۰۳_TAR_model

۲۹ مهر ۱۴۰۱

۱ مدل ترش‌هولد برای سهام مایکروسافت

[1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot

from statsmodels.tsa.stattools import adfuller, pacf
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from bds import bds

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

ابتدا فایل اطلاعات قیمتی را که در تمرین اول دریافت کردیم، می‌خوانیم:

[2]:
```
df_msft = pd.read_excel('./excel_files/01_NYSE_prices.xlsx', sheet_name='MSFT')
```

[3]:
```
df_msft[['Date', 'Adj Close']]
```

[3]:

	Date	Adj Close
0	2017-01-03	57.807819
1	2017-01-04	57.549183

```
2     2017-01-05   57.549183
3     2017-01-06   58.047997
4     2017-01-09   57.863251
…          …           …
1452 2022-10-10  229.250000
1453 2022-10-11  225.410004
1454 2022-10-12  225.750000
1455 2022-10-13  234.240005
1456 2022-10-14  228.559998

[1457 rows x 2 columns]
```

۱.۱ آماده‌سازی داده

سری زمانی قیمت‌های تعدیل شده را می‌سازیم:

[5]:
```python
prices_series = df_msft.set_index('Date')['Adj Close']
```

برای گپ‌های موجود به ترتیب این کارها را می‌کنیم:

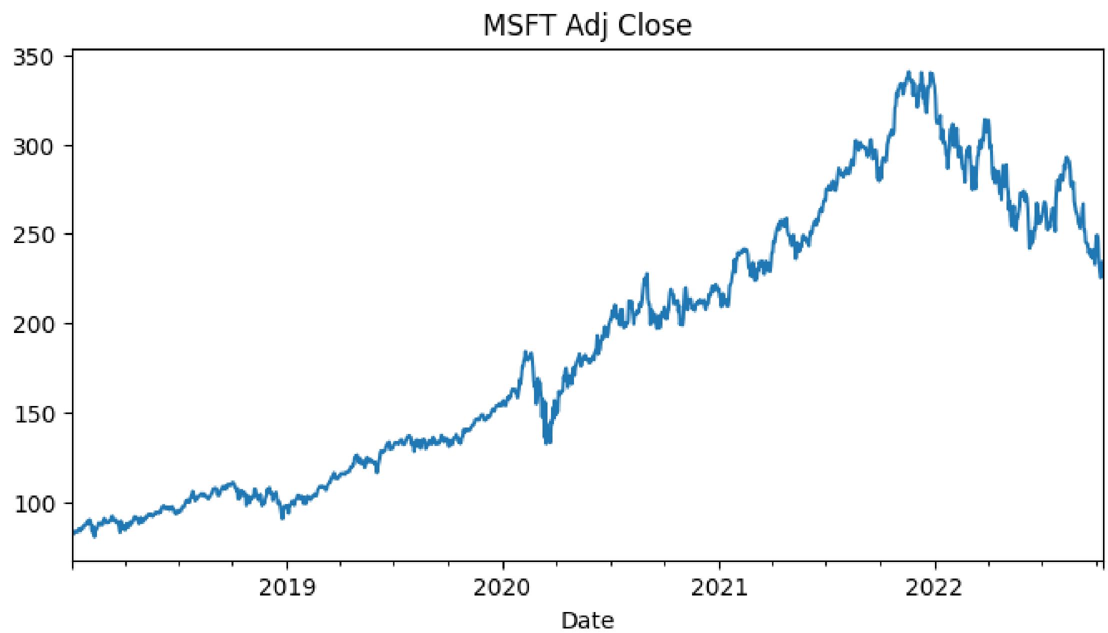
- فرکانس سری زمانی را روزانه می‌کنیم.
- با متد ffill مقادیر نال به وجود آمده را پر می‌کنیم.
- چون برای همه شنبه‌ها و یکشنبه‌ها، دیتایی در دسترس نبوده، بنابراین این دو روز را از سری زمانی حذف می‌کنیم.

[6]:
```python
prices_series = prices_series['2018':].asfreq('1D').ffill()
prices_series = prices_series[prices_series.index.weekday<5]
```

نگاهی به سری زمانی می‌اندازیم. این سری ناماناست.

