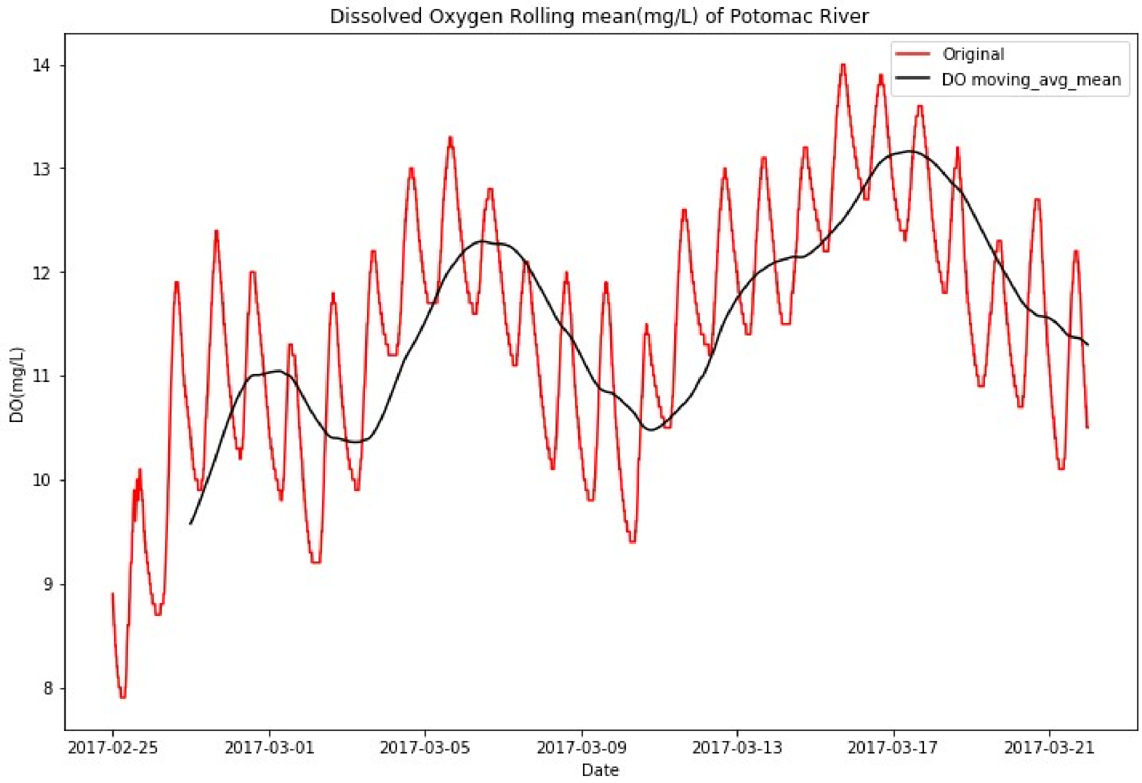


```
Results of Dickey Fuller Test:
--------For a stationary time series Test statistic is less than critical
values-----------
Test Statistic                -6.645701e+00
p-value                        5.272523e-09
#Lags Used                     1.500000e+01
Number of Observations Used    2.189000e+03
Critical Value (1%)           -3.433341e+00
Critical Value (5%)           -2.862861e+00
Critical Value (10%)          -2.567473e+00
dtype: float64
Dissolved Oxygen(mg/L)
```

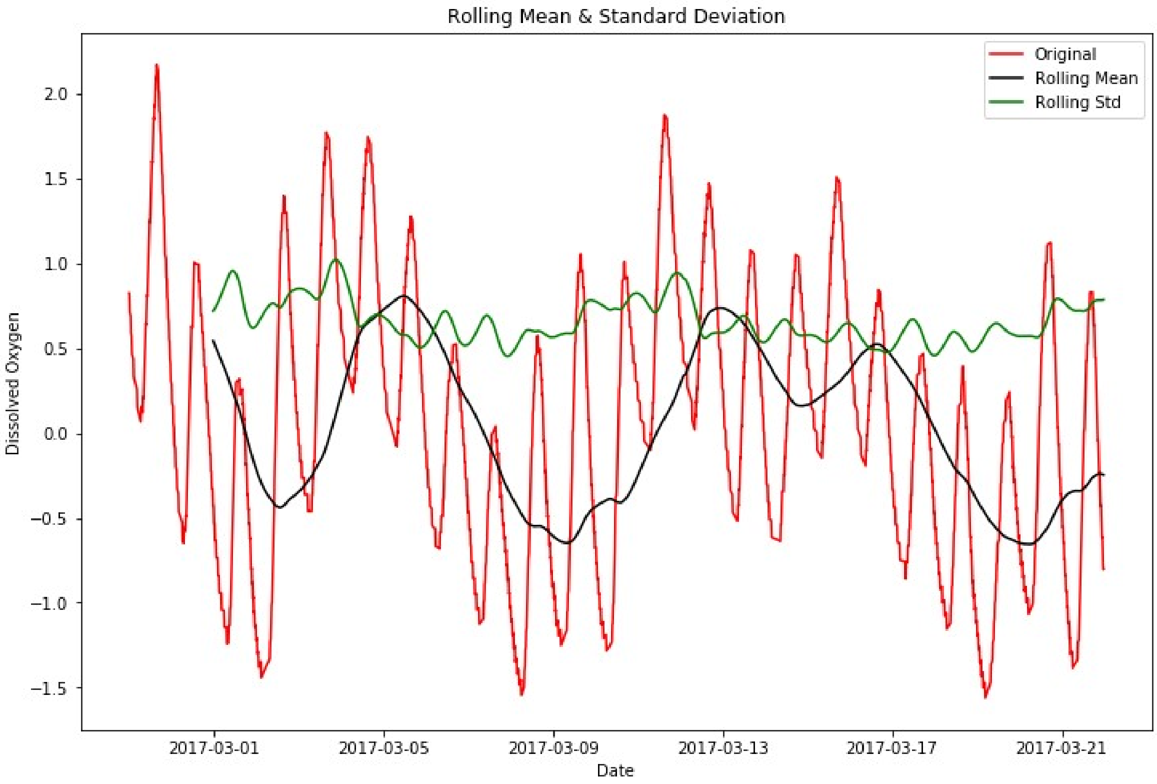


```
In [126]: def moving_average():
     ...:     #do_logScale = np.log(do)
     ...:     #plt.plot(do_logScale)
     ...:
     ...:     do_ma = do.rolling(window=192).mean() #window size 12 denotes 12 months,
giving rolling mean at yearly level
     ...:     #sc_movingSTD = sc_logScale.rolling(window=192).std()
     ...:
     ...:
     ...: #   plt.plot(sc_logScale)
     ...: #plt.plot(sc_moving_Average, color='blue')
     ...:
     ...:
     ...:     plt.figure(figsize=(12,8))
     ...:     plt.plot(do, color = "red",label = "Original")
     ...:     plt.plot(do_ma, color='black', label = "DO moving_avg_mean")
     ...:     plt.title("Dissolved Oxygen Rolling mean(mg/L) of Potomac River")
     ...:     plt.xlabel("Date")
     ...:     plt.ylabel("DO(mg/L)")
     ...:     plt.legend()
     ...:     plt.show()
     ...:
     ...:     do_ma_diff = do - do_ma
     ...:     #sc_LogScaleMinusMovingAverage.head(100)
     ...:
     ...:     do_ma_diff.dropna(inplace=True)
     ...:     #print(sc_rolmean,sc_rolstd)
     ...:
     ...:     check_adfuller(do_ma_diff['DO(mg/L)'])
```

```
   ...:     check_mean_std(do_ma_diff, 'Dissolved Oxygen(mg/L)')

In [127]: moving_average()
   ...:
```

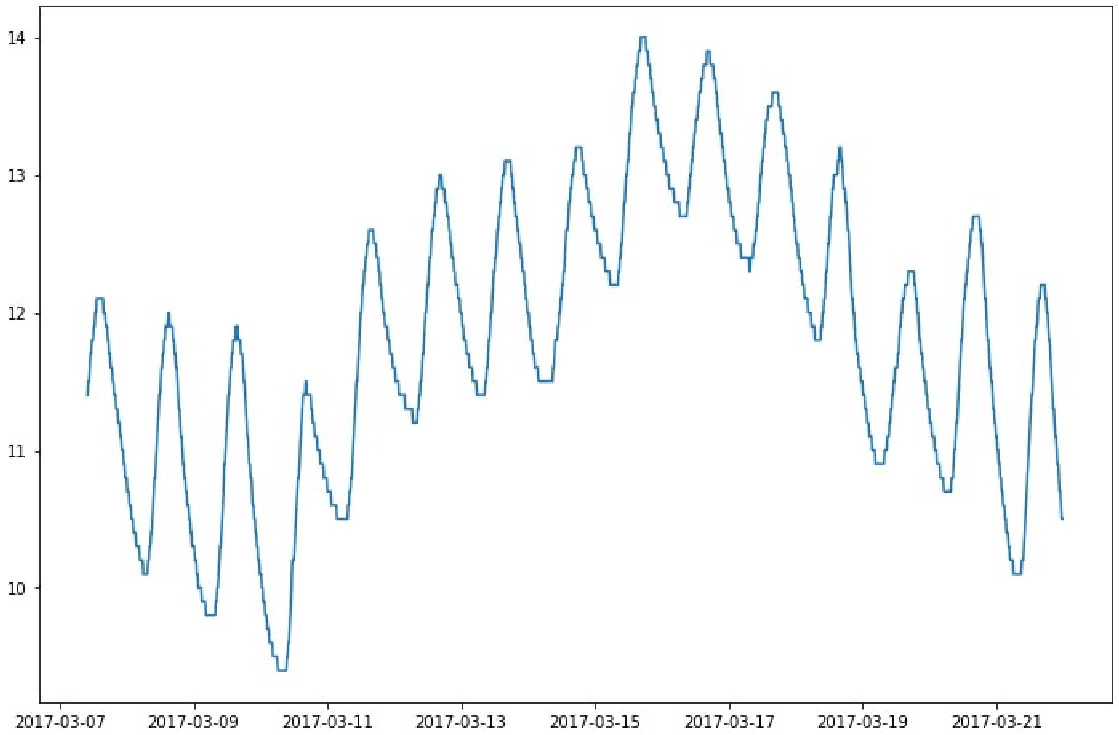


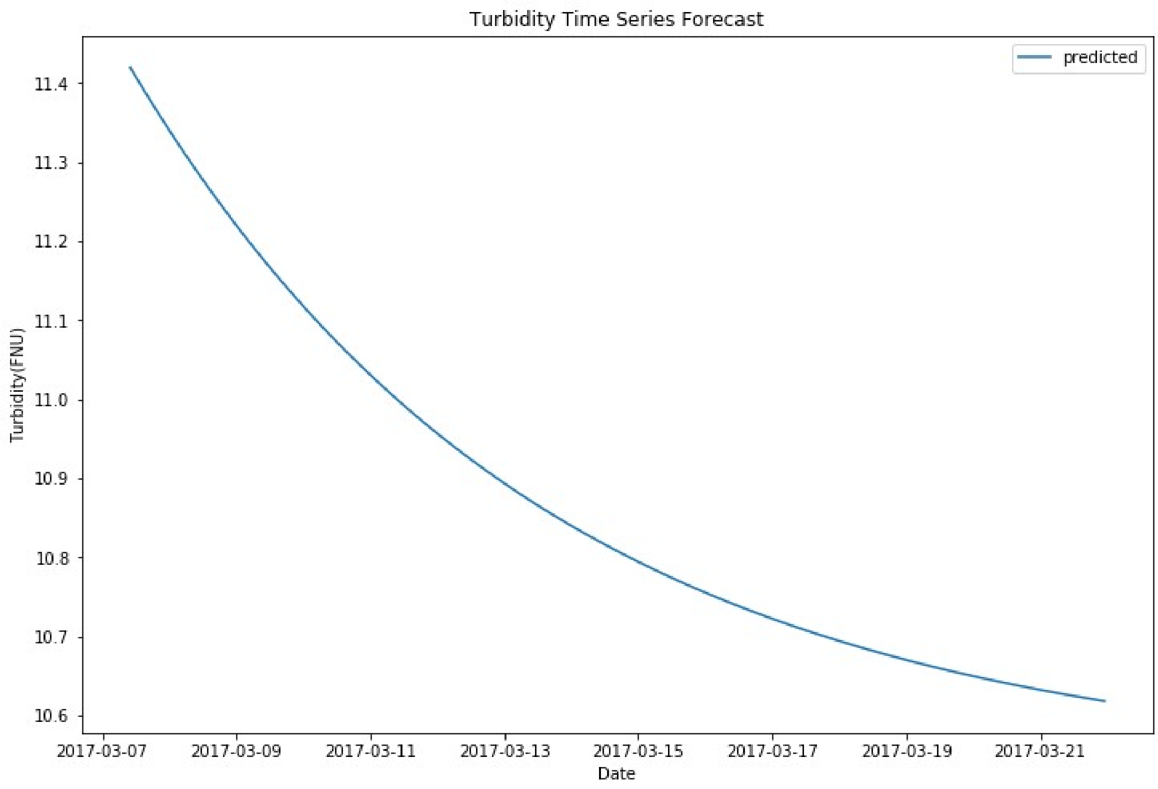
```
Results of Dickey Fuller Test:
--------For a stationary time series Test statistic is less than critical
values-----------
Test Statistic                -6.645701e+00
p-value                        5.272523e-09
#Lags Used                     1.500000e+01
Number of Observations Used    2.189000e+03
Critical Value (1%)           -3.433341e+00
Critical Value (5%)           -2.862861e+00
Critical Value (10%)          -2.567473e+00
dtype: float64
Dissolved Oxygen(mg/L)
```



```
In [128]: arima_model(do_train,(1,0,1))
     ...:
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:171:
ValueWarning: No frequency information was provided, so inferred frequency 15T will be
used.
  % freq, ValueWarning)
Traceback (most recent call last):

  File "<ipython-input-128-a923305cb3d8>", line 1, in <module>
