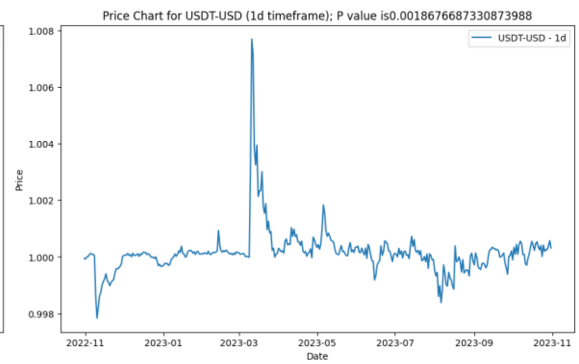
امیرعلی فرازمند

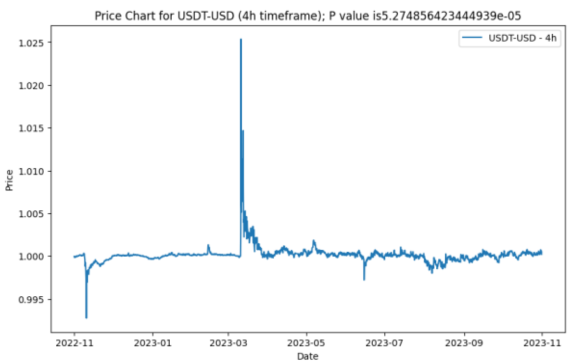
۹۹۵۲۲۳۲۹

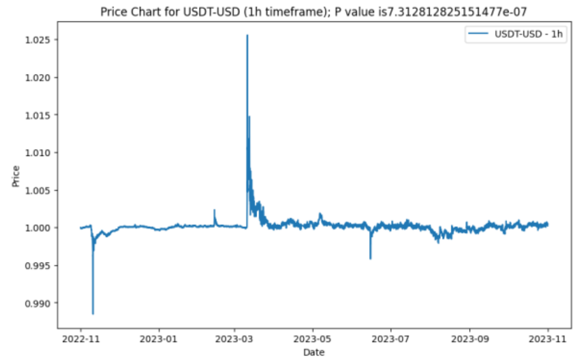
گزارش تمرین سوم

بخش اول، ADF test:

حالت هایی که بالای ۹۰٪ مانا هستند:








```python
# Function to collect historical data for a cryptocurrency using yfinance
def fetch_crypto_data(symbol, start_date, end_date, interval):···
# Function to perform the Augmented Dickey-Fuller (ADF) test
def perform_adf_test(series):···

# Function to plot the price chart
def plot_price_chart(crypto, timeframe, start_date, end_date, p_value=0):···

# Function to find the most stationary combination
def find_most_stationary_combination(cryptos, timeframes, start_date, end_date):···

```

[3] 0.0s Python

```python

# Main execution
top_cryptos = ['BTC-USD', 'ETH-USD', 'USDT-USD', 'BNB-USD', 'XRP-USD']
time_frames = ['1d', '4h', '1h']

start_date = datetime(2022, 11, 1, tzinfo=timezone('Asia/Tehran'))
end_date = datetime(2023, 11, 1, tzinfo=timezone('Asia/Tehran'))

# Find the most stationary combination
stationary_combination = find_most_stationary_combination(top_cryptos, time_frames, start_date, end_date)
print(stationary_combination)
if stationary_combination:
    crypto, timeframe, p_val= stationary_combination
    # Plot the price chart for the most stationary combination
    plot_price_chart(crypto, timeframe, start_date, end_date, p_value=p_val)
else:
    print("No combination found with at least 90% stationarity.")

