به نام خداوند

تمرین سوم
اقتصادسنجی پیشرفته
بهار ۱۴۰۱

علیرضا جمالی

* در ساخت این گزارش از مبدل لاتک در محیط ژوپیتر پایتون استفاده شده است. در بعضی از جاهای کد، کاراکترهای فارسی به درستی تبدیل نشده‌اند که در بالای آن، توضیحات مورد نیاز آورده شده است.

۰۱_nlntest

۲۹ آبان ۱۴۰۱

۱ آزمون خطی بودن

[113]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nlntest
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import bds
import yfinance as yf       # https://pypi.org/project/yfinance/
import pytse_client as tse # https://pypi.org/project/pytse-client/
```

۱.۱ دریافت داده

این بخش در تمرین شماره یک انجام گرفته و از همان کدها استفاده می‌شود.

دیتای اینتل و والمارت از api یاهوفایننس دریافت می‌شود:

[2]:
```
tickers = ['INTC', 'WMT']
data_nse = yf.download(tickers, group_by = 'ticker', start="2017-01-01",␣
↪end="2022-11-19")
```

```
[**********100%**********]  2 of 2 completed
```

دیتای دو سهم ایرانی شپنا و وبصادر نیز از سایت tse اسکریپ می‌شود:

[118]:
```
tickers = ['   ', '   '[
prices_dict = tse.download(symbols=tickers, adjust=True)
prices_dict_reform = {(outerKey, innerKey):
                              values for outerKey, innerDict
                              in prices_dict.items() for innerKey, values
                              in innerDict.iteritems()}
data_tse = pd.DataFrame(prices_dict_reform)
d = {'   ': 'Khodro', '   ': 'Shepna'}
data_tse = data_tse.rename(columns=d, level=0)
```

۲.۱ آشنایی با داده

نگاهی به دیتای شرکت اینتل می‌اندازیم:

[119]:
```
data_nse['INTC'][['Open', 'High', 'Low', 'Close', 'Adj Close']].tail()
```

[119]:

Date	Open	High	Low	Close	Adj Close
2022-11-14	30.340000	30.990000	30.180000	30.350000	30.350000
2022-11-15	31.100000	31.340000	30.170000	30.709999	30.709999
2022-11-16	30.110001	30.230000	29.440001	29.530001	29.530001
2022-11-17	29.070000	29.950001	29.000000	29.889999	29.889999
2022-11-18	30.260000	30.260000	29.610001	29.870001	29.870001

نگاهی به دیتای شرکت خودرو می‌اندازیم:

[141]:
```
cols = ['open', 'high', 'low', 'close', 'adjClose']
data_tse['Khodro'].set_index('date')[cols].tail()
```

[141]:

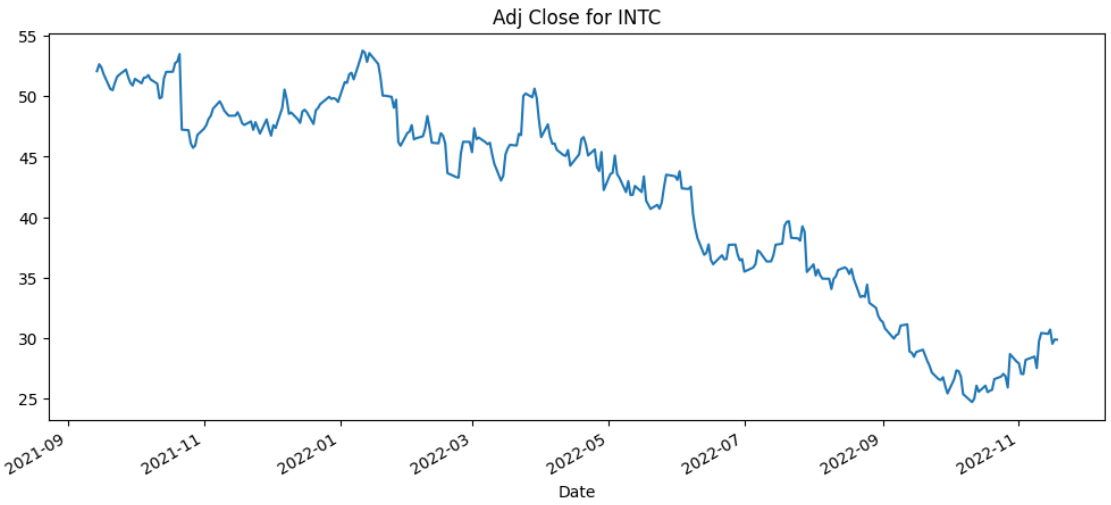
date	open	high	low	close	adjClose
2022-11-12	2230.0	2240.0	2157.0	2163.0	2188.0
2022-11-13	2152.0	2204.0	2123.0	2148.0	2159.0
2022-11-14	2160.0	2183.0	2109.0	2110.0	2135.0
2022-11-15	2173.0	2219.0	2151.0	2179.0	2180.0

