Time Series

May 9, 2021

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from pmdarima import auto_arima
```

```
        ---------------------------------------------------------------------------

        ModuleNotFoundError                       Traceback (most recent call last)

        <ipython-input-1-45d92691ba58> in <module>
          3 import matplotlib.pyplot as plt
          4 import statsmodels.api as sm
    ----> 5 from pmdarima import auto_arima


        ModuleNotFoundError: No module named 'pmdarima'
```

[27]:
```
df=pd.read_csv("Mahakal_TimeSeriesData.csv")
df.head()
```

[27]:

	Date	Tweets
0	11-Mar-21	2365
1	12-Mar-21	1211
2	13-Mar-21	400
3	14-Mar-21	285
4	15-Mar-21	203

[28]:
```
df['Date']=pd.to_datetime(df['Date'])
df.head()
```

[28]:

	Date	Tweets
0	2021-03-11	2365
1	2021-03-12	1211
2	2021-03-13	400
3	2021-03-14	285

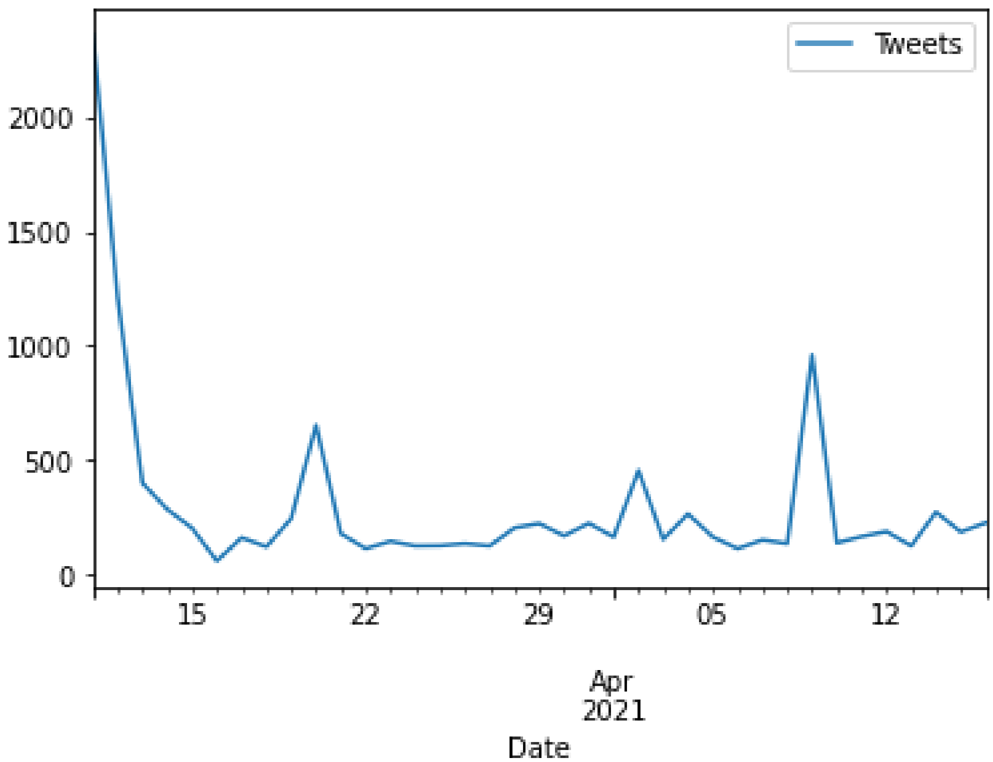
```
4 2021-03-15     203
```

[29]:
```
df.set_index('Date',inplace=True)
df.index.freq="D"
df.head()
```

[29]:

Date	Tweets
2021-03-11	2365
2021-03-12	1211
2021-03-13	400
2021-03-14	285
2021-03-15	203

[30]:
```
df.plot();
```



[31]:
```
# Testing For stationarity

from statsmodels.tsa.stattools import adfuller
```

[32]:
```
# H0: It is non stationary
# H1: It is stationary
```

```
def adfuller_test(Tweets):
    result = adfuller(Tweets)
    labels = ["ADF Test statistics","P-value","#Lags Used","Number of␣
↪Observation Used"]
    for value,labels in zip(result,labels):
        print(labels+' : '+str(value) )
    if result[1] <= 0.05:
        print("Strong evidence against null hypothesis")
    else:
        print("weak evidence against null hypothesis")
```

[34]:
```
adfuller_test(df["Tweets"])
```