```

[4] 9.8s Python

```
BTC-USD - 1d: p-value = 0.885654747606342
BTC-USD - 4h: p-value = 0.8395639696983417
BTC-USD - 1h: p-value = 0.8686516375719748
ETH-USD - 1d: p-value = 0.44705609281066927
ETH-USD - 4h: p-value = 0.4502347372954474
ETH-USD - 1h: p-value = 0.433499949452919
```

Price Chart for USDT-USD (1d timeframe); P value is0.0018676687330873988

تنها رمزاری که شرط بالای ۹۰٪ مانا بودن را برآورده میکرد USDT بود که در تایم فریم ۱ساعته بیشترین مانا بودن را دارد(طبق ۲ معیار گفته شده).

[Top cryptos by volume]

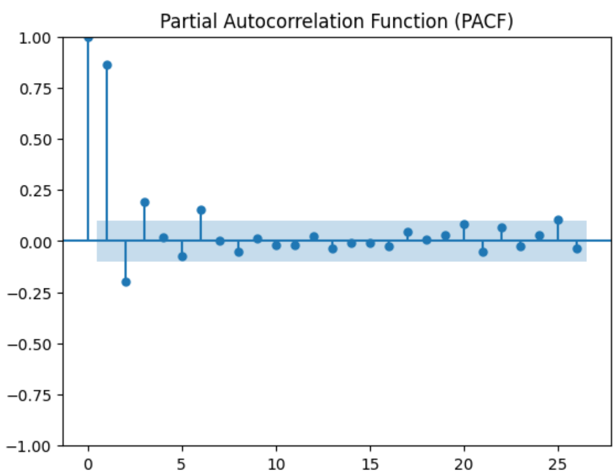
بخش دوم،PACF/ACF:

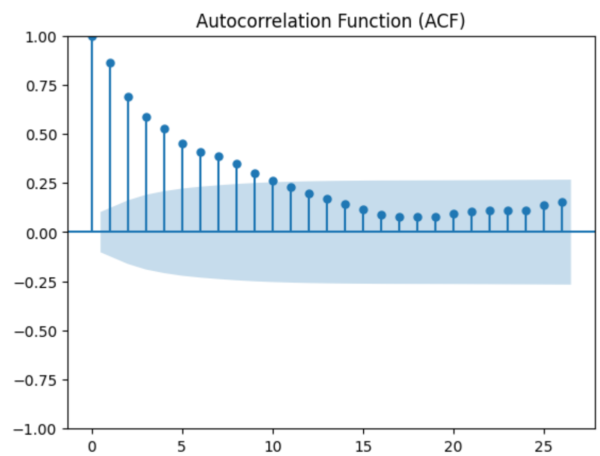
```python
# Plot ACF
def acf_plt(data):
    plot_acf(data)
    plt.title('Autocorrelation Function (ACF)')
    plt.show()

# Plot PACF
def dacf_plt(data):
    plot_pacf(data)
    plt.title('Partial Autocorrelation Function (PACF)')
    plt.show()
```
[5] 0.0s Python

```python

def fetch_crypto_data(symbol, start_date, end_date, priod):
    data = yf.Ticker(symbol)

    tickerDf = data.history(period=priod, start=start_date, end=end_date)
    tickerDf = tickerDf[['Close']]
    print(tickerDf)

    return tickerDf

def get_data_and_plot(crypto, timeframe,start,end):
    data = fetch_crypto_data(crypto, start, end, timeframe)
    acf_plt(data['Close'])
    dacf_plt(data['Close'])
    return data

def predict_and_plot(data:pd.DataFrame, train_end, test_end, p=0,q=0,plot_option=False): ...
```
[19] 0.0s Python

```python

train_end = datetime(2023,10,1, tzinfo=timezone('Asia/Tehran'))
test_end = datetime(2023,11,1, tzinfo=timezone('Asia/Tehran') )
data = get_data_and_plot('USDT-USD', '1d', start_date,end_date)