[7]:
```python
prices_series.plot(title='MSFT Adj Close', figsize=(8, 4))
```

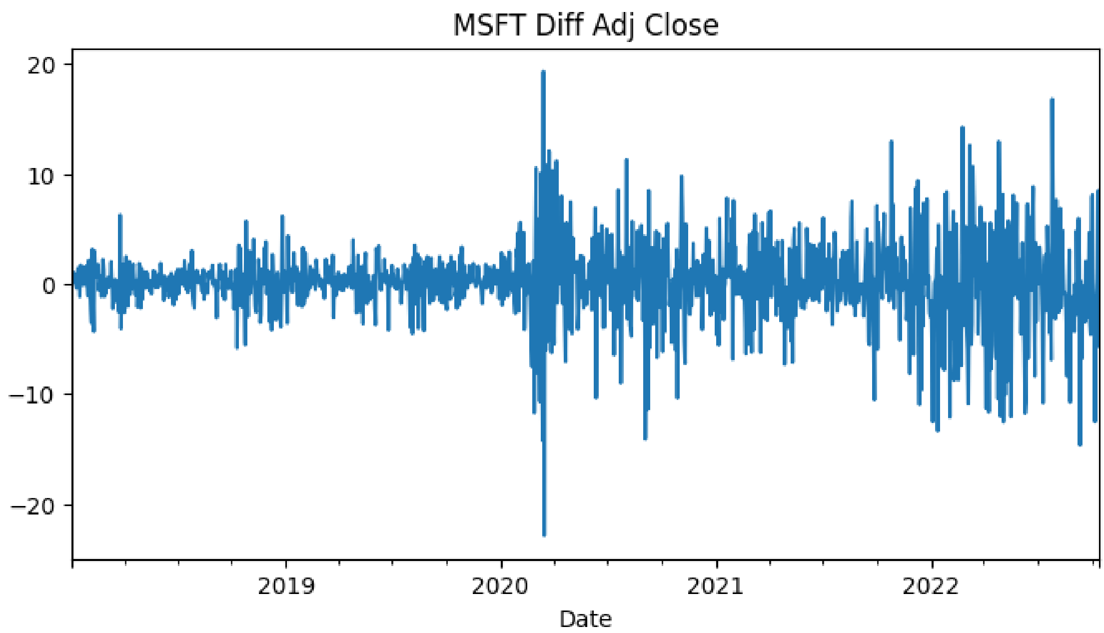
[7]:
```
<AxesSubplot:title={'center':'MSFT Adj Close'}, xlabel='Date'>
```



از سری دیفرنس می‌گیریم و نمودار آن را رسم می‌کنیم. به نظر می‌رسد با یک بار دیفرنس گرفتن، سری مانا شده است.

[9]:
```
diff_prices = prices_series.diff()
diff_prices = diff_prices['2018':]
diff_prices.plot(title='MSFT Diff Adj Close', figsize=(8, 4))
```

[9]: <AxesSubplot:title={'center':'MSFT Diff Adj Close'}, xlabel='Date'>



تست ADF را برای مانایی اجرا می‌کنیم. خروجی دوم، مقدار p-value را برای این تست نمایش نمی‌دهد. سری ماناست..

```
[10]: adfuller(diff_prices.reset_index()['Adj Close'].dropna())
```

