امیرعلی فرازمند

۹۹۵۲۲۳۲۹

گزارش تمرین پنجم

```python
# imports
import pandas as pd
import numpy as np
import yfinance as yf
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
from pytz import timezone
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from pandas.plotting import register_matplotlib_converters
from statsmodels.tsa.vector_ar.vecm import coint_johansen
from hurst import compute_Hc
register_matplotlib_converters()

```

[67] ✓ 0.0s Python

توابع برای محاسبه ی hurst و half-life:

پ.ن: برای محاسبه ی hurst تابع آماده بود اما ترجیح داده شد که با تابع نوشته شده کار شود.

```python
def calculate_half_life(series):
    # Add a small constant to series
    # series = series + 1e-10

    delta_series = np.log(series).diff().dropna()
    lagged_series = delta_series.shift(1).dropna()
    lagged_series = sm.add_constant(lagged_series)
    delta_series_lagged = delta_series[1:]  # Make sure delta_series_lagged is the same length as lagged_series

    # Check if delta_series_lagged and lagged_series are not empty
    if delta_series_lagged.size > 0 and lagged_series.size > 0:
        model = sm.OLS(delta_series_lagged, lagged_series)
        res = model.fit()

        # Calculate half-life
        half_life = -np.log(2) / res.params['const']
        return half_life
    else:
        return "Cannot calculate half-life because the series is empty."

```

✓ 0.0s Python

```python

# Function to calculate Hurst exponent
def hurst_exponent(ts):
    lags = range(2, 20)
    tau = [np.std(np.subtract(ts[lag:], ts[:-lag])) for lag in lags]
    hurst = np.polyfit(np.log(lags), np.log(tau), 1)[0]
    return hurst

```

✓ 0.0s Python

از آنجا که yahoo finance در interval های ۴ساعته به ما دیتا نمیدهد، خودمان با استفاده از interval ۱ساعته دیتافریم مدنظرمان را میسازیم:

```python
def download_4h_data(symbols, start_date, end_date):

    try:
        data = yf.download(symbols, start=start_date, end=end_date, interval='1h')['Close']
        # Resample to 4-hour intervals
        data = data.resample('4h').last()
    except:
        SyntaxError("Exception in downloading data")
    data.replace([np.inf, -np.inf], np.nan, inplace=True)
    data.dropna(inplace=True)
    # data = data.reset_index()
    # timestamp_column_name = data.columns[0]  # Assuming the timestamp is the first column
    # data = data[[timestamp_column_name, 'Close']]
    # data.columns = ['timestamp', 'price']
    # print(symbol, start_date, end_date, interval, data.shape[0])
    return data
```

0.0s Python

تابع برای پیدا کردن cointegration با p-value زیر ۵٪ میان لیست crypto. با استفاده از تابع Johansen:

پ.ن: ۲متغیری که در آخر return میشوند یکی برای شمارش این است که ۱۰ ترکیب را پیدا کنیم و دیگری هم برای این است که در آخر کار آن را در فایلی ذخیره کنیم(برای تمرین بعدی گفته شده که نیاز داریم).

```python
def find_cointegration(cryptos, start_date, end_date,counter, res_str=""):
    data = download_4h_data(cryptos, start_date=start_date, end_date=end_date)

    result = coint_johansen(data.values, det_order=0, k_ar_diff=1)
    eigenvectors = result.evec
    # print(eigenvectors)
    for i in range(eigenvectors.shape[1]):
        # print('**')
        stationary_series = np.dot(data.values, eigenvectors[:, i])
        result = adfuller(stationary_series)
        if result[1] < 0.05:
            hurst = hurst_exponent(stationary_series)
            half_life = calculate_half_life(pd.Series(stationary_series))
            print(*cryptos )
            print(*eigenvectors[:, i])
            print(f'P-Value: {result[1]}')
            print(f'Hurst Exponent: {hurst}')
            print(f'Half-Life: {half_life}')
            print('-----------------------------------------------------------------------------------------------------')
            res_str += str(cryptos).replace('[','').replace(']','') +'\n'
            res_str +=str(eigenvectors[:, i]).replace('[','').replace(']','') + '\n'
            res_str += "----\n"
            counter +=1
    return res_str, counter
```