```
[21] 0.2s Python

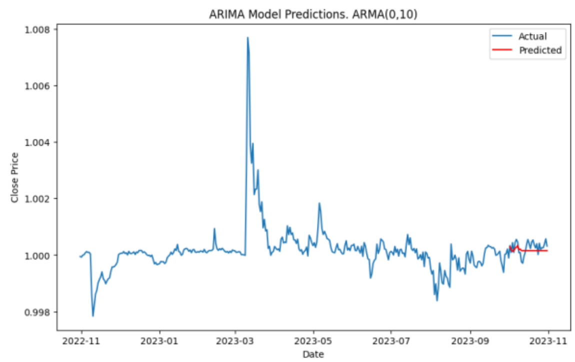
```
                                Close
Date
2022-10-31 00:00:00+00:00  0.999947
2022-11-01 00:00:00+00:00  0.999924
2022-11-02 00:00:00+00:00  0.999996
```

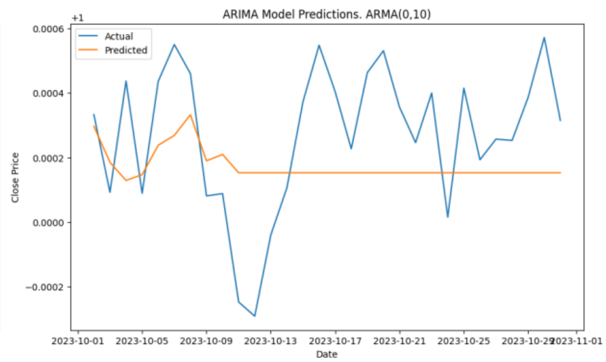




در ادامه مدل های ARMA(3,0), ARMA(5,0), ARMA(6,0), ARMA(0,9), ARMA(0,10), ARMA(0,11), ARMA(5,0), ARMA(3,10)ساخته و پس از predict کردنerror هایشان حساب شده اند که در آن هاMA(10) بهترین عملکرد را داشته.

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.0002      0.000   5352.074      0.000       1.000       1.001
ma.L1          1.0842      0.026     40.981      0.000       1.032       1.136
ma.L2          0.7565      0.042     17.972      0.000       0.674       0.839
ma.L3          0.5182      0.038     13.748      0.000       0.444       0.592
ma.L4          0.5437      0.031     17.569      0.000       0.483       0.604
ma.L5          0.4187      0.029     14.496      0.000       0.362       0.475
ma.L6          0.2971      0.026     11.603      0.000       0.247       0.347
ma.L7          0.3548      0.032     11.185      0.000       0.293       0.417
ma.L8          0.3709      0.038      9.656      0.000       0.296       0.446
ma.L9          0.3082      0.042      7.287      0.000       0.225       0.391
ma.L10         0.2521      0.034      7.488      0.000       0.186       0.318
sigma2      1.841e-07   6.79e-09     27.092      0.000    1.71e-07    1.97e-07
===================================================================================
Ljung-Box (L1) (Q):                   0.15   Jarque-Bera (JB):             21543.05
Prob(Q):                              0.70   Prob(JB):                         0.00
Heteroskedasticity (H):               2.59   Skew:                             4.06
Prob(H) (two-sided):                  0.00   Kurtosis:                        41.44
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 4.74e+16. Standard errors may be unstable.
Mean Absolute Percent Error: 0.00020169
Root Mean Squared Error: 0.00023514
```





پ.ن: طبق پیام داخل گروه، تنها روی تایم فریم ۱روزه کار شده است.

بخش سوم،پیدا کردن p, q:

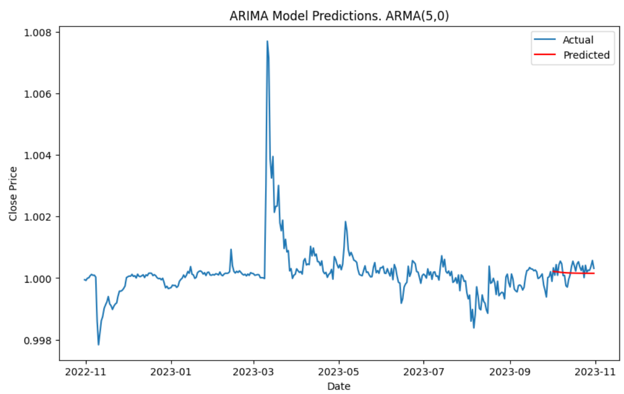
```python
def get_mse_msp(data:pd.DataFrame,train_data, test_data, train_end, test_end,p,q):
    model = ARIMA(train_data, order=(p,0,q))
    model_fit = model.fit()
    pred_start_date = test_data.index[0]
    pred_end_date = test_data.index[-1]
    predictions = model_fit.predict(start=pred_start_date, end=pred_end_date)
    residuals = test_data['Close'] - predictions
    # temp_mape = round(np.mean(abs(residuals/test_data.Close)),8)
    # temp_mse = round(np.sqrt(np.mean(residuals**2)),8)
    temp_mape = np.mean(abs(residuals/test_data.Close))
    temp_mse = np.sqrt(np.mean(residuals**2))