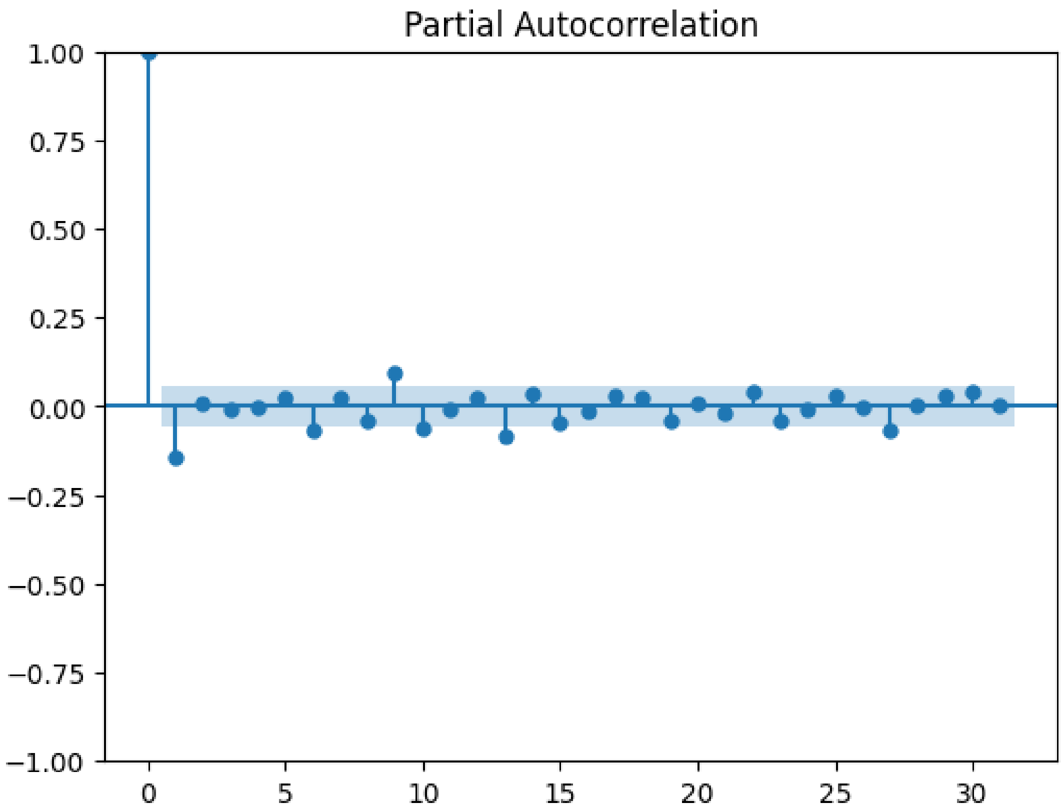
```
[10]: (-10.606369034110845,
 5.98823668149399e-19,
 12,
 1235,
 {'1%': -3.4356560275160835,
  '5%': -2.8638831211270817,
  '10%': -2.568017509711682},
 6654.980126159714)
```

۲.۱ مدل AR

اگر بتونیم سری را با مدل AR مدل کنیم و مانده‌های مدل نیز i.i.d باشند، می‌توان نتیجه گرفت که سری رفتار غیرخطی ندارد. بنابراین ابتدا باید مطمئن شویم که مانده‌های مدل AR یک سری i.i.d نیست.

تابع pacf سری را نمایش می‌دهیم:

```
[24]: plot_pacf(diff_prices.reset_index()['Adj Close'].dropna())
      pyplot.show()
```



برای پیدا کردن بهترین مرتبه مدل AR این تابع را می‌نویسیم:

```
[25]: def auto_ar_model(values, max_p=12):
          best_orders = None
          _best_aic = np.Inf
          for p in range(1, max_p+1):
              model = ARIMA(values, order=(p,0,0))
              results = model.fit()
              if results.aic < _best_aic:
                  _best_aic = results.aic
                  best_orders = model.order
          return best_orders
```

به کمک تابع بالا، بهترین مرتبه مدل AR را به دست می‌اوریم. به نظر می‌رسید مدل AR(۱۰) بهترین مدل از نظر معیار AIC است.

[14]:
```
diff_prices2 = diff_prices.reset_index()['Adj Close'].dropna()
best_order = auto_ar_model(diff_prices2)
best_order
```

[14]:
```
(10, 0, 0)
```

مدل‌سازی را به کمک بهترین مرتبه انجام می‌دهیم:

[15]:
```
model = ARIMA(diff_prices2, order=best_order)
results = model.fit()
```

نتایج مدل AR(۱۰) در زیر به طور خلاصه آورده شده است.

ضرایب لگ‌های اول، ششم، هشتم و دهم معنادارند.

