ARIMA Modelling for a Univariate Time Series

July 15, 2024

0.0.1 Harsh Mittal

ARIMA Modelling

[28]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

[29]:
```
df = pd.read_csv("C:/Users/harsh.hm.mittal/Downloads/MSFT.csv")
```

[30]:
```
df.head()
```

[30]:

	Date	Close
0	14-07-2014	35.871395
1	15-07-2014	36.135273
2	16-07-2014	37.522812
3	17-07-2014	37.905876
4	18-07-2014	38.042068

[31]:
```
df = df[["Close"]].copy()
```

[32]:
```
df.describe()
```

[32]:

	Close
count	2516.000000
mean	164.301813
std	115.265938
min	34.823288
25%	57.289910
50%	129.766113
75%	254.843505
max	467.559998

0.0.2 Train test split

[33]:
```
n = int(len(df) * 0.8)
train = df.Close[:n]
test = df.Close[n:]
```

[34]:
```
print(len(train))
print(len(test))
```

```
2012
504
```

0.0.3 ADF Test - Checking whether price series is stationary

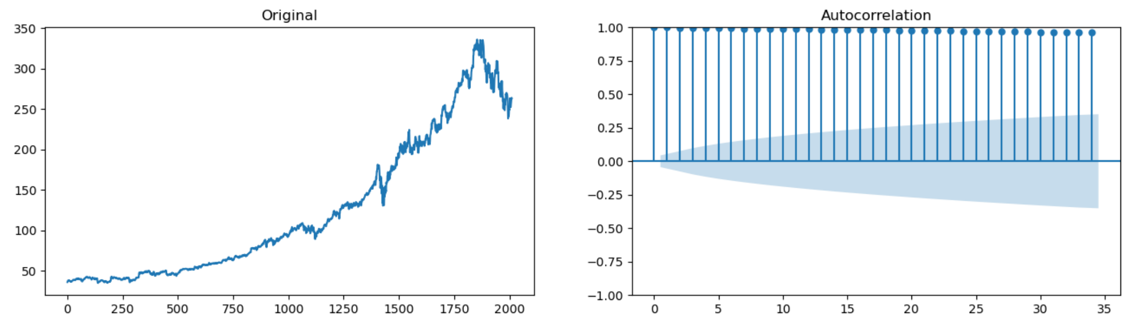
[35]:
```
from statsmodels.tsa.stattools import adfuller

result = adfuller(train)
print(f"ADF Statistics: {result[0]}")
print(f"p-value: {result[1]}")
```

```
ADF Statistics: 0.17655902198335482
p-value: 0.9709157868214724
```

0.0.4 Autocorrelation Function (ACF)

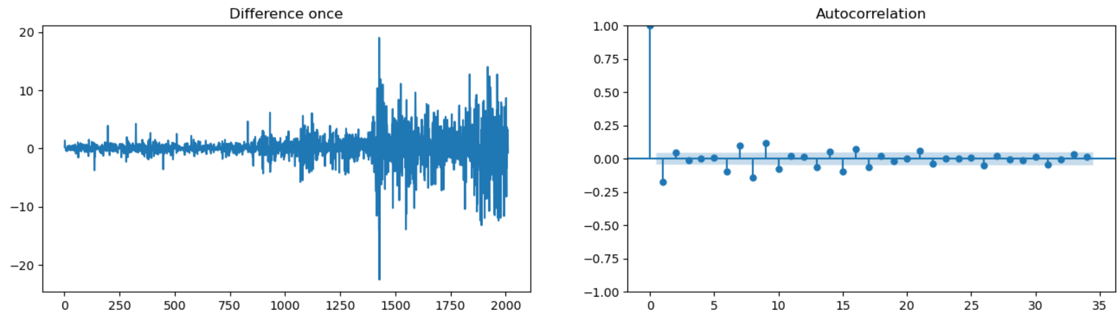
[36]:
```
from statsmodels.graphics.tsaplots import plot_acf
```