    return (temp_mape, temp_mse)

def get_best_combination(data:pd.DataFrame, train_end, test_end, def_p=0, def_q=0, loop_over_p=False, loop_over_q=False, max_loop = 5):

    mse, mape =99999,999999
    best_combination = {'p':def_p, 'q':def_q}
    train_data = data.loc[:train_end]#.loc???
    test_data = data.loc[train_end + timedelta(days=1):test_end]
    if loop_over_p and loop_over_q:
        for p in range(max_loop):
            for q in range(max_loop):
                temp_mape, temp_mse = get_mse_msp(data, train_data, test_data, train_end,test_end, p,q)
                print(f'ARMA({p},{q}) -> MSE={temp_mse}, MAPE={temp_mape}')
                # if  temp_mse <=mse: #and temp_mape <= mape
                if np.less(temp_mse, mse):
                    mse = temp_mse
                    best_combination['p'] = p
                    best_combination['q'] = q

    elif loop_over_p:
        for p in range(max_loop):
            temp_mape, temp_mse = get_mse_msp(data, train_data, test_data, train_end,test_end, p,def_q)
            print(f'ARMA({p},{def_q}) -> MSE={temp_mse}, MAPE={temp_mape}')
            # if  temp_mse <=mse:#temp_mape <= mape
            if np.less(temp_mse, mse):
                mse = temp_mse
                best_combination['p'] = p
                # best_combination['q'] = def_q

    elif loop_over_q:
        for q in range(max_loop):
            temp_mape, temp_mse = get_mse_msp(data, train_data, test_data, train_end,test_end, def_p,q)
            print(f'ARMA({def_p},{q}) -> MSE={temp_mse}, MAPE={temp_mape}')
            # if  temp_mse <=mse: #temp_mape <= mape
            if np.less(temp_mse, mse):
                mse = temp_mse
                # best_combination['p'] = def_p
                best_combination['q'] = q
    else:
        raise ValueError("looped not over p neither q, or something else")
    return best_combination
```

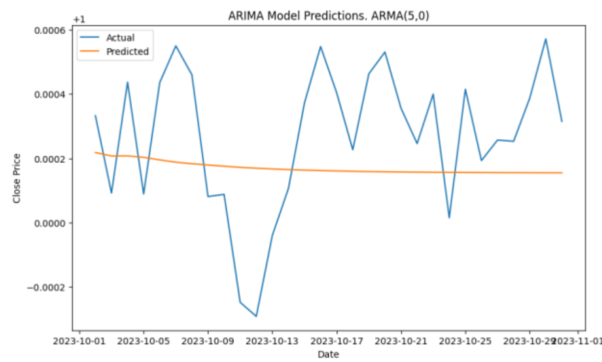
با توابع بالا یکبار روی p، یکبار روی q و یکبار روی هردو لوپ میزنیم ببینیم بهترین مقادیر چه هستند.برای اینکه p,q بهتری پیدا کنیم.مدلی که MSE کمتری دارد را انتخاب شده است. میتوانستیم and ۲معیارگفته شده را اعمال کنیم.

روی AR(5):p