    arima_model(do_train,(1,0,1))

  File "<ipython-input-116-6f284f1521a0>", line 7, in arima_model
    forecast = model_fit.predict(start=1919, end=5171)

  File "C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\base\wrapper.py", line 95,
in wrapper
    obj = data.wrap_output(func(results, *args, **kwargs), how)

  File "C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\base\data.py", line 416, in
wrap_output
    return self.attach_dates(obj)

  File "C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\base\data.py", line 560, in
attach_dates
    return Series(squeezed, index=self.predict_dates)

  File "C:\Users\admin\Anaconda3\lib\site-packages\pandas\core\series.py", line 262, in
__init__
    .format(val=len(data), ind=len(index)))
```

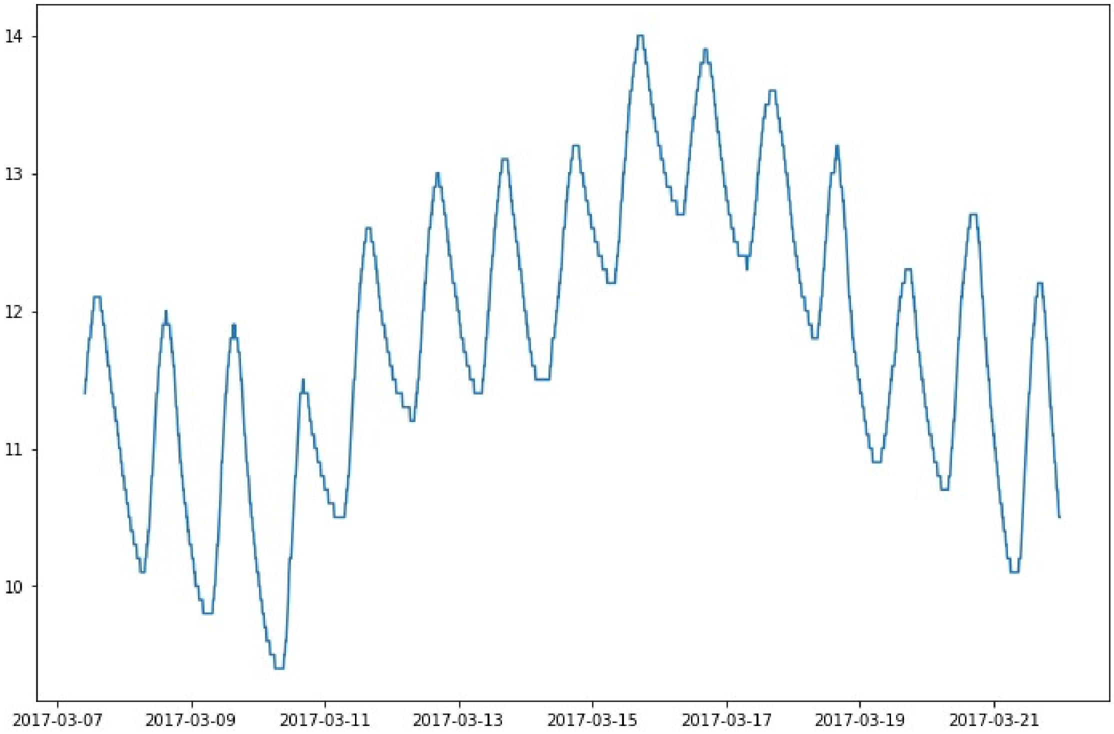
```
ValueError: Length of passed values is 4172, index implies 3253


In [129]:

In [129]: def arima_model(ts, order):
     ...:     # fit model
     ...:     model = ARIMA(ts, order=order) # (ARMA) = (p,d,q)
     ...:     model_fit = model.fit(disp=0)
     ...:
     ...:     # predict
     ...:     forecast = model_fit.predict(start=1000, end=2396)
     ...:
     ...:     # visualization
     ...:     plt.figure(figsize=(12,8))
     ...:     plt.plot(do_test,label = "original")
     ...:     plt.figure(figsize=(12,8))
     ...:     plt.plot(forecast,label = "predicted")
     ...:     plt.title("Turbidity Time Series Forecast")
     ...:     plt.xlabel("Date")
     ...:     plt.ylabel("Turbidity(FNU)")
     ...:     plt.legend()
     ...:     plt.show()

In [130]: arima_model(do_train,(1,0,1))
     ...:
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:171:
ValueWarning: No frequency information was provided, so inferred frequency 15T will be
used.