```
2022-11-16  2189.0  2228.0  2152.0  2192.0     2200.0
```

نمودار شرکت اینتل را در یک سال آخر رسم می‌کنیم تا با حرکت کلی سهم آشنا شویم:

[121]:
```
data_nse['INTC', 'Adj Close'][-300:].plot(title='Adj Close for INTC',␣
↪figsize=(12,5))
```

[121]:
```
<AxesSubplot:title={'center':'Adj Close for INTC'}, xlabel='Date'>
```



۳.۱ مدل‌سازی

۱.۳.۱ تست ایستا بودن دیفرنس قیمت‌ها

برای اینکه بدانیم آیا سری قیمت‌ها با یک‌بار دیفرنس گرفتن ایستا می‌شوند، تست ADF را انجام می‌دهیم و در صورتی که pvalue این تست کم‌تر از پنج‌درصد باشد، فرض صفر را رد می‌کنیم و سری را ایستا در نظر می‌گیریم.

در صورتی که سری با یک بار دیفرنس گرفتن ایستا شود، بعدا در مدل ARIMA مقدار d را برابر با یک قرار می‌دهیم.

تست ایستا بودن دیفرنس قیمت‌های اینتل:

[123]:
```
diff_adjClose = data_nse['INTC', 'Adj Close'].diff()
adfuller(np.array(diff_adjClose)[1:])[1] < 0.05
```

[123]: True

تست ایستا بودن دیفرنس قیمت‌های والمارت:

```
[124]: diff_adjClose = data_nse['WMT', 'Adj Close'].diff()
       adfuller(np.array(diff_adjClose)[1:])[1] < 0.05
```

[124]: True

تست ایستا بودن دیفرنس قیمت‌های خودرو:

```
[126]: diff_adjClose = data_tse['Khodro', 'adjClose'].diff()
       diff_adjClose = diff_adjClose[~np.isnan(diff_adjClose)]
       adfuller(diff_adjClose) [1] < 0.05
```

[126]: True

تست ایستا بودن دیفرنس قیمت‌های شپنا:

```
[127]: diff_adjClose = data_tse['Shepna', 'adjClose'].diff()
       diff_adjClose = diff_adjClose[~np.isnan(diff_adjClose)]
       adfuller(diff_adjClose) [1] < 0.05
```

[127]: True

همگی سری‌های قیمتی با یک بار دیفرنس گرفتن ایستا می‌شوند؛ بنابراین می‌توانیم در مدل ARIMA مقدار d را برای همگی یک در نظر بگیریم.

برای یافتن بهترین مدل، یک تابع می‌نویسیم. این تابع مدل‌های ARIMA مختلف را امتحان می‌کند و مدلی را که کم‌ترین aic دارد به عنوان خروجی پس می‌دهد.

```
[144]: def best_arima(data):
           p_max = 12
           min_aic = np.inf
           best_model = None
```

```
    for p in range(1, p_max):
        model = ARIMA(data, order=(p, 1, 0)).fit()
        if min_aic > model.aic:
            min_aic = model.aic
            best_model = model

    return best_model
```

۴.۱ نتایج

۱.۴.۱ سهم خودرو

پس از گرفتن مانده بهترین مدل ARIMA تست خطی بودن را اجرا می‌کنیم.

نتیجه: همه آزمون‌های خطی، فرض صفر را رد کرده‌اند. بنابراین سری قیمتی خودرو، رفتاری غیرخطی دارد.

[129]:
```
symbol = 'Khodro'
not_null_data = data_tse[data_tse[symbol, 'adjClose'].notna()][symbol,␣
↪'adjClose']
arim_model = best_arima(not_null_data)
residuals1 = arim_model.resid
nlntest.nlntstuniv(np.array(residuals1))
```