```python
best_comb = get_best_combination(data, train_end,test_end, loop_over_p=True,max_loop=50)
print(best_comb)
predict_and_plot(data, train_end,test_end,p= best_comb['p'],q = best_comb['q'],plot_option=True)
```

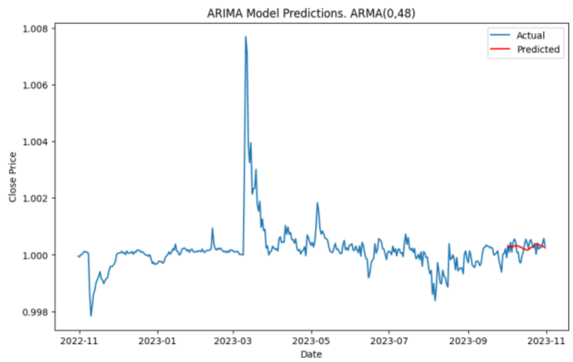
```
{'p': 5, 'q': 0}
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  335
Model:                 ARIMA(5, 0, 0)   Log Likelihood                2126.834
Date:                Wed, 13 Dec 2023   AIC                          -4239.667
Time:                        13:09:59   BIC                          -4212.968
Sample:                    10-31-2022   HQIC                         -4229.023
                         - 09-30-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.0002      0.000   4260.868      0.000       1.000       1.001
ar.L1          1.0925      0.022     49.338      0.000       1.049       1.136
ar.L2         -0.4232      0.023    -18.544      0.000      -0.468      -0.378
ar.L3          0.1887      0.026      7.395      0.000       0.139       0.239
ar.L4          0.0618      0.031      1.993      0.046       0.001       0.123
ar.L5         -0.0576      0.034     -1.719      0.086      -0.123       0.008
sigma2      1.757e-07   5.62e-09     31.290      0.000    1.65e-07    1.87e-07
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):             24129.65
Prob(Q):                              0.88   Prob(JB):                         0.00
Heteroskedasticity (H):               2.95   Skew:                             4.09
...
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 4.56e+16. Standard errors may be unstable.
Mean Absolute Percent Error: 0.00021146
Root Mean Squared Error: 0.00024267
```

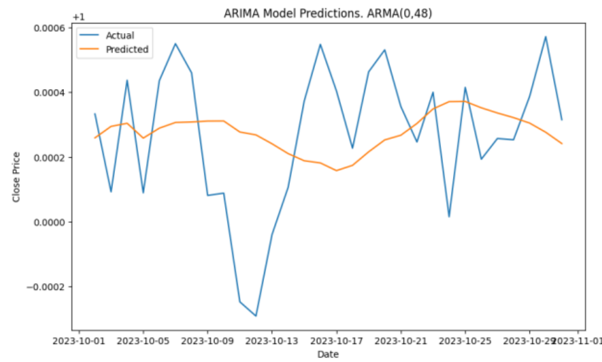




روی MA(48):q