[37]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (16, 4))

ax1.plot(train)
ax1.set_title("Original")

plot_acf(train, ax=ax2);
```

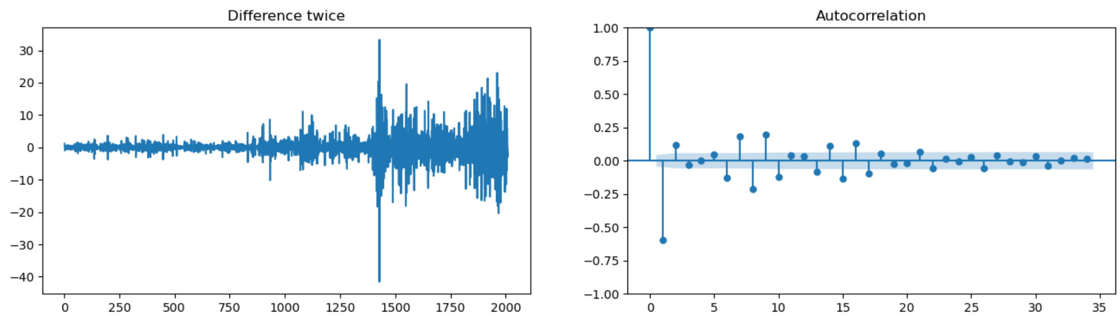


[38]:
```
diff = train.diff().dropna()
```

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (16, 4))

ax1.plot(diff)
ax1.set_title("Difference once")
plot_acf(diff, ax=ax2);
```



[39]:
```
diff = train.diff().diff().dropna()

fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (16, 4))

ax1.plot(diff)
ax1.set_title("Difference twice")
plot_acf(diff, ax=ax2);
```



pmdarima package to get the number of differencing

[40]:
```
!pip install pmdarima
```

```
Requirement already satisfied: pmdarima in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (2.0.4)
Requirement already satisfied: statsmodels>=0.13.2 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from pmdarima) (0.13.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from pmdarima) (0.29.32)
```

```
Requirement already satisfied: numpy>=1.21.2 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from pmdarima) (1.21.5)
Requirement already satisfied: scipy>=1.3.2 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from pmdarima) (1.9.1)
Requirement already satisfied: scikit-learn>=0.22 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from pmdarima) (1.0.2)
Requirement already satisfied: urllib3 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from pmdarima) (1.26.11)
Requirement already satisfied: pandas>=0.19 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from pmdarima) (1.4.4)
Requirement already satisfied: joblib>=0.11 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from pmdarima) (1.1.0)
Requirement already satisfied: packaging>=17.1 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from pmdarima) (21.3)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from pmdarima) (63.4.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from
packaging>=17.1->pmdarima) (3.0.9)
Requirement already satisfied: pytz>=2020.1 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from
pandas>=0.19->pmdarima) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from
pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from scikit-
learn>=0.22->pmdarima) (2.2.0)
Requirement already satisfied: patsy>=0.5.2 in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from
statsmodels>=0.13.2->pmdarima) (0.5.2)
Requirement already satisfied: six in
c:\users\harsh.hm.mittal\anaconda3\lib\site-packages (from
patsy>=0.5.2->statsmodels>=0.13.2->pmdarima) (1.16.0)
```

[41]:
```python
from pmdarima.arima.utils import ndiffs
```

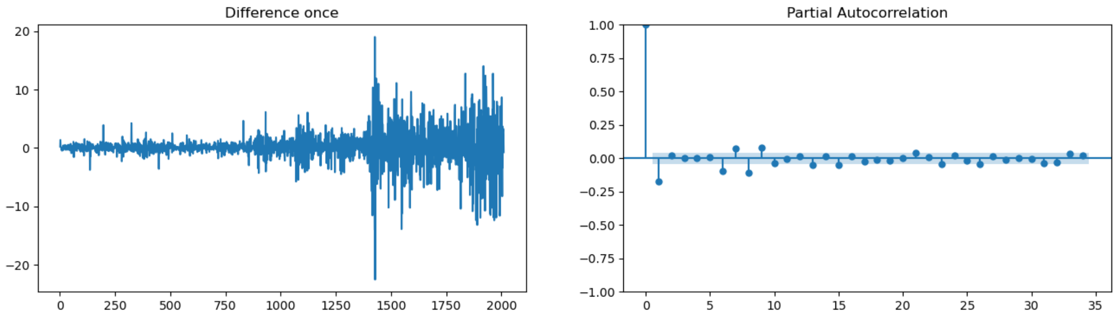
[42]:
```python
ndiffs(train, test="adf")
```

[42]: 1

p
p is the order of the Auto Regressive (AR) term. It refers to tthe number of lags to used as predictors.
Required number of AR terms can be found out by inspecting the Partial Autocorrelation (PACF) plot.
The partial autocorrelation represents the correlation between the series and its lags.

[43]:
```python
from statsmodels.graphics.tsaplots import plot_pacf
```

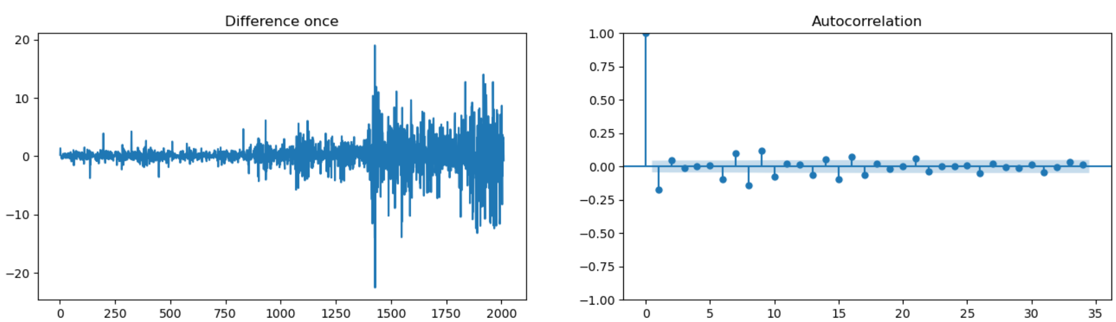
[44]:
```python
diff = train.diff().dropna()

fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (16, 4))

ax1.plot(diff)
ax1.set_title("Difference once")
ax2.set_ylim(0, 1)
plot_pacf(diff, ax=ax2);
```



We can observe that the PACF lag 9 is significant as it's above the significance line.

q
q is the order of the Moving Average (MA) term. It refers to the number of lagged forecast errors that should go into the ARIMA model.
ACF plot is to be looked out for MA terms.

[45]:
```python
diff = train.diff().dropna()

fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (16, 4))

ax1.plot(diff)
ax1.set_title("Difference once")
ax2.set_ylim(0, 1)
plot_acf(diff, ax=ax2);
```



0.0.5 Fitting the ARIMA model

[46]:
```python
from statsmodels.tsa.arima.model import ARIMA

# ARIMA Model
model = ARIMA(train, order=(9, 1, 9))
result = model.fit()
```