```
ADF Test statistics : -8.772784249101026
P-value : 2.494858049465755e-14
#Lags Used : 0
Number of Observation Used : 36
Strong evidence against null hypothesis
```

[36]:
```
#from statsmodels.tsa.stattools import grangercausalitytests
```

[83]:
```
#grangercausalitytests(df[['Tweets','Tweets_First_Difference']],maxlag=3);
```

1 Differencing

[34]:
```
#df['Tweets_First_Difference'] = df['Tweets']-df['Tweets'].shift(2)
#df['Tweets'].shift(2)
```

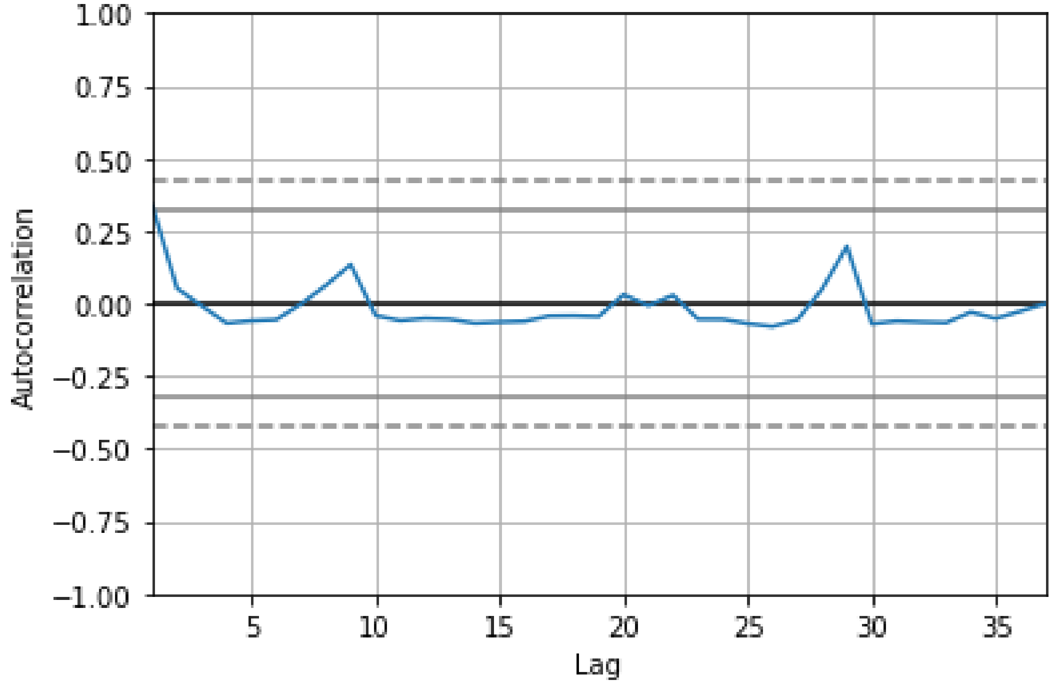
[33]:
```
# Again test dickey fuller test
#adfuller_test(df['Tweets_First_Difference'].dropna())
```

[117]:
```
#df['Tweets_First_Difference'].plot();
```

2 Auto Regressive Model

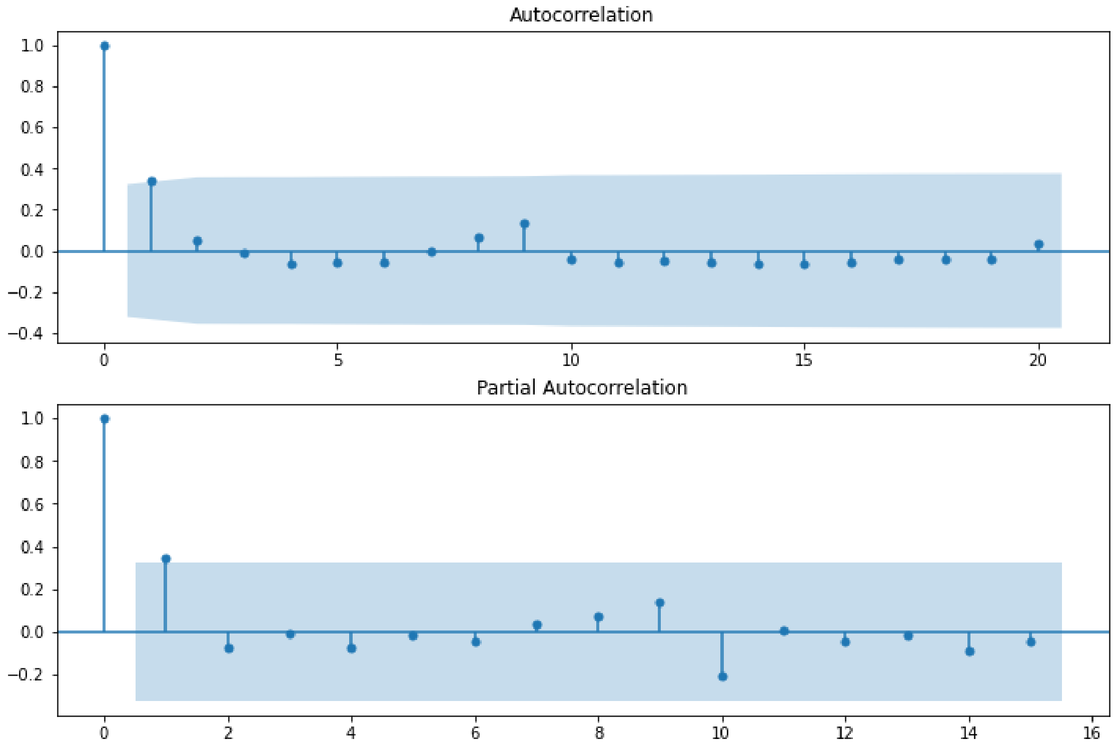
- Identification of an AR model is often best done with the PACF.
- Identification of an MA model is often best done with the ACF rather than PACF.

[35]:
```
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(df['Tweets']);
```



[36]:
```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
import statsmodels.api as sm
```

[38]:
```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df['Tweets'].iloc[0:],lags=20,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df['Tweets'].iloc[0:],lags=15,ax=ax2)
```



[ ]:

```
```

[39]:

```
# For non-seasonal data
# p=1, d=0, q=0
from statsmodels.tsa.arima_model import ARMA
from statsmodels.tsa.arima_model import ARIMA
```

[40]:

```
model = ARMA(df['Tweets'],order=(1,0))
model_fit1=model.fit()
```

```
C:\Users\asus\anaconda3\lib\site-packages\statsmodels\tsa\arima_model.py:472:
FutureWarning:
statsmodels.tsa.arima_model.ARMA and statsmodels.tsa.arima_model.ARIMA have
been deprecated in favor of statsmodels.tsa.arima.model.ARIMA (note the .
between arima and model) and
statsmodels.tsa.SARIMAX. These will be removed after the 0.12 release.

statsmodels.tsa.arima.model.ARIMA makes use of the statespace framework and
is both well tested and maintained.

To silence this warning and continue using ARMA and ARIMA until they are
removed, use:
```

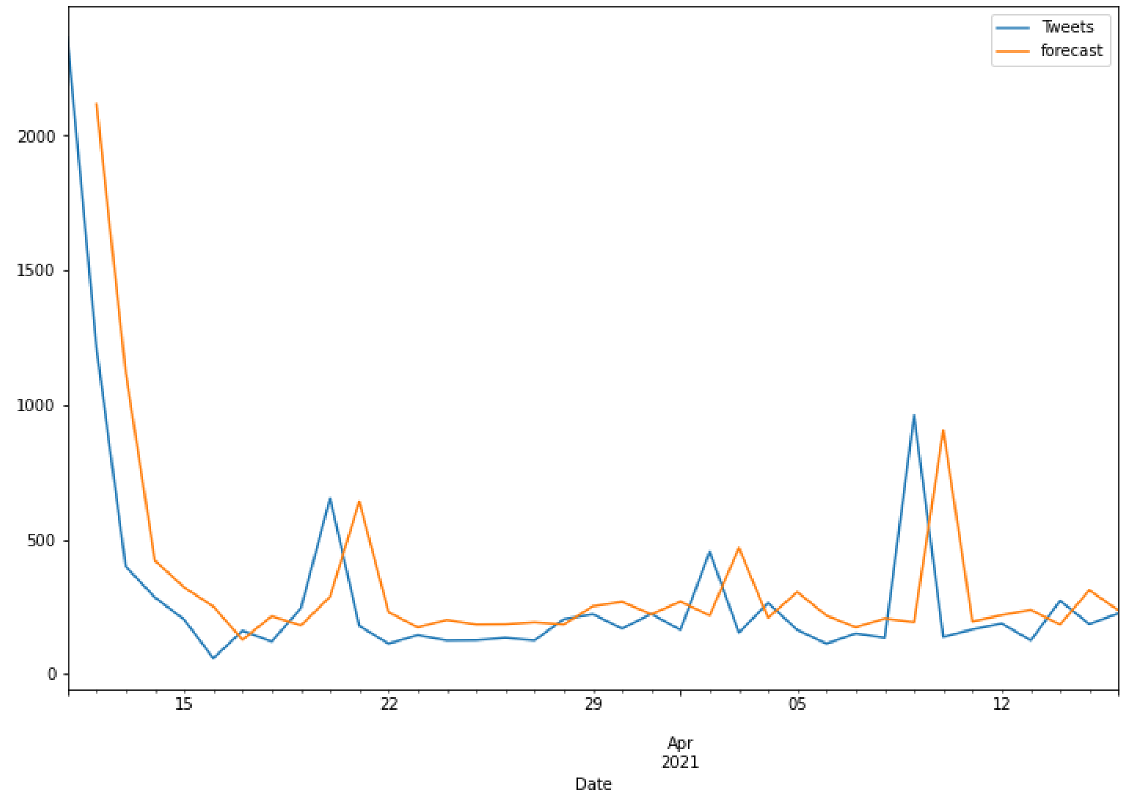
```
import warnings
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARMA',
                        FutureWarning)
warnings.filterwarnings('ignore', 'statsmodels.tsa.arima_model.ARIMA',
                        FutureWarning)
```