[16]:
```
results.summary()
```

[16]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results
==============================================================================
Dep. Variable:              Adj Close   No. Observations:                 1248
Model:                ARIMA(10, 0, 0)   Log Likelihood               -3372.774
Date:                Fri, 21 Oct 2022   AIC                           6769.548
Time:                        21:15:25   BIC                           6831.099
Sample:                             0   HQIC                          6792.689
                               - 1248
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.1188      0.093      1.273      0.203      -0.064       0.302
ar.L1         -0.1297      0.019     -6.724      0.000      -0.167      -0.092
```

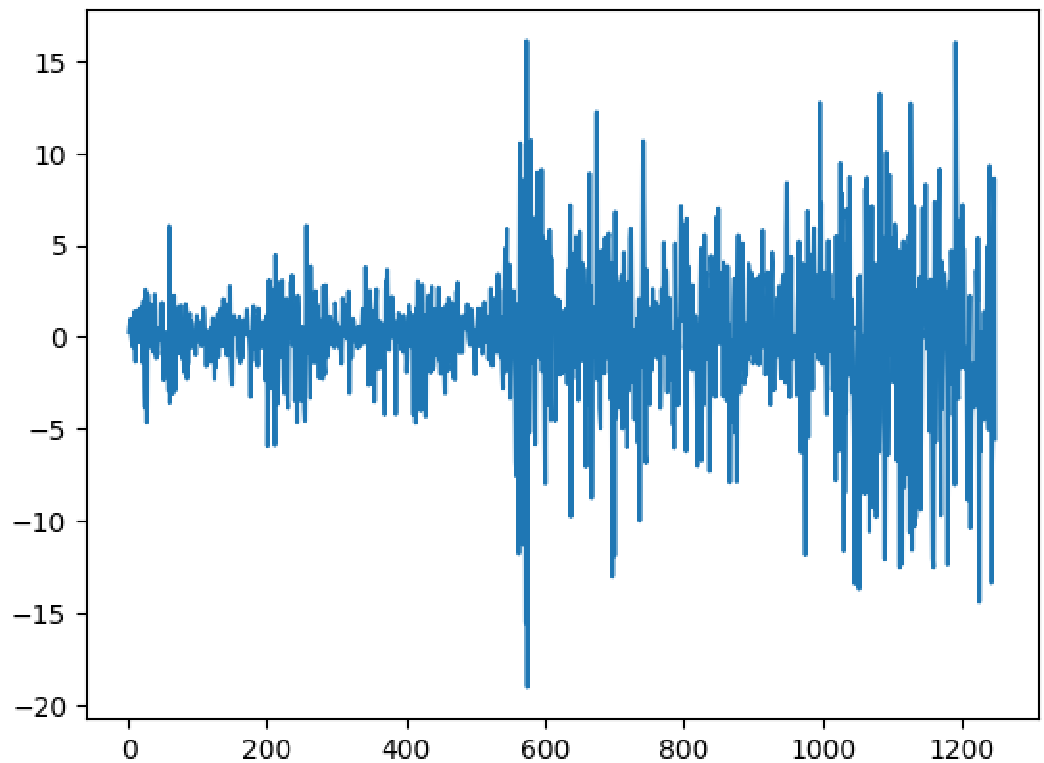
```
ar.L2          0.0013      0.021      0.062      0.950      -0.039       0.042
ar.L3         -0.0026      0.021     -0.125      0.900      -0.043       0.038
ar.L4         -0.0042      0.021     -0.202      0.840      -0.045       0.037
ar.L5          0.0118      0.021      0.574      0.566      -0.029       0.052
ar.L6         -0.0637      0.021     -3.096      0.002      -0.104      -0.023
ar.L7          0.0177      0.022      0.805      0.421      -0.025       0.061
ar.L8         -0.0267      0.024     -1.128      0.260      -0.073       0.020
ar.L9          0.0893      0.022      4.108      0.000       0.047       0.132
ar.L10        -0.0622      0.020     -3.042      0.002      -0.102      -0.022
sigma2        13.0273      0.333     39.131      0.000      12.375      13.680
================================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
667.99
Prob(Q):                              0.99   Prob(JB):
0.00
Heteroskedasticity (H):               8.78   Skew:
-0.45
Prob(H) (two-sided):                  0.00   Kurtosis:
6.47
================================================================================
===

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

نمودار مانده را می‌کشیم:

[17]:
```
results.resid.plot()
```

[17]: <AxesSubplot:>



تست BDS را برای i.i.d بودن مانده اجرا می‌کنیم. فرض صفر این تست، i.i.d بودن فرایند است. مقادیر p−value بسیار کوچک‌اند و بنابراین می‌توانیم فرض صفر را رد کنیم. این موضوع موجب می‌شود که بتوانیم امکان روابط غیرخطی در فرایند را بررسی کنیم:

[18]:
```
bds(results.resid, 3)
```

[18]:
```
(array([ 9.98937247, 12.89277953]), array([1.69652321e-23, 4.94296165e-38]))
```

۳.۱ مدل TAR

مدل TAR در هیچ پکیج پایتونی پیاده‌سازی نشده است. بنابراین باید این کار را شخصا انجام دهیم.

تابع زیر یک سری زمانی را به همراه لیستی از ترش‌هولدها به عنوان ورودی دریافت می‌کند و برای همه مقادیر ممکن، مدل‌سازی را انجام می‌دهد. سپس RMSE هر مدل را در فایل 03_TAR_RMSE_LOG.txt می‌نویسد تا بعدا مورد استفاده قرار بگیرند:

[20]:
```
import itertools
```

```python
def switching_treshold_model(data, tresholds, max_p=5):
    data = pd.DataFrame(data)
    col_name = data.columns[0]

    for p in range(1, max_p+1):
        model = ARIMA(data[col_name], order=(p, 0, 0))
        results = model.fit()
        data[f'ar_{p}'] = results.predict()


    iterate_matrix = []
    for d in range(len(tresholds)+1):
        iterate_matrix = iterate_matrix.__add__([range(1, max_p+1)])

    f = open('03_TAR_RMSE_LOG.txt', 'a')

    for orders in itertools.product(*iterate_matrix):
        switching_pred = data.loc[data[col_name].shift(1) <= tresholds[0],␣
↪f'ar_{orders[0]}']
        for i in range(len(tresholds)):
            lower_band = tresholds[i]
            upper_band = tresholds[i+1] if len(tresholds)>i+1 else np.Inf
            ar_tmp = data.loc[(data[col_name].shift(1) > lower_band) &␣
↪(data[col_name].shift(1) <= upper_band), f'ar_{orders[i+1]}']
            switching_pred = switching_pred.append(ar_tmp).sort_index()

        err = mean_squared_error(data[col_name][1:], switching_pred)
        f.write(f'{orders} = {err}\n')
    f.close()
```

تابع بالا را برای دو ترش‌هولد ران می‌کنیم:

- ترش‌هولد برابر با صفر: زمانی که در روز معاملاتی قبل، بازده مثبت یا منفی بوده
- ترش‌هولد دوتایی: زمانی که بازده نزدیک صفر بوده یا با آن فاصله مثبت/منفی داشته است.

[21]:
```
switching_treshold_model(diff_prices2, tresholds=[0])
switching_treshold_model(diff_prices2, tresholds=[-1.8, 1.8])
```

لاگ را می‌خوانیم و مدلی که کم‌ترین RMSE داشته را به عنوان مدل نهایی انتخاب می‌کنیم:

[22]:
```
with open('03_TAR_RMSE_LOG.txt') as file:
    lines = file.readlines()
    lines = [line.rstrip() for line in lines]

tar_orders_rmse = [[x.split(' = ')[0], x.split(' = ')[1]] for x in lines]
least_rmse = np.Inf
for i, elem in enumerate(tar_orders_rmse):
    if float(elem[-1]) < least_rmse:
        best_rmse_index = i
        least_rmse = float(elem[-1])
```

[23]:
```
tar_orders_rmse[best_rmse_index][0]
```

[23]:
```
'(3, 5, 5)'
```

با اینکه مدل ۲ ترش‌هولد دارد (سه بخشی است) اما بخش دوم و سوم یک مدل مشترک را نمایش می‌دهند: AR(۵)

بنابراین مدل نهایی دو بخشی خواهد بود. مقدار ترش‌هولد -۱.۸ و به ترتیب مدل‌های AR(۳) و AR(۵) برای این دو بخش مناسب خواهند بود.