```
------------------ Linearity Test of Univariate time Series--------------------

H0: Model is linear,  PValue of Ramsey Test             1.1102230246251565e-16
H0: Model is linear,  PValue of Keenan Test             1.1102230246251565e-16
H0: Model is linear,  PValue of Tsay Test               1.1102230246251565e-16
H0: Model is linear,  PValue of Terasvirta et al. Test  1.1102230246251565e-16
--------------------------------------------------------------------------------
Ref. Mohammadi S.(2019). Neural network for univariate and multivariate
nonlinearity tests. Stat Anal Data Min: The ASA DataSci Journal.13:50-70.
https://doi.org/10.1002/sam.11441
```

[129]:
```
(array([1.11022302e-16]),
 array([1.11022302e-16]),
```

```
 array([1.11022302e-16]),
 array([1.11022302e-16]))
```

۲.۴.۱ سهم شپنا

پس از گرفتن مانده بهترین مدل ARIMA تست خطی بودن را اجرا می‌کنیم.

نتیجه: همه آزمون‌های خطی، فرض صفر را رد کرده‌اند. بنابراین سری قیمتی شپنا، رفتاری غیرخطی دارد.

[131]:
```
symbol = 'Shepna'
not_null_data = data_tse[data_tse[symbol, 'adjClose'].notna()][symbol,␣
 ↪'adjClose']
arim_model = best_arima(not_null_data)
residuals = arim_model.resid
nlntest.nlntstuniv(np.array(residuals))
```

```
------------------ Linearity Test of Univariate time Series--------------------

H0: Model is linear,  PValue of Ramsey Test             1.1102230246251565e-16
H0: Model is linear,  PValue of Keenan Test             2.9976021664879227e-15
H0: Model is linear,  PValue of Tsay Test               2.9976021664879227e-15
H0: Model is linear,  PValue of Terasvirta et al. Test  1.1102230246251565e-16
------------------------------------------------------------------------------
Ref. Mohammadi S.(2019). Neural network for univariate and multivariate
nonlinearity tests. Stat Anal Data Min: The ASA DataSci Journal.13:50-70.
https://doi.org/10.1002/sam.11441
```

[131]:
```
(array([1.11022302e-16]),
 array([2.99760217e-15]),
 array([2.99760217e-15]),
 array([1.11022302e-16]))
```

۳.۴.۱ سهم اینتل

پس از گرفتن مانده بهترین مدل ARIMA تست خطی بودن را اجرا می‌کنیم.

نتیجه: هیچ‌کدام از آزمون‌های خطی، فرض صفر را رد نکرده‌اند. بنابراین سری قیمتی اینتل، رفتاری خطی دارد.

[137]:
```
symbol = 'INTC'
not_null_data = data_nse[data_nse[symbol, 'Adj Close'].notna()][symbol, 'Adj␣
↪Close']
not_null_data = np.array(not_null_data)
arim_model = best_arima(not_null_data)
residuals = arim_model.resid
nlntest.nlntstuniv(np.array(residuals))
```

```
------------------ Linearity Test of Univariate time Series---------------------

H0: Model is linear,  PValue of Ramsey Test             0.12002599396046165
H0: Model is linear,  PValue of Keenan Test             0.48056564883432895
H0: Model is linear,  PValue of Tsay Test               0.48056564883432984
H0: Model is linear,  PValue of Terasvirta et al. Test  0.02576320329911097
--------------------------------------------------------------------------------
Ref. Mohammadi S.(2019). Neural network for univariate and multivariate
nonlinearity tests. Stat Anal Data Min: The ASA DataSci Journal.13:50-70.
https://doi.org/10.1002/sam.11441
```

[137]:
```
(array([0.12002599]),
 array([0.48056565]),
 array([0.48056565]),
 array([0.0257632]))
```

۴.۴.۱ سهم والمارت

پس از گرفتن مانده بهترین مدل ARIMA تست خطی بودن را اجرا می‌کنیم.

نتیجه: آزمون ترسورتا فرض صفر را کرده است. اما سه آزمون دیگر در سطح اطمینان ۹۵ درصد فرض صفر را رد نکرده‌اند. هر چند تست رمزی نیز بسیار به مقدار بحرانی نیزدیک است. در مورد سری قیمتی والمارت نمی‌توان با قطعیت نظر دارد.

[138]:
```
symbol = 'WMT'
not_null_data = data_nse[data_nse[symbol, 'Adj Close'].notna()][symbol, 'Adj␣
↪Close']
not_null_data = np.array(not_null_data)
```