```
  warnings.warn(ARIMA_DEPRECATION_WARN, FutureWarning)
```

[41]:
```
model_fit1.summary()
```

[41]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                              ARMA Model Results
==============================================================================
Dep. Variable:                 Tweets   No. Observations:                   37
Model:                     ARMA(1, 0)   Log Likelihood                -267.954
Method:                       css-mle   S.D. of innovations            331.825
Date:                Sat, 17 Apr 2021   AIC                            541.908
Time:                        22:06:59   BIC                            546.741
Sample:                    03-11-2021   HQIC                           543.612
                         - 04-16-2021
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const          557.8855    407.796      1.368      0.171    -241.379    1357.150
ar.L1.Tweets     0.8630      0.138      6.259      0.000       0.593       1.133
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.1587           +0.0000j            1.1587            0.0000
-----------------------------------------------------------------------------
"""
```

[43]:
```
df['forecast']=model_fit1.predict(start=1,end=39, dynamic=False)
#pd.Series(model_fit1.fittedvalues,copy=True)
df[['Tweets','forecast']].plot(figsize=(12,8));
```



[106]:
```
#y16=799.22+(1.5339)*df['Tweets'][15]-0.6678*df['Tweets'][14]
```

[44]:
```
model_fit1.forecast(steps=2)[0]
```

[44]:
```
array([271.45534191, 310.68507806])
```

[45]:
```
import pmdarima as pm
```

[51]:
```
def arimamodel(df):
    automodel=pm.
↪auto_arima(df,start_p=0,start_q=0,max_p=4,max_q=4,test="adf",seasonal=False,trace=True)
    return automodel
```

[52]:
```
arimamodel(df["Tweets"])
```

```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=568.887, Time=0.01 sec
 ARIMA(1,0,0)(0,0,0)[0]             : AIC=541.635, Time=0.03 sec
 ARIMA(0,0,1)(0,0,0)[0]             : AIC=inf, Time=0.03 sec
 ARIMA(2,0,0)(0,0,0)[0]             : AIC=543.617, Time=0.05 sec
 ARIMA(1,0,1)(0,0,0)[0]             : AIC=543.620, Time=0.05 sec
 ARIMA(2,0,1)(0,0,0)[0]             : AIC=inf, Time=0.11 sec
```

```
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=541.908, Time=0.02 sec

Best model:  ARIMA(1,0,0)(0,0,0)[0]
Total fit time: 0.312 seconds
```

[52]:
```
ARIMA(order=(1, 0, 0), scoring_args={}, suppress_warnings=True,
      with_intercept=False)
```

[ ]:

[ ]:

[ ]:

[ ]:

[26]:
```
#model = sm.tsa.
 ↪SARIMAX(df["Tweets"],trend='c',order=(2,0,1),enforce_stationarity=False,enforce_invertibility
#model_fit2=model.fit()
```

[355]:
```
#model_fit2.summary()
```

[356]:
```
#df['forecast']=model_fit2.predict(start=1,end=40,dynamic=True)
#df[['Tweets','forecast']].plot(figsize=(12,8))
```

[32]:
```
#fc,se,conf=fitte
```

[34]:
```
#arima_model = auto_arima(df,n_fits=16,seasonal=False,error_action="ignore")
```

[35]:
```
#arima_model.summary()
```

[36]:
```
#prediction = pd.DataFrame(arima_model.predict(n_periods=20),index=df.index)
#prediction.columns=['prediction_tweets']
#prediction
```

[37]:
```
#plt.figure(figsize=(8,5))
#plt.plot(df)
#plt.plot(prediction);
```

[ ]: