ZINC

November 5, 2022

1 Import required libraries

[25]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib.pyplot import rcParams
rcParams['figure.figsize'] = 15,6
from pandas.tseries.offsets import DateOffset

import math
from datetime import datetime as dt
from statsmodels.tsa.stattools import adfuller, acf, pacf
from statsmodels.tsa.arima_model import ARIMA

import warnings
warnings.filterwarnings('ignore')
```

2 Visualize the time series

[2]:
```
df = pd.read_csv('zinc_prices_IMF.csv')
```

[3]:
```
df['Date'].head()
```

[3]:
```
0    1-Jan-80
1    1-Feb-80
2    1-Mar-80
3    1-Apr-80
4    1-May-80
Name: Date, dtype: object
```

[4]:
```
df = df.set_index('Date')
df.head
```

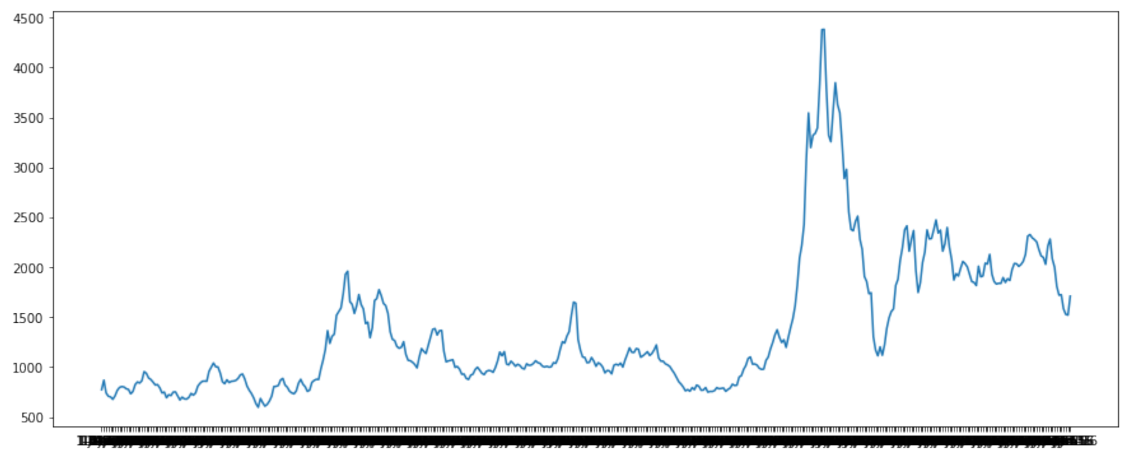
[4]: <bound method NDFrame.head of   Price
```
Date
1-Jan-80    773.82
1-Feb-80    868.62
1-Mar-80    740.75
1-Apr-80    707.68
1-May-80    701.07
…              …
1-Oct-15   1724.34
1-Nov-15   1583.31
1-Dec-15   1527.79
1-Jan-16   1520.36
1-Feb-16   1709.85

[434 rows x 1 columns]>
```

[5]:
```
ts = df['Price']
```

[6]:
```
plt.plot(ts)
```

[6]: [<matplotlib.lines.Line2D at 0x7f2a07e89710>]



3 Check for the stationarity of your data using Rolling Statistics and Dickey-Fuller test

[7]:
```
ts_log = np.log(ts)
```

[8]:
```
def test_stationarity (timeseries):
    rolmean = timeseries.rolling(window=52, center = False).mean()
```

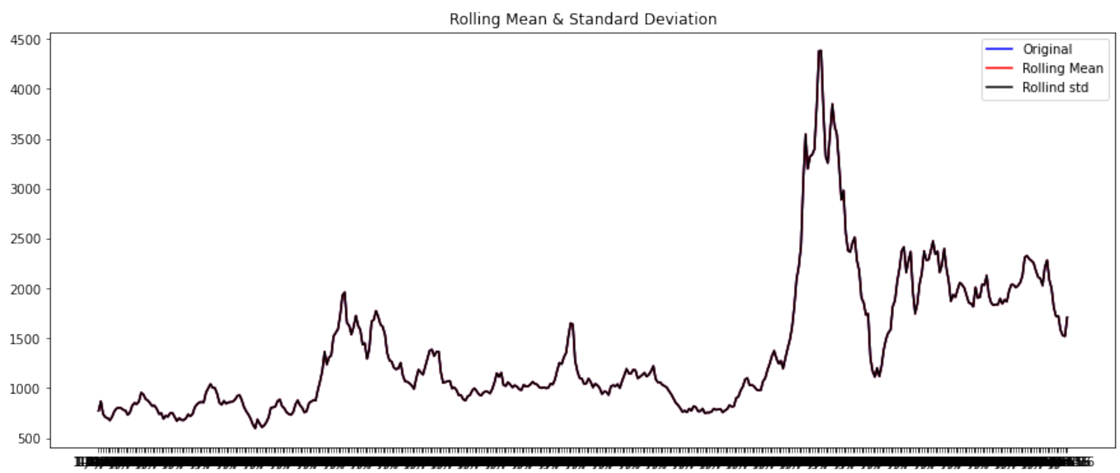
```
    rolstd = timeseries.rolling(window=52, center = False).std()