0.0s Python

سل اصلی که با فراخوانی تابع بالا به ما ۱۰ ترکیب را نمایش میدهد:

```python
res_str = ""
cryptos = ['BTC-USD', 'ETH-USD', 'USDT-USD', 'SOL-USD', 'USDC-USD', 'STETH-USD', 'BNB-USD', 'XRP-USD', 'DOGE-USD']
start_date = datetime(2022, 11, 1, tzinfo=timezone('Asia/Tehran'))
end_date = datetime(2023, 11, 1, tzinfo=timezone('Asia/Tehran'))
counter = 0
i=0
j=i+1
cryptos_len = len(cryptos)
while( counter <10):
    crypto_group = [cryptos[i], cryptos[j]]
    res_str, counter = find_cointegration( crypto_group, start_date, end_date, counter, res_str)
    j += 1
    if j==cryptos_len:
        i +=1
        j =i+1

```

نمونه خروجی سل:

```
[**********100%**********]  2 of 2 completed
ETH-USD STETH-USD
0.11448220741591973 -0.11140246135193305
P-Value: 0.0039930283494715784
Hurst Exponent: 0.1552399935551251
Half-Life: -5008.190345471643
--------------------------------------------------------------------------------------------------------------
[**********100%**********]  2 of 2 completed
[**********100%**********]  2 of 2 completed
[**********100%**********]  2 of 2 completed
ETH-USD DOGE-USD
78.91366165704673 0.001539425572830436
P-Value: 0.0003172277544472306
Hurst Exponent: 0.494621609528452
Half-Life: 3386.37709447098
--------------------------------------------------------------------------------------------------------------
```

ذخیره کردن res_str در یک فایل txt. :

```
1  file_path = "output.txt"
2  with open(file_path, 'w') as file:
3      file.write(res_str)
```

✓ 0.0s Python

src.ipynb | output.txt | prj.ipynb | new_attempt.ipynb

output.txt

```
'BTC-USD', 'USDT-USD'
1.64642604e-05 -9.75239265e+02
----
'BTC-USD', 'SOL-USD'
0.00025146 -0.23745998
----
'BTC-USD', 'USDC-USD'
1.12410297e-05 -2.61601355e+02
----
'BTC-USD', 'XRP-USD'
2.34532559e-04 -1.50526951e+01
----
'BTC-USD', 'DOGE-USD'
8.90779426e-05 8.62475789e+01
----
'ETH-USD', 'USDT-USD'
6.05030251e-04 -9.79749463e+02
----
'ETH-USD', 'USDC-USD'
1.30019273e-04 -2.61425629e+02
----
'ETH-USD', 'STETH-USD'
0.11448221 -0.11140246
----
'ETH-USD', 'DOGE-USD'
7.89136617e+01 1.53942557e-03
----
'USDT-USD', 'SOL-USD'
2.06361921e-02 -9.73833195e+02
----
```

*تابع find_cointegration میتواند بین بالای چند رمز ارز cointegeration پیدا کند، لزومی ندارد ۲تا باشد(چون از Johansen استفاده کردیم).

```
1  find_cointegration(['BTC-USD', 'ETH-USD', 'SOL-USD'],start_date, end_date,0,'')
```

✓ 0.2s Python

```
[**********100%**********]  3 of 3 completed
BTC-USD ETH-USD SOL-USD
0.0004857416180584321 -0.006749029071452546 -0.17347195608095864
P-Value: 0.0007332982345492613
Hurst Exponent: 0.5492012637408994
Half-Life: Cannot calculate half-life because the series is empty.
--------------------------------------------------------------------------------------------------------------

("'BTC-USD', 'ETH-USD', 'SOL-USD'\n 0.00048574 -0.00674903 -0.17347196\n----\n",
 1)
```

تست اینکه درست جلو رفتیم روی نمونه ی بالا:

منابع:

کد تمرین های قبلی

Chat-GPT