```
  warnings.warn("Maximum Likelihood optimization failed to "
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  335
Model:                 ARIMA(0, 0, 48)   Log Likelihood                2066.676
Date:                Wed, 13 Dec 2023   AIC                          -4033.351
Time:                        13:19:19   BIC                          -3842.645
Sample:                    10-31-2022   HQIC                         -3957.322
                         - 09-30-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.0002      0.001    907.260      0.000       0.998       1.002
ma.L1          0.7044      0.036     19.816      0.000       0.635       0.774
ma.L2          0.7582      0.036     21.272      0.000       0.688       0.828
ma.L3          0.7954      0.047     17.079      0.000       0.704       0.887
ma.L4          0.8151      0.062     13.241      0.000       0.694       0.936
ma.L5          0.8120      0.047     17.404      0.000       0.721       0.903
ma.L6          0.8024      0.044     18.330      0.000       0.717       0.888
ma.L7          0.7856      0.044     17.887      0.000       0.700       0.872
ma.L8          0.7537      0.049     15.247      0.000       0.657       0.851
ma.L9          0.7162      0.053     13.427      0.000       0.612       0.821
ma.L10         0.6819      0.036     18.761      0.000       0.611       0.753
ma.L11         0.6512      0.054     12.159      0.000       0.546       0.756
ma.L12         0.6253      0.053     11.818      0.000       0.522       0.729
...
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 2.58e+18. Standard errors may be unstable.
Mean Absolute Percent Error: 0.00019248
Root Mean Squared Error: 0.00023289
```





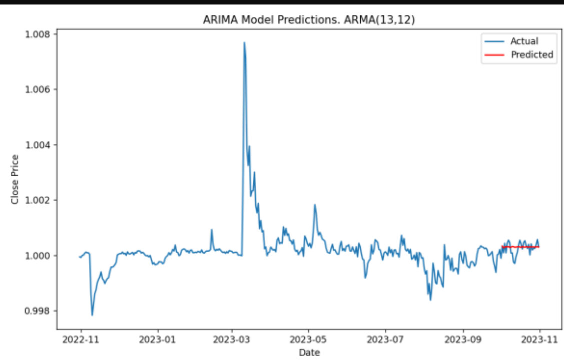
روی ARMA(13,12):p,q

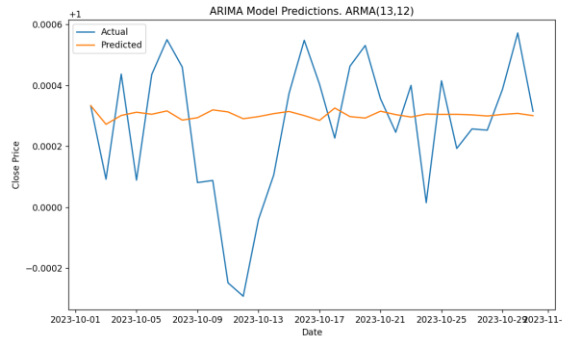
```python
best_comb = get_best_combination(data, train_end,test_end,loop_over_p=True,loop_over_q=True ,max_loop=20)
print(best_comb)
predict_and_plot(data, train_end,test_end,p= best_comb['p'],q = best_comb['q'],plot_option=True)
```

[35] ✓ 8m 40.6s Python

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                  335
Model:                ARIMA(13, 0, 12)   Log Likelihood                2101.044
Date:                Wed, 13 Dec 2023   AIC                          -4148.089
Time:                        13:15:37   BIC                          -4045.107
Sample:                    10-31-2022   HQIC                         -4107.033
                         - 09-30-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.8984      0.000   8102.105      0.000       0.898       0.899
ar.L1         -3.7693      0.001  -6335.875      0.000      -3.771      -3.768
ar.L2         -7.7800      0.002  -4399.125      0.000      -7.783      -7.777
```

```
ma.L10         7.1652      0.001    1.15e+04      0.000       7.164       7.166
ma.L11         2.1998      0.000    5011.618      0.000       2.199       2.201
ma.L12         0.3600   3.24e-05    1.11e+04      0.000       0.360       0.360
sigma2      1.974e-07    4.9e-09      40.301      0.000    1.88e-07    2.07e-07
===================================================================================
Ljung-Box (L1) (Q):                  11.47   Jarque-Bera (JB):              5721.60
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               3.65   Skew:                             1.77
Prob(H) (two-sided):                  0.00   Kurtosis:                        22.93
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Mean Absolute Percent Error: 0.00017665
Root Mean Squared Error: 0.00022274
```





در سل آخر، p,q در رنج ۰ تا ۲۰ تغیر میکنند نه ۰تا۵۰ چراکه ران تایم خیلی خیلی زیادی میگرفت(حجم نوت بوک هم زیاد میشد).
همچنین سل آخر نوت بوک اجرا نشده و در عوض last_part.py اجرا شده است.