[47]:
```python
print(result.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 2012
Model:                 ARIMA(9, 1, 9)   Log Likelihood               -4802.858
Date:                Mon, 15 Jul 2024   AIC                           9643.716
Time:                        20:19:26   BIC                           9750.237
Sample:                             0   HQIC                          9682.817
                               - 2012
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1545      0.114      1.357      0.175      -0.069       0.378
ar.L2          0.6666      0.099      6.707      0.000       0.472       0.861
ar.L3         -0.3644      0.093     -3.923      0.000      -0.546      -0.182
ar.L4         -0.3521      0.111     -3.184      0.001      -0.569      -0.135
ar.L5          0.7325      0.059     12.472      0.000       0.617       0.848
ar.L6         -0.2822      0.092     -3.067      0.002      -0.463      -0.102
ar.L7         -0.6090      0.107     -5.684      0.000      -0.819      -0.399
ar.L8          0.2455      0.075      3.256      0.001       0.098       0.393
ar.L9          0.5351      0.077      6.927      0.000       0.384       0.686
ma.L1         -0.2764      0.116     -2.380      0.017      -0.504      -0.049
ma.L2         -0.6165      0.104     -5.911      0.000      -0.821      -0.412
ma.L3          0.4208      0.092      4.574      0.000       0.240       0.601
ma.L4          0.2749      0.112      2.446      0.014       0.055       0.495
ma.L5         -0.7440      0.054    -13.894      0.000      -0.849      -0.639
ma.L6          0.3122      0.090      3.481      0.000       0.136       0.488
ma.L7          0.6126      0.105      5.860      0.000       0.408       0.818
ma.L8         -0.3490      0.075     -4.636      0.000      -0.497      -0.201
ma.L9         -0.4270      0.078     -5.462      0.000      -0.580      -0.274
sigma2         6.9017      0.106     65.107      0.000       6.694       7.109
================================================================================
===
Ljung-Box (L1) (Q):                   0.67   Jarque-Bera (JB):
5424.09
```

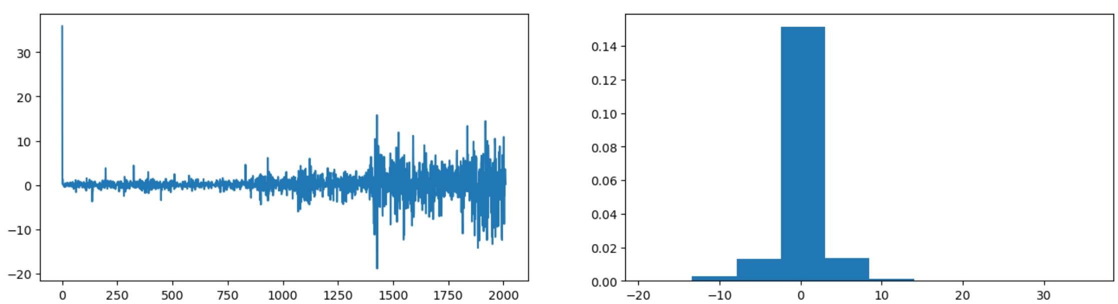
```
Prob(Q):                              0.41   Prob(JB):
0.00
Heteroskedasticity (H):              42.50   Skew:
-0.64
Prob(H) (two-sided):                  0.00   Kurtosis:
10.94
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
```

[48]:

```python
# Plot residual errors
residuals = pd.DataFrame(result.resid)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 4))

ax1.plot(residuals)
ax2.hist(residuals, density=True)
```

[48]:

```
(array([2.72469915e-04, 2.54305254e-03, 1.31693792e-02, 1.51493273e-01,
        1.37143191e-02, 1.27152627e-03, 1.81646610e-04, 0.00000000e+00,
        0.00000000e+00, 9.08233051e-05]),
 array([-18.85221279, -13.37985201,  -7.90749123,  -2.43513045,
          3.03723033,   8.50959111,  13.98195188,  19.45431266,
         24.92667344,  30.39903422,  35.871395  ]),
 <BarContainer object of 10 artists>)
```



[49]:

```python
step = 30

forecast = result.get_forecast(steps=step)
fc = forecast.predicted_mean
se = forecast.se_mean
conf = forecast.conf_int()
```

[50]:
```
fc = pd.Series(fc, index=test[:step].index)
lower = pd.Series(conf.iloc[:, 0], index=test[:step].index)
upper = pd.Series(conf.iloc[:, 1], index=test[:step].index)
```

[51]:
```
plt.figure(figsize=(16, 8))
plt.plot(test[:step], label="actual")
plt.plot(fc, label="forecast")
plt.fill_between(lower.index, lower, upper, color="k", alpha=0.1)
plt.title("Forecast vs Actual")
plt.legend(loc="upper left")
```

[51]: <matplotlib.legend.Legend at 0x2195242a310>