  % freq, ValueWarning)
```

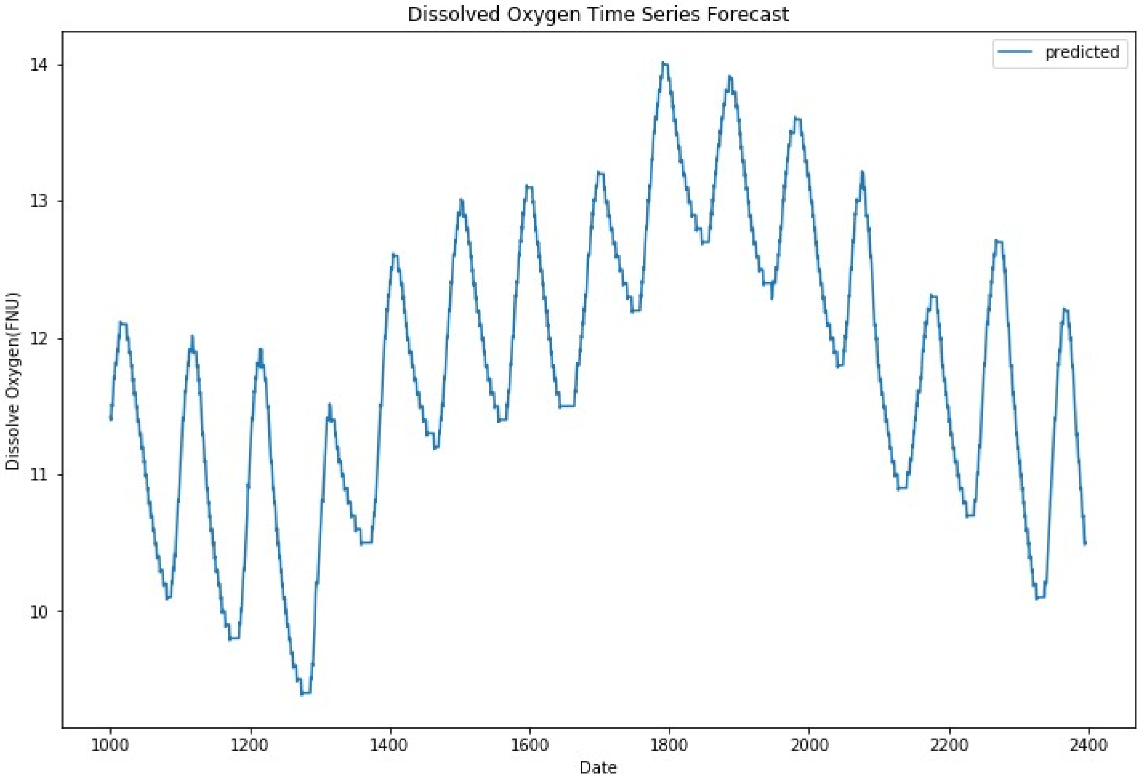




```
In [131]: def arima_model(ts, order):
     ...:     # fit model
     ...:     model = ARIMA(ts, order=order) # (ARMA) = (p,d,q)
     ...:     model_fit = model.fit(disp=0)
     ...:
     ...:     # predict
     ...:     forecast = model_fit.predict(start=1000, end=2396)
     ...:
     ...:     # visualization
     ...:     plt.figure(figsize=(12,8))
     ...:     plt.plot(do_test,label = "original")
     ...:     plt.figure(figsize=(12,8))
     ...:     plt.plot(forecast,label = "predicted")
     ...:     plt.title("Dissolved Oxygen Time Series Forecast")
     ...:     plt.xlabel("Date")
     ...:     plt.ylabel("Dissolve Oxygen(FNU)")
     ...:     plt.legend()
     ...:     plt.show()

In [132]: arima_model(do,(1,0,1))
     ...:
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:225:
ValueWarning: A date index has been provided, but it has no associated frequency
information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:531:
ValueWarning: No supported index is available. Prediction results will be given with an
integer index beginning at `start`.