    orig = plt.plot(timeseries,color='blue', label='Original')
    mean = plt.plot(timeseries, color='red', label='Rolling Mean')
    std = plt.plot(timeseries, color='black', label='Rollind std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block= False)

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

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

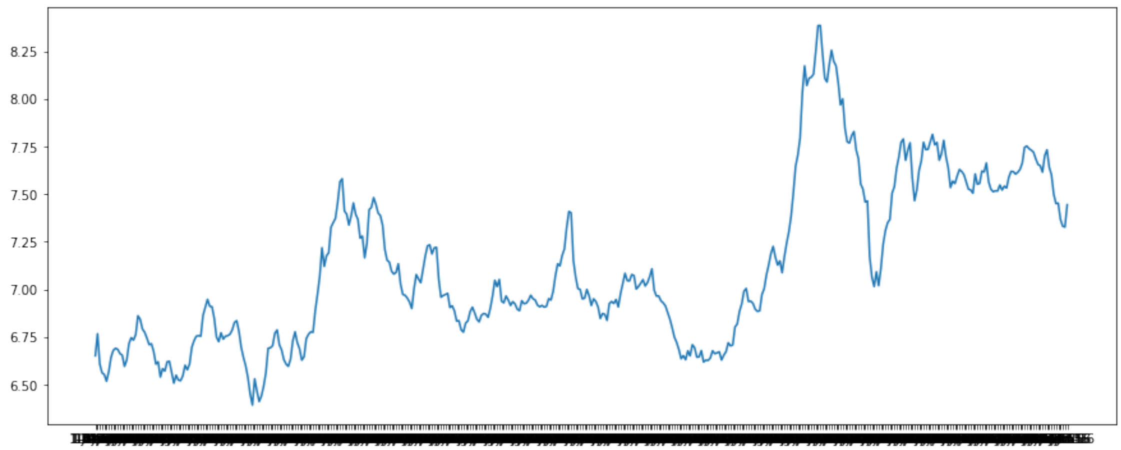
[9]:
```
test_stationarity(df['Price'])
```




```
Results of Dickey-Fuller Test:
Test Statistic                  -3.139601
p-values                         0.023758
#Lags Used                       7.000000
Number of Observations Used    426.000000
Critical Value (1%)             -3.445794
Critical Value (5%)             -2.868349
Critical Value (10%)            -2.570397
dtype: float64
```