```
arim_model = best_arima(not_null_data)
residuals = arim_model.resid
nlntest.nlntstuniv(np.array(residuals))
```

```
------------------ Linearity Test of Univariate time Series--------------------

H0: Model is linear,  PValue of Ramsey Test             0.005049658765449827
H0: Model is linear,  PValue of Keenan Test             0.7167640875642718
H0: Model is linear,  PValue of Tsay Test               0.7167640875642678
H0: Model is linear,  PValue of Terasvirta et al. Test  0.0005997956456036402
------------------------------------------------------------------------------
Ref. Mohammadi S.(2019). Neural network for univariate and multivariate
nonlinearity tests. Stat Anal Data Min: The ASA DataSci Journal.13:50-70.
https://doi.org/10.1002/sam.11441
```

[138]:
```
(array([0.00504966]),
 array([0.71676409]),
 array([0.71676409]),
 array([0.0005998]))
```

۰۲_algotrading

۲۹ آبان ۱۴۰۱

۱ معاملات الگوریتمی

در این تمرین، سعی می‌شود تا بر اساس نتایج پیش‌بینی تمرین شماره ۲ یک ربات ترید نوشته شود.

[96]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor
import yfinance as yf       # https://pypi.org/project/yfinance/
import pytse_client as tse # https://pypi.org/project/pytse-client/
```

۱.۱ دریافت داده

این بخش در تمرین شماره یک انجام گرفته و از همان کدها استفاده می‌شود.

دیتای پنج سهم از یاهوفاینانس دریافت می‌شود: اینتل، والمارت، اپل، پپسی و بانک آمریکا

[64]:
```
tickers_nse = ['INTC', 'WMT', 'AAPL', 'PEP', 'BAC']
data_nse = yf.download(tickers_nse, group_by = 'ticker', start="2018-01-01",␣
↪end="2022-11-04")
```

```
[**********100%**********]  5 of 5 completed
```

دیتای پنج سهم بازار بورس تهران دریافت می‌شود: وبصادر، شپنا، صندوق آگاس، فارس و کرمان

نام همه این سهام به معادل فینگلیش تغییر می‌کند تا در کدها راحت‌تر قابل استفاده باشد.

[3]:
```
tickers_tse = ['    ', '   ', '   ', '   ', '    '[
prices_dict = tse.download(symbols=tickers_tse, adjust=True)
prices_dict_reform = {(outerKey, innerKey):
                              values for outerKey, innerDict
                              in prices_dict.items() for innerKey, values
                              in innerDict.iteritems()}
data_tse = pd.DataFrame(prices_dict_reform)
d = {'    ': 'Websader', '   ': 'Shepna', '   ' : 'Agas', '   ' : 'Fars', '    ' :␣
 ↪'Kerman'}
data_tse = data_tse.rename(columns=d, level=0)
```

۲.۱ مدل‌سازی

این قسمت شبیه تمرین شماره ۲ است.

تابعی برای ساخت ماتریس لگ‌ها:

[4]:
```
def lagmat(df, T=21) -> (np.array, np.array):
    X = []
    Y = []
    df['DiffLogP'] = df['LogP'].diff()
    series = df['DiffLogP'].to_numpy()[1:]
    for t in range(len(series) - T):
        x = series[t:t+T]
        X.append(x)
        y = series[t+T]
        Y.append(y)

    X = np.array(X).reshape(-1 ,T)
    Y = np.array(Y)

    return X, Y
```

تابعی برای دریافت داده آموزش و تست:

[5]:
```
def get_train_test_set(df, test_count=20) -> (pd.DataFrame, pd.DataFrame):
    train_set = df[:-test_count]
```

```
    test_set = df[-test_count:]
    return train_set, test_set
```

تابع زیر در تمرین شماره دو نوشته شده است.

این تابع، سری قیمت‌ها و مدل موردنظر را به عنوان ورودی می‌گیرد و سه پیش‌بینی انجام می‌دهد: تک‌گام، چندگام استاتیک و چندگام دینامیک

مدل‌سازی بر روی لگاریتم قیمت‌ها صورت می‌گیرد.

[6]:
```
def one_step_and_multistep_forecast(df, model, tag, test_count=20) -> pd.
↪DataFrame:
    lags = 21   # number of lags to pass into lagmat function
    X, Y = lagmat(df, lags)

    # splitting the lagmat output to train and test
    x_train, y_train = X[:-test_count], Y[:-test_count]
    x_test, y_test = X[-test_count:], Y[-test_count:]
    # getting the train_set (different from x_train with 21 lags)
    train_set, _ = get_train_test_set(df)

    # fitting the model that was passed into the function with x_train and␣
↪y_train
    model.fit(x_train, y_train)

    # storing the index of train and test dataset
    train_idx = df.index <= train_set.index[-1]
    test_idx = ~train_idx
    train_idx[:lags+1] = False

    df = pd.DataFrame(df)

    ### one step forecast
    # wee need to predict and undiffernce the result
    prev = df['LogP'].shift(1)
    df.loc[train_idx, f'{tag}_1step_train'] = \
        prev[train_idx] + model.predict(x_train)
```

```
    df.loc[test_idx, f'{tag}_1step_test'] = \
        prev[test_idx] + model.predict(x_test)

    ## multistep static forecast
    last_train = train_set.iloc[-1]['LogP']
    p = model.predict(x_test)
    df.loc[test_idx, f'{tag}_multistep_test_static'] = \
        last_train + np.cumsum(p)

    # multistep dynamic forecast
    multistep_predictions = []
    last_x = x_test[0]
    while len(multistep_predictions) < test_count:
        p = model.predict(last_x.reshape(1,-1))[0]

        multistep_predictions.append(p)

        last_x = np.roll(last_x, -1)
        last_x[-1] = p

    last_train = train_set.iloc[-1]['LogP']
    df.loc[test_idx, f'{tag}_multistep_test'] = \
        last_train + np.cumsum(multistep_predictions)

    return df
```

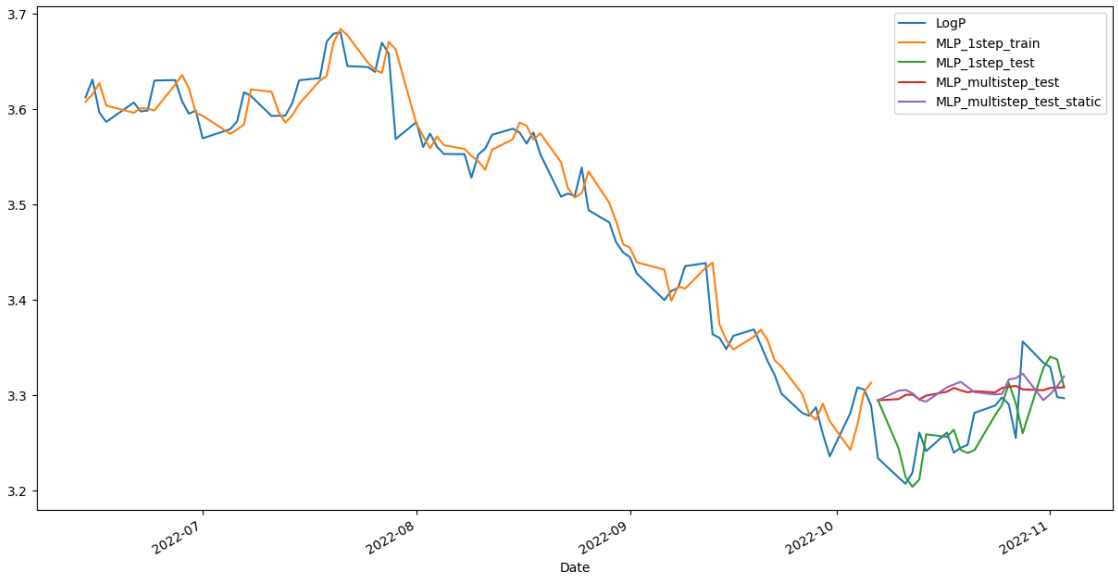
تابعی برای رسم نمودار پیش‌بینی‌ها

[7]:
```
def plot(df, tag):
    df[['LogP', f'{tag}_1step_train', f'{tag}_1step_test',
        f'{tag}_multistep_test', f'{tag}_multistep_test_static']][-100:].
↪plot(figsize=(15,8))
```