```

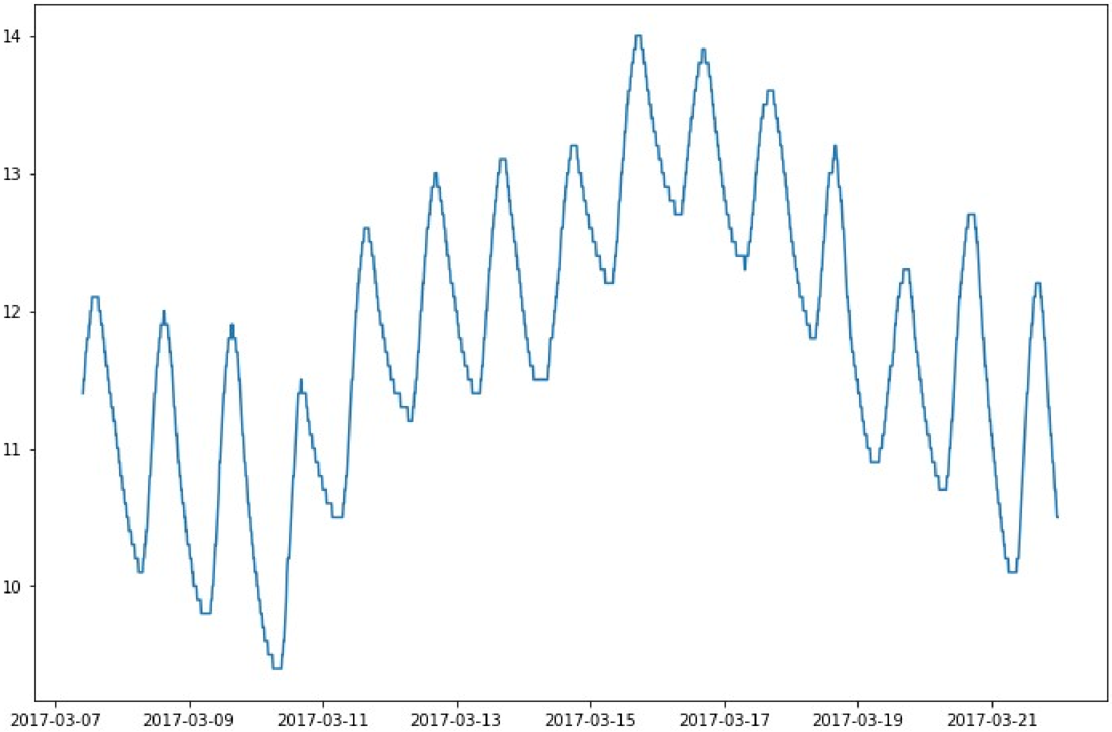
```
 ValueWarning)
```

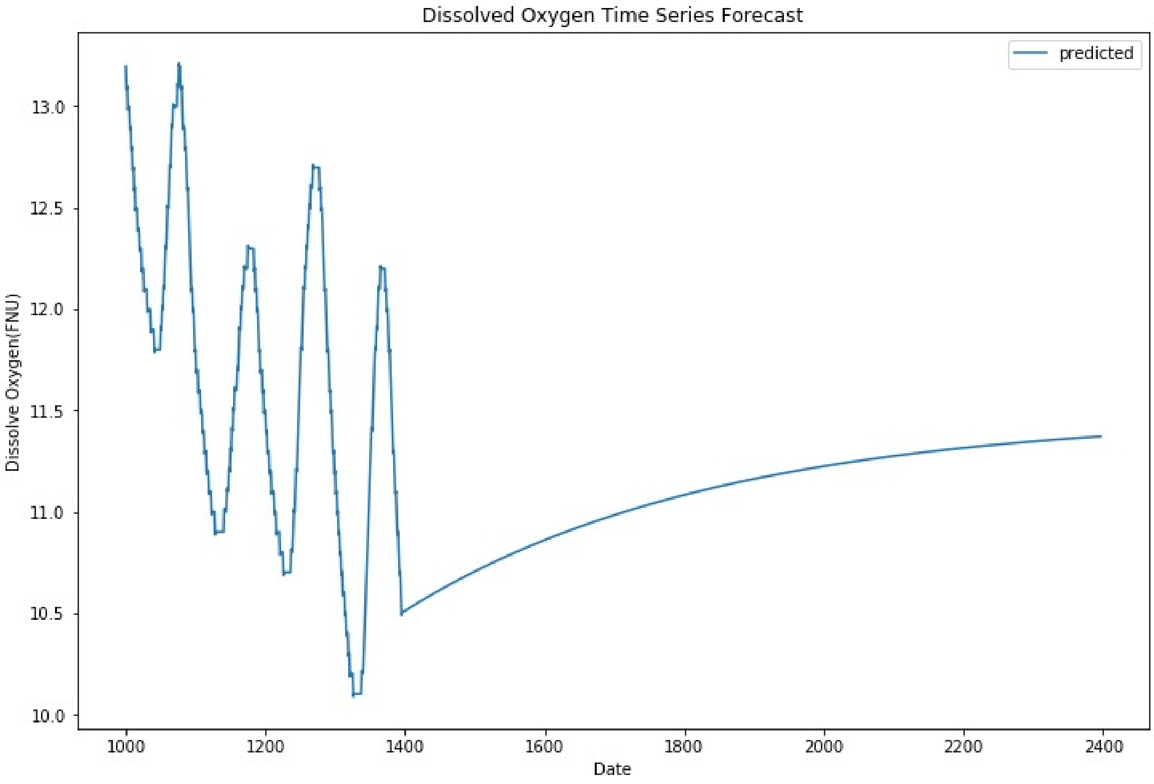




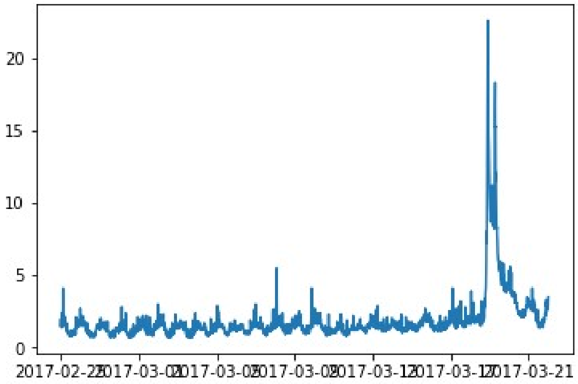
```
In [133]: train_size,test_size = 1500, 896
     ...: do_train,do_test = tts(do,test_size = test_size, random_state=0, shuffle=False)
     ...: #dataset.fillna(method ='bfill')
```

```
In [134]: arima_model(do_test,(1,0,1))
     ...:
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:171:
ValueWarning: No frequency information was provided, so inferred frequency 15T will be
used.
  % freq, ValueWarning)
Traceback (most recent call last):

  File "<ipython-input-134-022018485372>", line 1, in <module>
    arima_model(do_test,(1,0,1))

  File "<ipython-input-131-b27d8fe34b80>", line 4, in arima_model
    model_fit = model.fit(disp=0)

  File "C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py", line
946, in fit
    start_ar_lags)

  File "C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py", line
562, in _fit_start_params
    start_params = self._fit_start_params_hr(order, start_ar_lags)

  File "C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py", line
541, in _fit_start_params_hr
    raise ValueError("The computed initial AR coefficients are not "

ValueError: The computed initial AR coefficients are not stationary
You should induce stationarity, choose a different model order, or you can
pass your own start_params.


In [135]:

In [135]: train_size,test_size = 1000, 1396
     ...: do_train,do_test = tts(do,test_size = test_size, random_state=0, shuffle=False)
     ...: #dataset.fillna(method ='bfill')

In [136]: arima_model(do_test,(1,0,1))
     ...:
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:225:
ValueWarning: A date index has been provided, but it has no associated frequency
information and so will be ignored when e.g. forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
C:\Users\admin\Anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:531:
ValueWarning: No supported index is available. Prediction results will be given with an
integer index beginning at `start`.
  ValueWarning)
```





```
In [137]: turb = dataset.filter(['Turb(FNU)'], axis=1)

In [138]: plt.plot(turb)
Out[138]: [<matplotlib.lines.Line2D at 0x16fbe92b588>]
```



```
In [139]: def input_data():
     ...:     def parser(x):
     ...:         return datetime.strptime(x,'%Y-%m-%d %H:%M')
     ...:     dataset = pd.read_csv('Data7.csv',header=0, delimiter=',',index_col=0,
parse_dates=[0], date_parser=parser)
     ...:     dataset = dataset.fillna(method ='pad')
     ...:     turb = dataset.filter(['Turb(FNU)'], axis=1)
     ...:     train_size,test_size = 1000, 1396
     ...:     turb_train,turb_test = tts(turb,test_size = test_size, random_state=0,
shuffle=False)
     ...:     #dataset.fillna(method ='bfill')

In [140]: input_data()

In [141]:
```