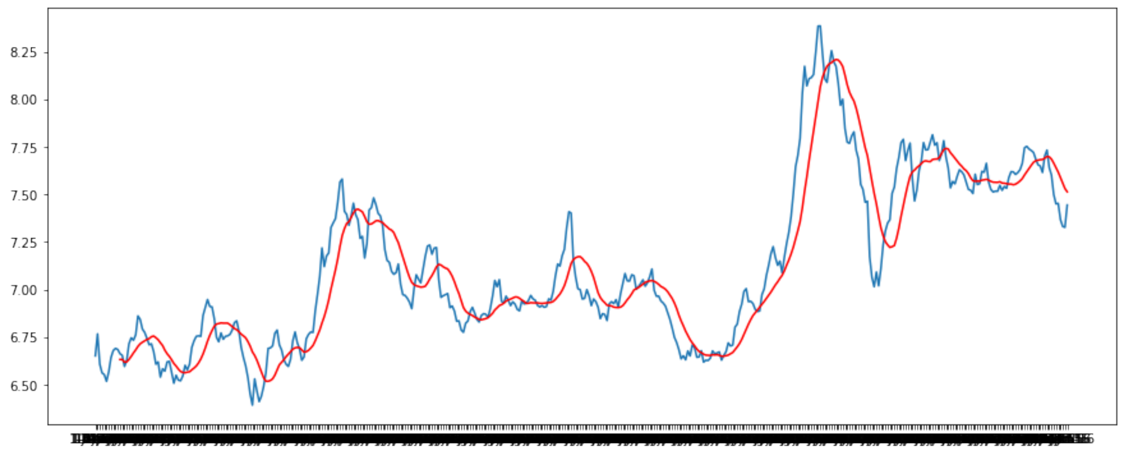
[10]:
```
plt.plot(ts_log)
```

[10]: [<matplotlib.lines.Line2D at 0x7f2a0552e110>]



[11]:
```
MovingAverage = ts_log.rolling(window=12).mean()
MovingSTD = ts_log.rolling(window=12).std()
plt.plot(ts_log)
plt.plot(MovingAverage, color='red')
```

[11]: [<matplotlib.lines.Line2D at 0x7f2a04a840d0>]



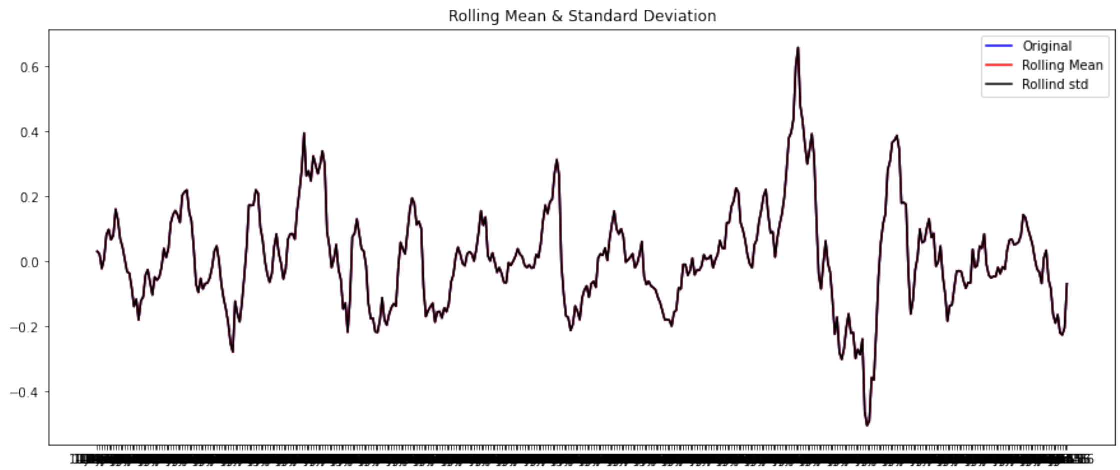
[12]:
```
ts_log_mv_diff = ts_log - MovingAverage
ts_log_mv_diff.head(12)
```

[12]: Date
```
1-Jan-80          NaN
1-Feb-80          NaN
1-Mar-80          NaN
1-Apr-80          NaN
1-May-80          NaN
1-Jun-80          NaN
1-Jul-80          NaN
1-Aug-80          NaN
1-Sep-80          NaN
1-Oct-80          NaN
1-Nov-80          NaN
1-Dec-80     0.030472
Name: Price, dtype: float64
```

[13]:
```
ts_log_mv_diff.dropna(inplace=True)
ts_log_mv_diff.head(12)
```

[13]:
```
Date
1-Dec-80     0.030472
1-Jan-81     0.021753
1-Feb-81    -0.022485
1-Mar-81     0.008392
1-Apr-81     0.082191
1-May-81     0.097617
1-Jun-81     0.066587
1-Jul-81     0.078914
1-Aug-81     0.160180
1-Sep-81     0.127928
1-Oct-81     0.068802
1-Nov-81     0.043854
Name: Price, dtype: float64
```

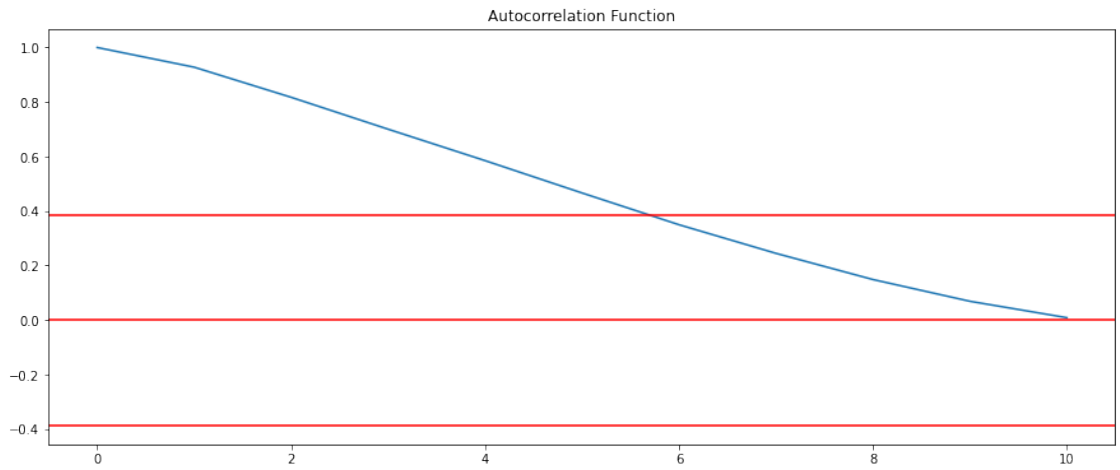
[14]:
```
test_stationarity(ts_log_mv_diff)
```



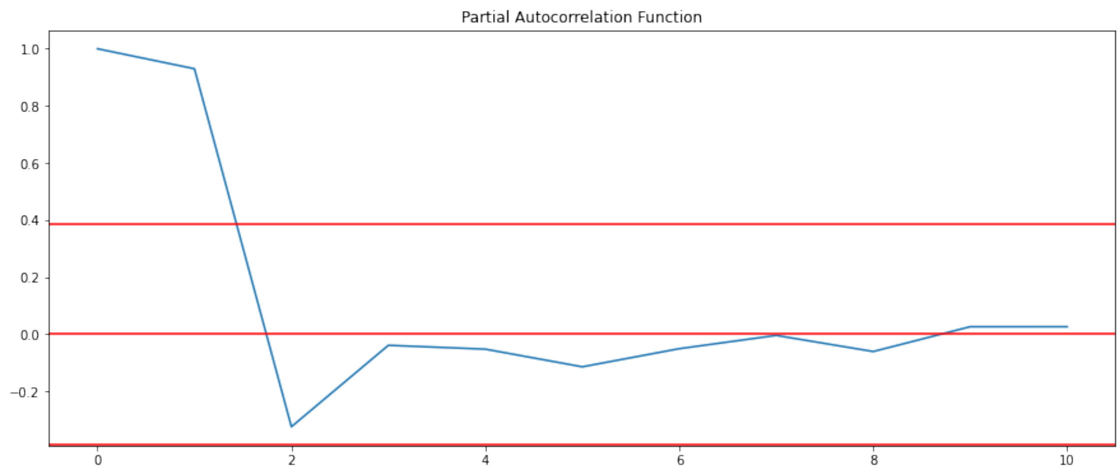

```
Results of Dickey-Fuller Test:
Test Statistic                 -5.898484e+00
p-values                        2.814411e-07
#Lags Used                      4.000000e+00
Number of Observations Used     4.180000e+02
Critical Value (1%)            -3.446091e+00
Critical Value (5%)            -2.868479e+00
Critical Value (10%)           -2.570466e+00
dtype: float64
```

4 Plot ACF and PACF plots

[15]:
```
plt.plot(np.arange(0,11), acf(ts_log_mv_diff,nlags=10))
plt.axhline(y=0, linestyle='-',color='red')
plt.axhline(y=-7.96/np.sqrt(len(ts_log_mv_diff)),linestyle='-', color='red')
plt.axhline(y=7.96/np.sqrt(len(ts_log_mv_diff)),linestyle='-', color='red')
plt.title('Autocorrelation Function')
plt.show()
```



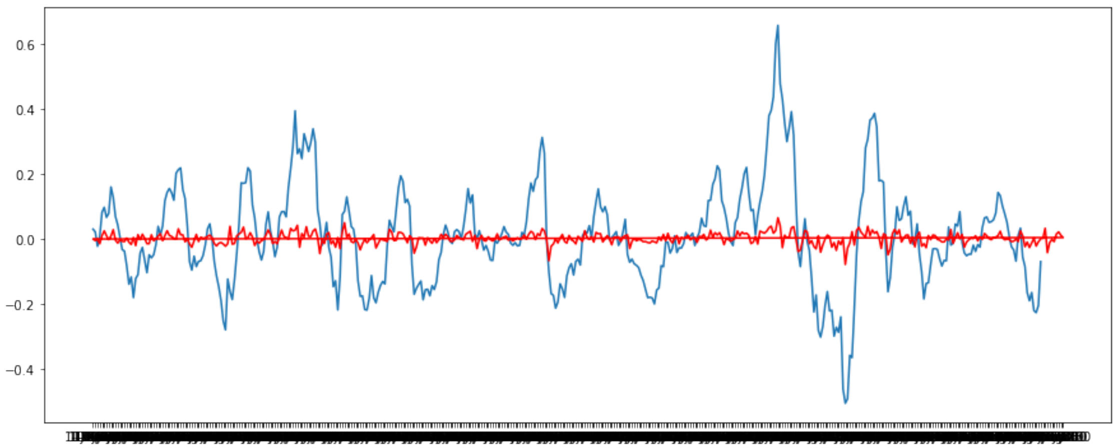
[16]:
```
plt.plot(np.arange(0,11), pacf(ts_log_mv_diff,nlags=10))
plt.axhline(y=0, linestyle='-',color='red')
plt.axhline(y=-7.96/np.sqrt(len(ts_log_mv_diff)),linestyle='-', color='red')
plt.axhline(y=7.96/np.sqrt(len(ts_log_mv_diff)),linestyle='-', color='red')
plt.title('Partial Autocorrelation Function')
plt.show()
```



5 Perform ARIMA modeling

[17]:
```
model = ARIMA (ts_log, order=(1,1,0))
results_ARIMA = model.fit(disp=-1)
plt.plot(ts_log_mv_diff)
plt.plot(results_ARIMA.fittedvalues, color ='red')
```

[17]:
```
[<matplotlib.lines.Line2D at 0x7f29f87195d0>]
```



[18]:
```
predictions_ARIMA_diff = pd.Series(results_ARIMA.fittedvalues, copy=True)
predictions_ARIMA_diff.head()
```

[18]:
```
Date
1-Feb-80    0.002030
1-Mar-80    0.033049
1-Apr-80   -0.042031
1-May-80   -0.011002
1-Jun-80   -0.001089
dtype: float64
```

[19]:
```
predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()
predictions_ARIMA_diff_cumsum.head()
```

[19]:
```
Date
1-Feb-80    0.002030
1-Mar-80    0.035079
1-Apr-80   -0.006952
1-May-80   -0.017955
1-Jun-80   -0.019043
dtype: float64
```

[20]:
```
predictions_ARIMA_log  = pd.Series(ts_log[0], index=ts_log.index)
```

```
predictions_ARIMA_log  = predictions_ARIMA_log.
 ↪add(predictions_ARIMA_diff_cumsum, fill_value=0)
predictions_ARIMA_log.head()
```

[20]: Date

```
1-Apr-00    7.109676
1-Apr-01    7.099660
1-Apr-02    7.061613
1-Apr-03    7.069247
1-Apr-04    7.177723
dtype: float64
```

6 Forecast the prices using the new model

[31]:
```
predictions_ARIMA = np.exp(predictions_ARIMA_log)
plt.plot(ts)
plt.plot(predictions_ARIMA)
```

[31]: [<matplotlib.lines.Line2D at 0x7f2a050b62d0>]