یک پیش‌بینی برای اینتل انجام می‌دهیم و نمودار خروجی را می‌کشیم (مشابه تمرین پیشین):

[68]:
```
df_data = data_nse['INTC']
df_data = df_data[['Adj Close']]
df_data = df_data.rename({'Adj Close': 'P'}, axis=1)
df_data['LogP'] = np.log(df_data['P'])

df_data = one_step_and_multistep_forecast(df_data, MLPRegressor(), "MLP")
plot(df_data, tag="MLP")
```



۳.۱ معاملات الگوریتم

تابعی برای معاملات الگوریتمی می‌نویسم.

در صورتی که پیش‌بینی بازده فردا، مثبت باشد آنگاه دستور خرید صادر می‌شود و در صورتی که منفی باشد، دستور فروش (برای بازار نیویورک و نه تهران) صادر می‌شود.

پس از هر روز، بازده مربوط به معامله آن روز محاسبه می‌شود و سری تجمعی بازده‌های این الگوریتم و سری تجمعی بازده‌های واقعی به دیتافریم ورودی اضافه می‌شود و در خروجی تابع قرار می‌گیرد.

[99]:
```
def algo_trade(df, is_short_sell=True):
    df['is_going_up'] = df['LogP'] < df['MLP_1step_test'].shift(-1)
    df['ret'] = df['P'].pct_change().shift(-1)
    if is_short_sell:
```

```
        df['buy_sell'] = -1
    else:
        df['buy_sell'] = 0
    df.loc[df['is_going_up'], 'buy_sell'] = 1
    df['algo_ret'] = df['ret']*df['buy_sell']
    df['algo_ret_cumsum'] = -df.iloc[-20]['algo_ret'] + np.cumsum(df[-20:
↪]['algo_ret'])
    df['actual_ret_cumsum'] = -df.iloc[-20]['ret'] + np.cumsum(df[-20:]['ret'])

    return df
```

برای پنج سهم بورس نیویورک، تابع بالا فراخوانی می‌شود و نمودار بازده تحقق‌یافته و بازده معاملات الگوریتمی رسم می‌شود.

[87]:
```
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(15, 14))
for ax, ticker in zip(axes.flat, tickers_nse):

    df_data = data_nse[ticker]
    df_data = df_data[['Adj Close']]
    df_data = df_data.rename({'Adj Close': 'P'}, axis=1)
    df_data['LogP'] = np.log(df_data['P'])

    df_data = one_step_and_multistep_forecast(df_data, MLPRegressor(), "MLP")

    x = algo_trade(df=df_data)[['algo_ret_cumsum', 'actual_ret_cumsum']].plot(␣
↪ax=ax)
    x.set_title(ticker, x=0.5, y=0.90)
plt.show()
```

در مجموع به نظر می‌رسد مدل و الگوریتم معاملاتی، موفق عمل نکرده‌اند.

همان کار بالا را برای پنج سهم دریافت‌شده بازار بورس تهران انجام می‌دهیم.

[101]:
```
tickers = list(d.values())
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(15, 14))
for ax, ticker in zip(axes.flat, tickers):

    df_data = data_tse[ticker]
    df_data = df_data.rename({'adjClose': 'Adj Close'}, axis=1)
    df_data = df_data[['Adj Close']].dropna()
    df_data = df_data.rename({'Adj Close': 'P'}, axis=1)
    df_data['LogP'] = np.log(df_data['P'])
```

```
    df_data = one_step_and_multistep_forecast(df_data, MLPRegressor(), "MLP")

    x = algo_trade(df=df_data, is_short_sell=False)[['algo_ret_cumsum',␣
↪'actual_ret_cumsum']].plot( ax=ax)
    x.set_title(ticker, x=0.5, y=0.90)
plt.show()
```

نتایج این مدل والگوریتم برای بازار بورس تهران، به نظر بهتر می‌رسد. هر چند که با چند بار ران کردن کد، نتایج متفاوت می‌شوند و نمی‌توان در حالت کلی به این الگوریتم و مدل اعتماد کرد.