

STOCK PRICE PREDICTION

Lý Nguyễn Thiên Bảo

MỤC LỤC

1. Giới thiệu về bộ dữ liệu
2. Quy trình phân tích
3. Quy trình xử lý
4. Data Engineering
5. Tổng kết

Vinamilk (Mã: VNM) được xem là một trong những mã cổ phiếu lớn nhất và có giá trị nhất trên thị trường chứng khoán Việt Nam, và là một trong những cổ phiếu thuộc VN30 - danh sách 30 cổ phiếu có vốn hóa thị trường lớn nhất.

Trong vài năm trở lại đây, mã cổ phiếu này đang có xu hướng mang lại nhiều lợi nhuận. Tuy nhiên, các nhà đầu tư vẫn cần một mô hình dự báo giá trị của mã cổ phiếu để có thể an tâm trong bối cảnh suy thoái kinh tế như hiện nay. Ta có bộ dữ liệu bao gồm các thông tin sau:

- **Date** (giao dịch mỗi ngày trong 6 năm: từ 2013 đến 2019)
- **OPEN/CLOSE** (Giá cổ phiếu đầu phiên và cuối phiên giao dịch)
- **VOLUME** (số lượng cổ phiếu được trade trong 1 phiên giao dịch)
- **HIGH/LOW** (giá cao nhất và thấp nhất của mã cổ phiếu trong phiên giao dịch...)

Tên bộ dữ liệu: Cổ phiếu Vinamilk 2013 - 2018

Định dạng dữ liệu: .csv

Nguồn: BaoViet Sercuritirs (open access)

HƯỚNG PHÂN TÍCH

Identify Situation



- Đánh giá vấn đề
- Xác định mục tiêu phân tích

Gather Data



- Tìm kiếm các dữ liệu cần có
- Thu thập dữ liệu

Clean Data



- Sử dụng các công cụ để làm sạch dữ liệu
- Trích xuất và tải dữ liệu về để sử dụng

Analyze Data



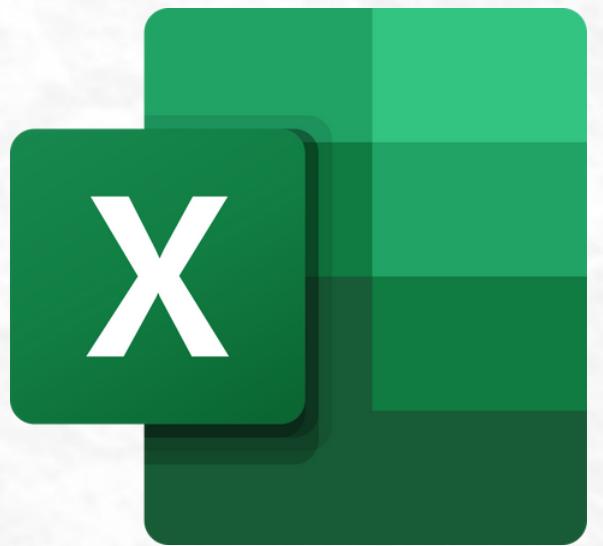
- Nhập dữ liệu vào các công cụ phân tích
- Tiến hành phân tích dữ liệu

Visualize Data



- Trực quan hóa dữ liệu
- Trình bày ý tưởng

CÁC CÔNG CỤ SỬ DỤNG



Xử lý và làm sạch dữ liệu



Tiến hành phân tích và học
máy để đưa ra mô hình

XỬ LÝ VÀ LÀM SẠCH DỮ LIỆU

EXCEL

TICKER	DATE	OPEN	HIGH	LOW	CLOSE	VOLUME
VNM	1/2/2013	87.5	88	87	87	199930
VNM	1/3/2013	87	87.5	86	86.5	146370
VNM	1/4/2013	86	88	86	88	147250
VNM	1/7/2013	88	89	88	88	135530
VNM	1/8/2013	88	92	88	92	629670
VNM	1/9/2013	96.5	96.5	93.5	95.5	455050
VNM	1/10/2013	95.5	100	95	100	593410
VNM	1/11/2013	102	104	101	103	362160
VNM	1/14/2013	101	102	98.5	99.5	371220
VNM	1/15/2013	99	101	99	100	192400
VNM	1/16/2013	100	101	99	99.5	215810
VNM	1/17/2013	99.5	99.5	98	98	291160
VNM	1/18/2013	98	98.5	96.5	97.5	206860
VNM	1/21/2013	98	98.5	96.5	96.5	75950
VNM	1/22/2013	96	97.5	95.5	96.5	246990

PYTHON

1. Khai báo các thư viện cần sử dụng

```
# Basic
import pandas as pd
import numpy as np
import datetime
%matplotlib inline
import math

# Visualize
import matplotlib.pyplot as plt
import seaborn as sns
from plotly.graph_objs import Candlestick, Figure
from plotly.offline import iplot\

# Statistic and machine learning
import copy
from scipy import stats
from statsmodels.tsa.api import adfuller, kpss, seasonal_decompose
from sklearn.metrics import mean_absolute_percentage_error, mean_absolute_error, mean_squared_error
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from itertools import product
import warnings
warnings.filterwarnings('ignore')
plt.rcParams["figure.figsize"] = [7, 7]
```

2. Tiến hành import và đọc dữ liệu

```
▶ from google.colab import drive
drive.mount('/content/drive')

[ ] Drive already mounted at /content/drive; to attempt to forcibly re
```



```
[ ] from google.colab import files
import pandas as pd
```



```
[ ] df=pd.read_csv('/content/drive/MyDrive/Data/VNM.csv', sep=';')
df['DATE'] = pd.to_datetime(df['DATE'])
df.head()
```

	TICKER	DATE	OPEN	HIGH	LOW	CLOSE	VOLUME
0	VNM	2013-01-02	87.5	88.0	87.0	87.0	199930
1	VNM	2013-01-03	87.0	87.5	86.0	86.5	146370
2	VNM	2013-01-04	86.0	88.0	86.0	88.0	147250
3	VNM	2013-01-07	88.0	89.0	88.0	88.0	135530
4	VNM	2013-01-08	88.0	92.0	88.0	92.0	629670

XỬ LÝ VÀ LÀM SẠCH DỮ LIỆU

PYTHON

3. Data preparation and pre-processing

Data preparation and pre-processing

```

start = datetime.datetime(2013, 7, 1)
end = datetime.datetime(2019, 10, 14)

[ ] # Get the number of rows and columns
print(df.shape)
# get the column names
print(df.columns)

(1143, 7)
Index(['TICKER', 'DATE', 'OPEN', 'HIGH', 'LOW', 'CLOSE', 'VOLUME'], dtype='object')

```

▶ # Get general information about the data
`df.info()`
`print("-" * 50)`
`print(pd.value_counts(df.dtypes))`

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1143 entries, 0 to 1142
Data columns (total 7 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   TICKER    1143 non-null   object  
 1   DATE      1143 non-null   datetime64[ns]
 2   OPEN      1143 non-null   float64 
 3   HIGH      1143 non-null   float64 
 4   LOW       1143 non-null   float64 
 5   CLOSE     1143 non-null   float64 
 6   VOLUME    1143 non-null   int64   
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 62.6+ KB

```

[] `df.describe()`

	OPEN	HIGH	LOW	CLOSE	VOLUME
count	1143.000000	1143.000000	1143.000000	1143.000000	1.143000e+03
mean	128.950394	129.995363	128.026947	129.036745	1.080757e+05
std	19.890977	19.994699	19.837876	19.969250	1.994541e+05
min	86.000000	87.500000	86.000000	86.500000	0.000000e+00
25%	109.500000	111.000000	108.000000	110.000000	2.107650e+04

4. Chuyển giá trị theo ngày thành giá trị theo tháng

```

#calculate returns an empty data frame
returns = pd.DataFrame()
# df['Return'] = df['CLOSE'].pct_change()
df['Return'] = df['CLOSE']
df = df.dropna()
df.head()

```

	TICKER	DATE	OPEN	HIGH	LOW	CLOSE	VOLUME	Return
0	VNM	2013-01-02	87.5	88.0	87.0	87.0	199930	87.0
1	VNM	2013-01-03	87.0	87.5	86.0	86.5	146370	86.5
2	VNM	2013-01-04	86.0	88.0	86.0	88.0	147250	88.0
3	VNM	2013-01-07	88.0	89.0	88.0	88.0	135530	88.0
4	VNM	2013-01-08	88.0	92.0	88.0	92.0	629670	92.0

▶ # Calculate monthly value
`df = df.groupby(df['DATE'].dt.to_period('M'))['Return'].avg()`
`print(df)`

▶ `df.index = df.index.to_timestamp()`

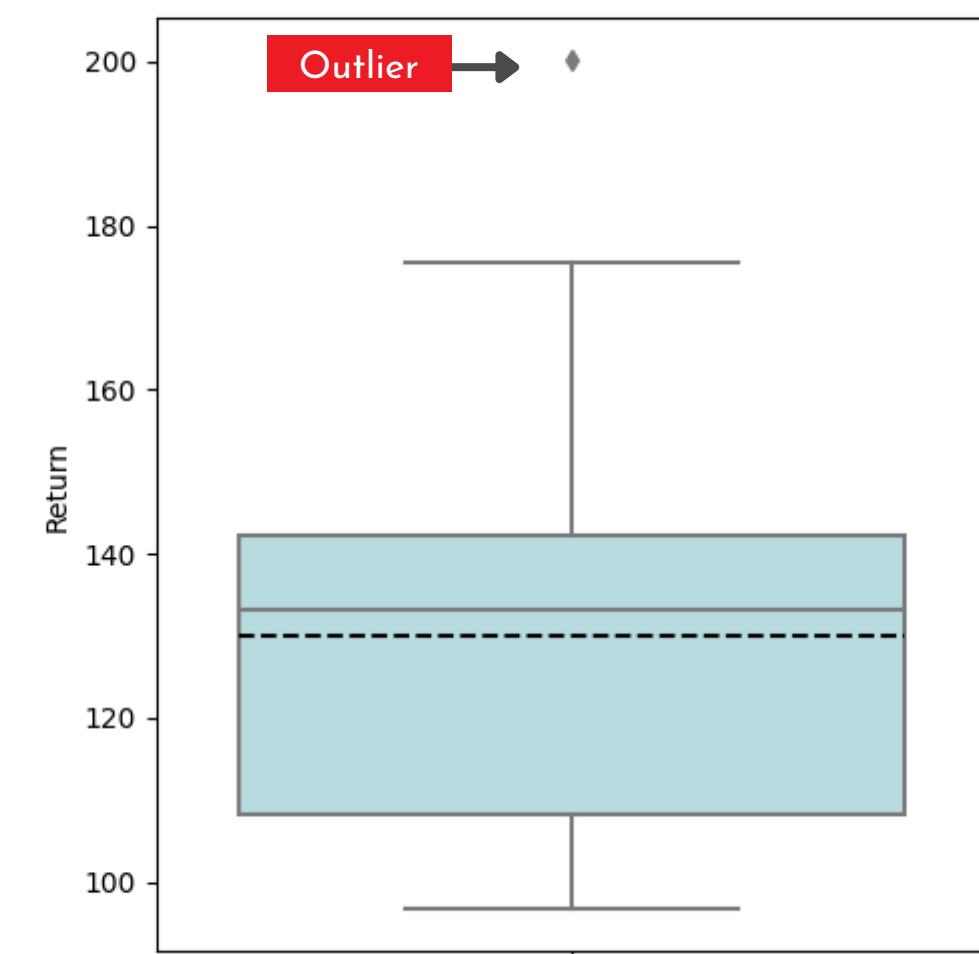
▶ `df.head()`

DATE	Return
2013-01-01	2137.0
2013-02-01	1528.5
2013-03-01	2153.0
2013-04-01	2361.0
2013-05-01	2797.0

Freq: MS, Name: Return, dtype: float64

XỬ LÝ GIÁ TRỊ OUTLIERS

```
# Set the color for the boxes
box_color = '#B0E0E6'
# Create the subplots
fig, axes = plt.subplots(1, 1, figsize=(5, 5))
# Box plot
sns.boxplot(y=df, color=box_color, showmeans=True, meanline=True
             , meanprops=dict(color="black", linewidth=1.5))
# Adjust the layout
plt.tight_layout()
# Show the graph
plt.show()
```



Bước 1: Xác định các giá trị ngoại lai bằng biểu đồ Box plot

```
# Detect outliers
import numpy as np

def detect_outliers(series):
    """
        series: 1-D numpy array input
    """
    Q1 = np.quantile(series, 0.25)
    Q3 = np.quantile(series, 0.75)
    IQR = Q3-Q1
    lower_bound = Q1-1.5*IQR
    upper_bound = Q3+1.5*IQR
    lower_compare = series <= lower_bound
    upper_compare = series >= upper_bound
    outlier_idxs = np.where(lower_compare | upper_compare)[0]
    return outlier_idxs

outlier_idxs=detect_outliers(df)
print("Outlier indices: ", outlier_idxs)
print("Outlier values: ", df[outlier_idxs])
```

```
Outlier indices: [59]
Outlier values: DATE
2017-12-01 200.164286
Freq: MS, Name: Return, dtype: float64
```

```
# Remove outliers
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
df = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR)))]
```

2. Bước 2: Tìm ra các dòng có giá trị ngoại lai

Vì đây là chuỗi có trend nên việc có các giá trị rất thấp ở phía đuôi hoặc rất cao ở phía đầu dễ bị nhầm lẫn thành outlier.

Do đó, cần cân nhắc cẩn thận trước khi loại bỏ giá trị Outliers.

Note: đã so sánh và thấy loại bỏ Outliers cho kết quả tốt hơn

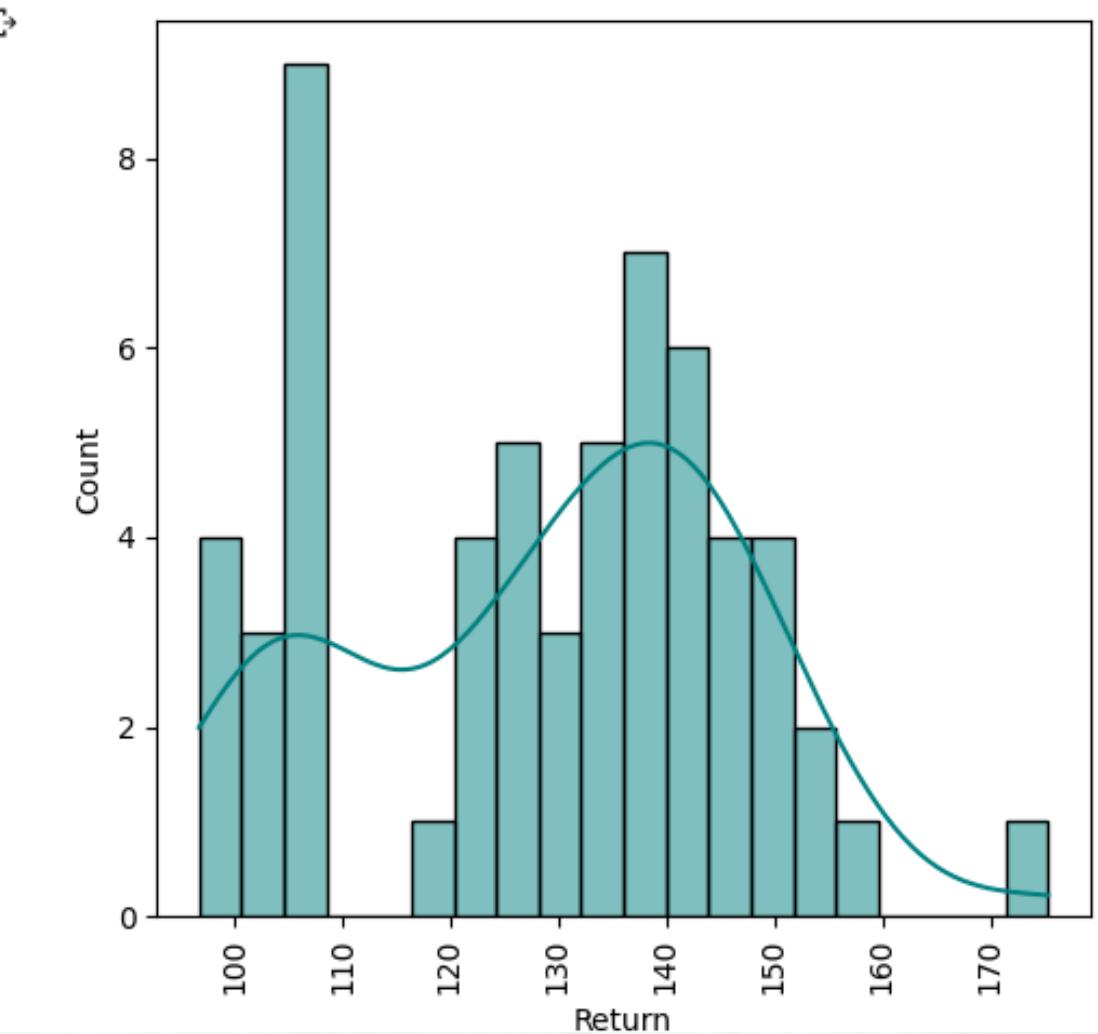
3. Bước 3: Dùng code để loại bỏ các giá trị ngoại lai

VISUALIZE ĐÁNH GIÁ DỮ LIỆU

Histogram plot

```
▶ sns.histplot(df, color='teal', edgecolor='black', kde=True, bins=20)
  plt.xticks(rotation=90)

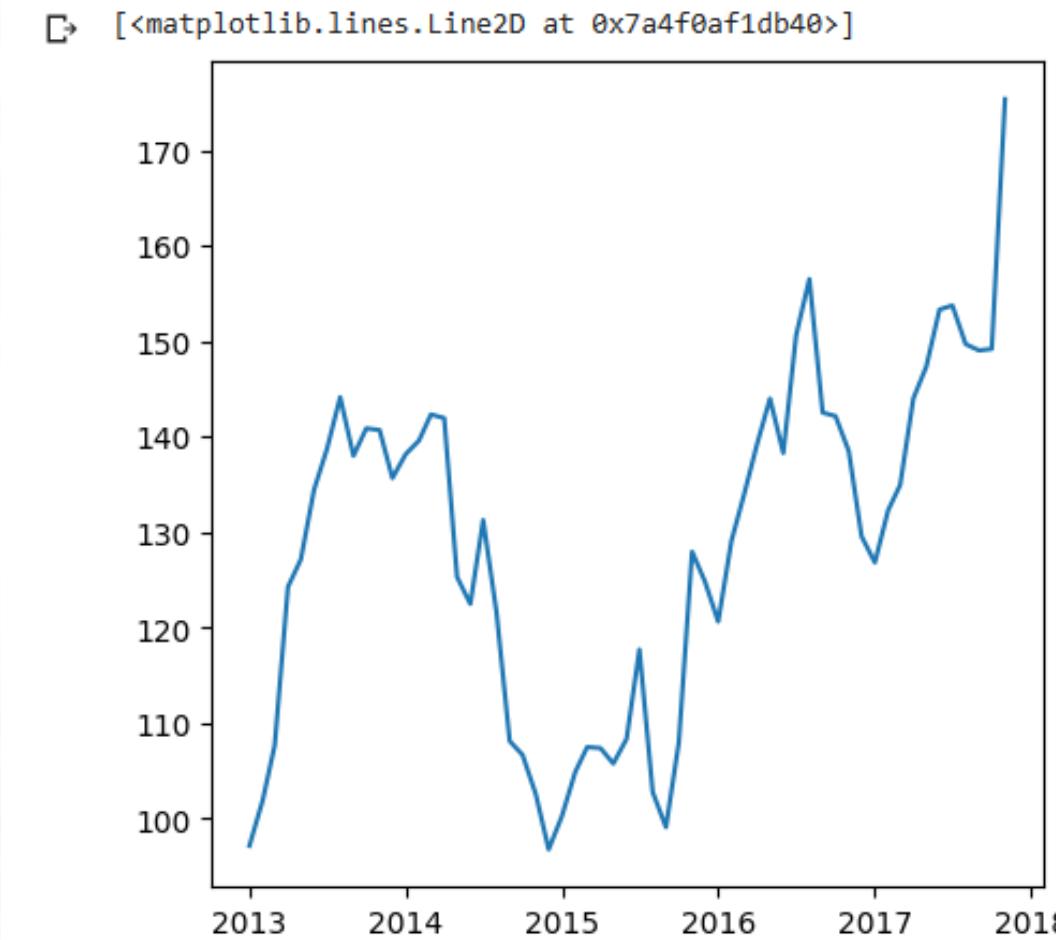
  plt.tight_layout()
  plt.show()
```



Các giá trị phân phối xấp xỉ chuẩn và có xu hướng lệch về phía bên trái

Average monthly price of Vinamilk stock

```
▶ import statsmodels
  import statsmodels.api as sm
  from statsmodels.tsa.stattools import coint, adfuller
  plt.plot(df)
```



Giá cả của cổ phiếu cho xu hướng biến động theo trend, chu kỳ tăng và giảm, nhưng nhìn chung vẫn có xu hướng tăng trưởng

DATA ENGINEERING

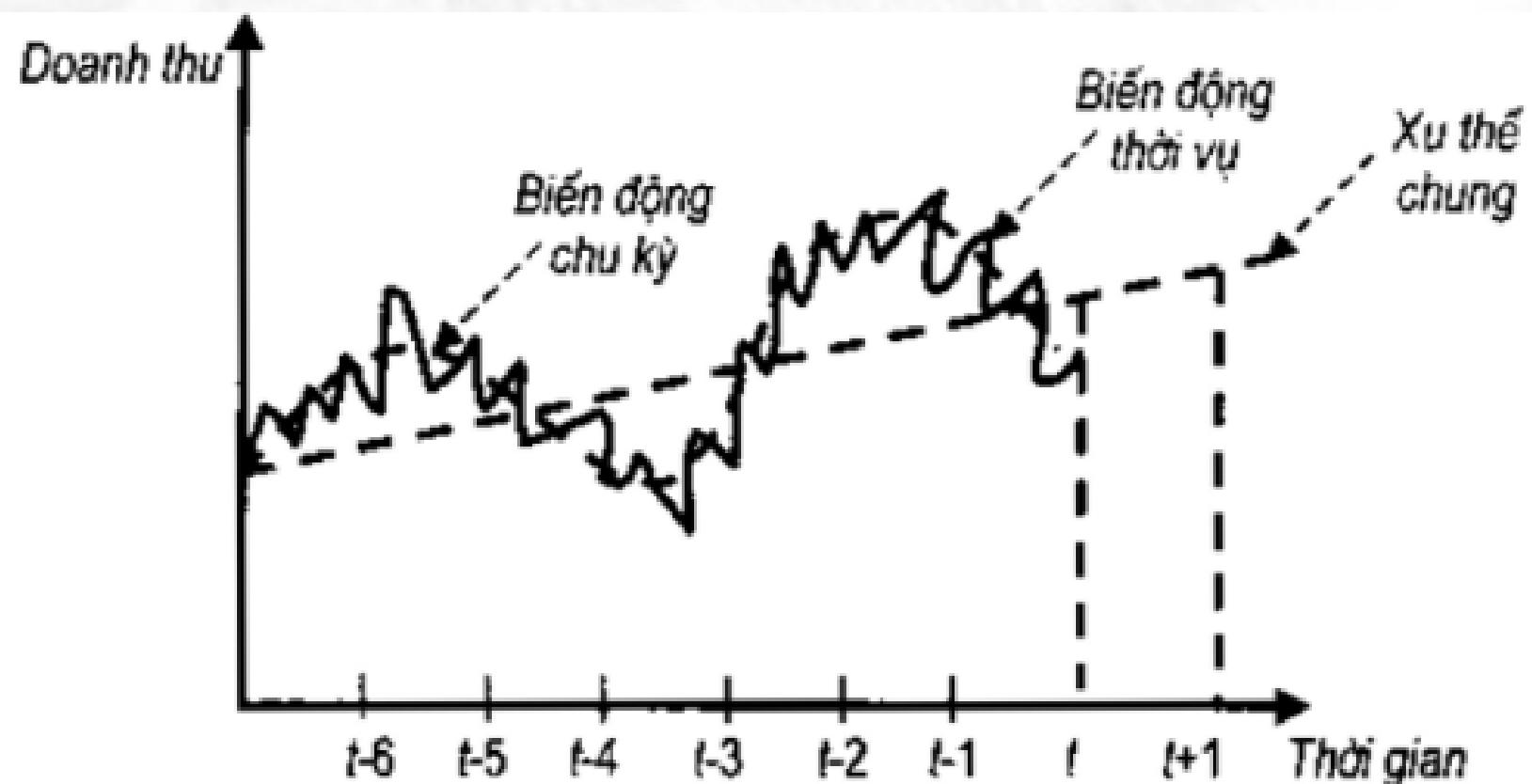
Vì phân tích dữ liệu theo thời gian phụ thuộc nhiều yếu tố. Vậy nên, cần xác định những yếu tố đó trước khi tiến hành đưa ra mô hình dự báo.

Additive model:

$$Y = \text{level} + \text{trend} + \text{seasonality} + \text{noise}$$

Multiplicative model:

$$Y = \text{level} * \text{trend} * \text{seasonality} * \text{noise}$$

**Các bước xử lý lag, noise**

- Check stationarity:

[] 4 3 cells hidden

- Elimination of Trend and Seasonal Components

[] 4 6 cells hidden

- Differencing

[] 4 3 cells hidden

- Fitting model

[] 4 2 cells hidden

- Grid search

[] 4 9 cells hidden

- Tests for model

▶ 4 5 cells hidden

CHECK STATIONARY

Check bằng kiểm định

Check stationarity:

Check stationarity:

- Kiểm định Augmented Dickey-Fuller: Với giả thuyết H_0 : The series has a unit root.

Với p-value > 0.05, ta chấp nhận thuyết H_0 hay tập chuỗi không có định.

- Kiểm định KPSS (Kwiatkowski-Phillips-Schmidt-Shin): Với giả thuyết H_0 : the series is stationary.

If p_value < 0.05, ta bác bỏ giả thuyết H_0 hay chuỗi không có tính dừng.

Nếu một chuỗi thời gian là stationary, thì nó sẽ không thay đổi tính chất thống kê của mình theo thời gian. Cụ thể, nó sẽ có các tính chất sau:

- Giá trị trung bình của chuỗi không thay đổi theo thời gian.
- Phương sai của chuỗi không thay đổi theo thời gian.
- Hàm tự tương quan giữa các giá trị trong chuỗi chỉ phụ thuộc vào khoảng cách giữa chúng và không phụ thuộc vào thời gian.

```
[115]: def check_stationarity(df):
    adf_results = adfuller(df)[1:3]
    kpss_results = kpss(df)[1:3]
    adf_s = 'Non-Stationary'
    kpss_s = 'Non-Stationary'
    if adf_results[0] < 0.05:
        adf_s = 'Stationary'
    if kpss_results[0] > 0.05:
        kpss_s = 'Stationary'

    tests = ['ADF', 'KPSS']
    pvalues = [adf_results[0], kpss_results[0]]
    is_stationary = [adf_s, kpss_s]
    stationary_df = pd.DataFrame({'tests': tests, 'p-value': pvalues, 'is_stationary': is_stationary})

    return stationary_df

check_stationarity(df)
```

tests	p-value	is_stationary
0 ADF	0.695542	Non-Stationary
1 KPSS	0.067438	Stationary

Kiểm định ADF: Non stationary
 Kiểm định KPSS: Stationary

Check bằng Stationary plot

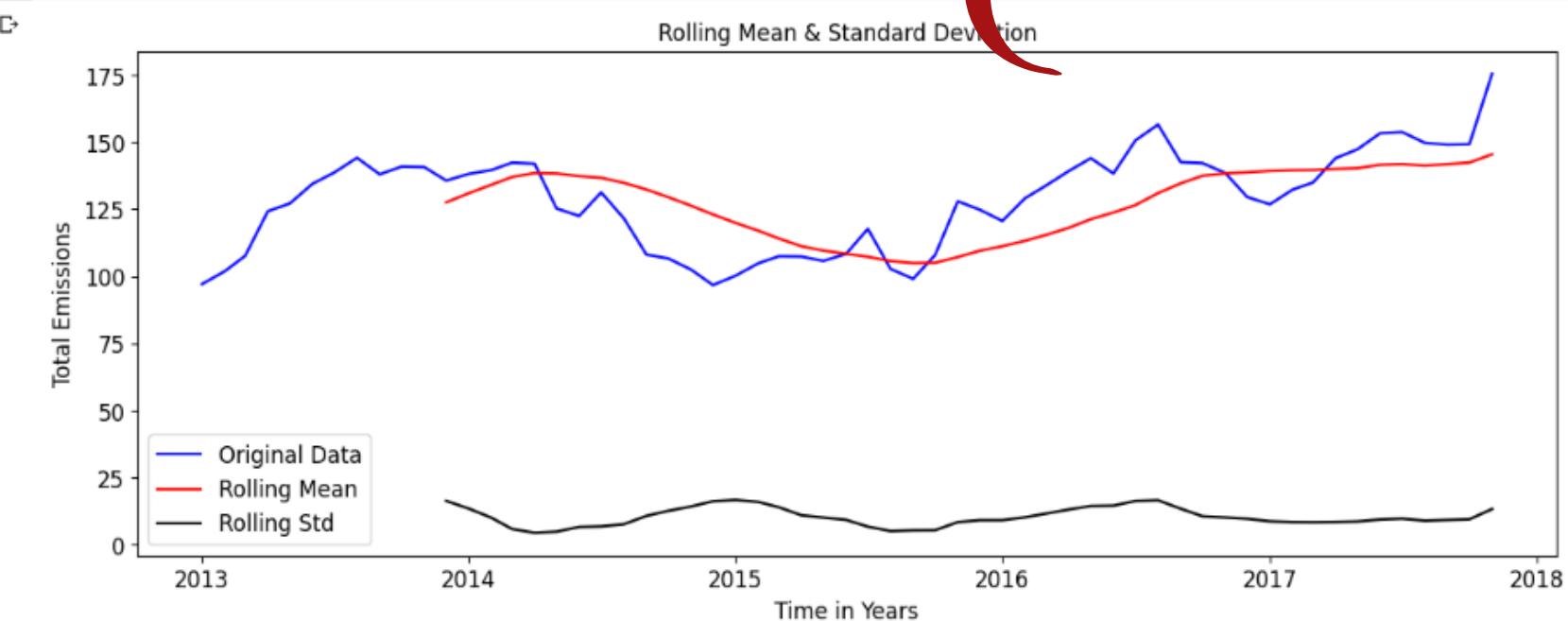
Dữ liệu sẽ Stationary nếu:

- Đồ thị thể hiện xu hướng không đổi theo thời gian (không có xu hướng tăng hoặc giảm).
- Đồ thị có phương sai (variance) không đổi theo thời gian.
- Đồ thị có mức độ tự tương quan (autocorrelation) không đổi theo thời gian.

```
def TeststationaryPlot(ts):
    rol_mean = ts.rolling(window = 12, center = False).mean()
    rol_std = ts.rolling(window = 12, center = False).std()

    plt.figure(figsize=(14, 5))
    plt.plot(ts, color = 'blue',label = 'Original Data')
    plt.plot(rol_mean, color = 'red', label = 'Rolling Mean')
    plt.plot(rol_std, color = 'black', label = 'Rolling Std')
    plt.xticks(fontsize = 12)
    plt.yticks(fontsize = 12)

    plt.xlabel('Time in Years', fontsize = 12)
    plt.ylabel('Total Emissions', fontsize = 12)
    plt.legend(loc='best', fontsize = 12)
    plt.title('Rolling Mean & Standard Deviation', fontsize = 12)
    plt.show(block= True)
TestStationaryPlot(df)
```



Không thỏa mãn
 → Non_Stationary

ELIMINATE TREND & SEASON

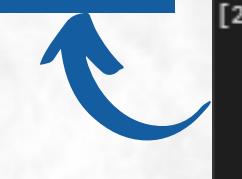
Tạo hàm trend_estimation được sử dụng để ước lượng xu hướng của chuỗi thời gian.



Ước lượng và vẽ đồ thị của thành phần xu hướng (trend component) trong chuỗi thời gian sử dụng hàm trend_estimation



Loại bỏ thành phần trend để dễ dàng phân tích các thành phần khác



Chuẩn bị phân tích và visualize các thành phần khác (seasonal, residual)



```
[275] def trend_estimation(series, period):
    trend_list = []
    q = int(period/2)
    history = [series[i] for i in range(period)]
    test = [series[i] for i in range(period, len(series))]

    for t in range(len(test)):
        length = len(history)
        yhat = np.mean([history[i] for i in range(length-period,length)])
        obs = test[t]
        trend_list.append(yhat)
        history.append(obs)

    return trend_list, test
```

```
▶ series = df.values
period = 12
q = int(period/2)
index=df.index[q:-q]
trend, test = trend_estimation(series, period)
trend = pd.Series(data = trend, index=index)
test = pd.Series(data = test, index=index)

# Plot trend estimation
plt.plot(test)
plt.plot(trend, label='trend component')
plt.title('The trend component of return')
plt.legend()
plt.show()
```

```
[277] # Detrending data
detrended = df - trend
plt.plot(detrended, marker='s', mfc='none')
plt.title('The detrended return')
plt.show()
```

```
[278] from statsmodels.tsa.seasonal import seasonal_decompose
df = df.to_frame()
df = df.reset_index()
result = seasonal_decompose(df["Return"], model='additive', period=12)

[279] from statsmodels.tsa.seasonal import seasonal_decompose
df= df.reset_index()
result = seasonal_decompose(df["Return"], model='additive', period=12)
```

```
▶ # Plot
f, (ax1, ax2) = plt.subplots(2, 2)
fig.set_size_inches(10, 12)

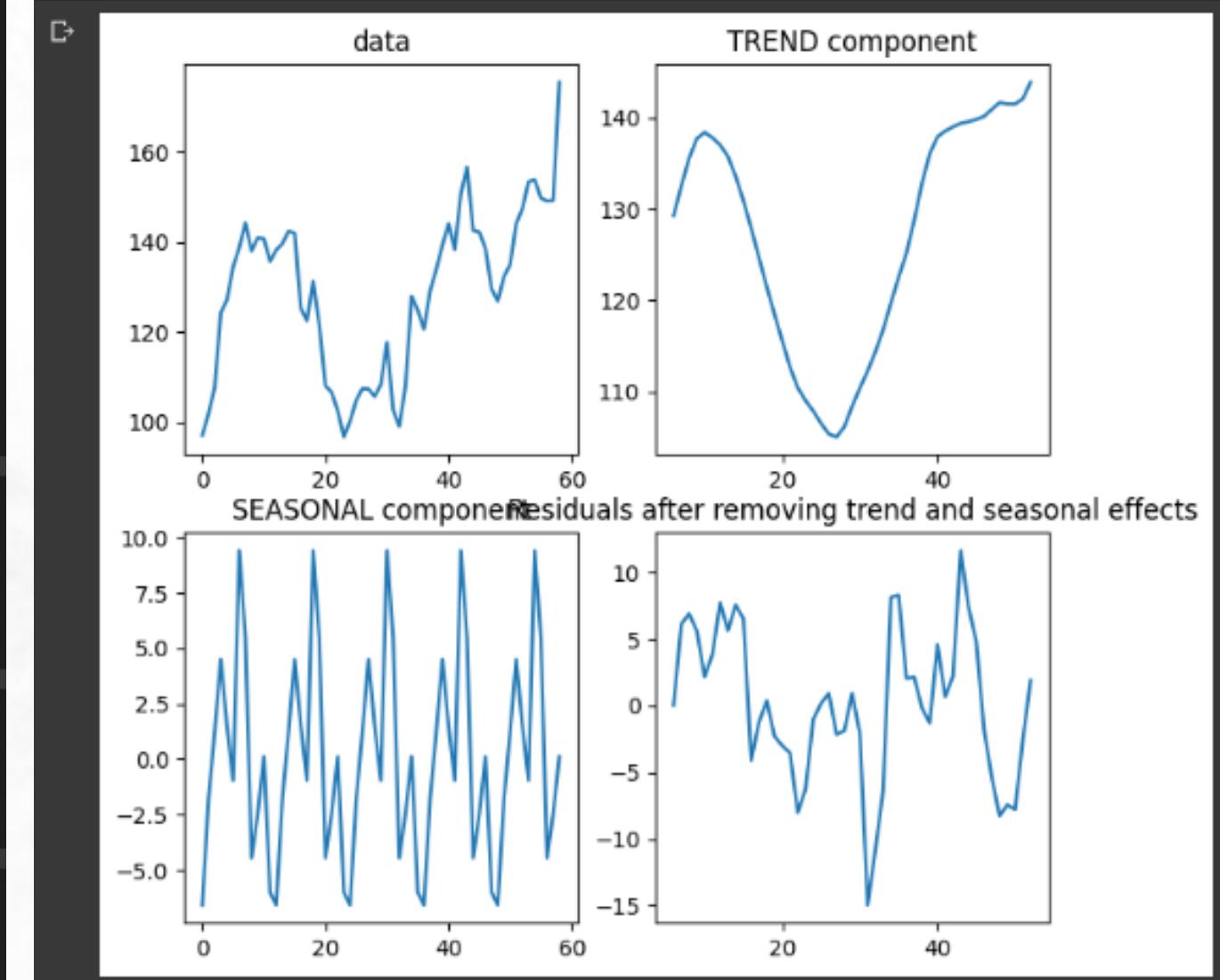
ax1[0].plot(df["Return"])
ax1[0].set_title('data')

ax1[1].plot(result.trend)
ax1[1].set_title('TREND component')

ax2[0].plot(result.seasonal)
ax2[0].set_title('SEASONAL component')

ax2[1].plot(result.resid)
ax2[1].set_title('Residuals after removing trend and seasonal effects')
plt.show()
```

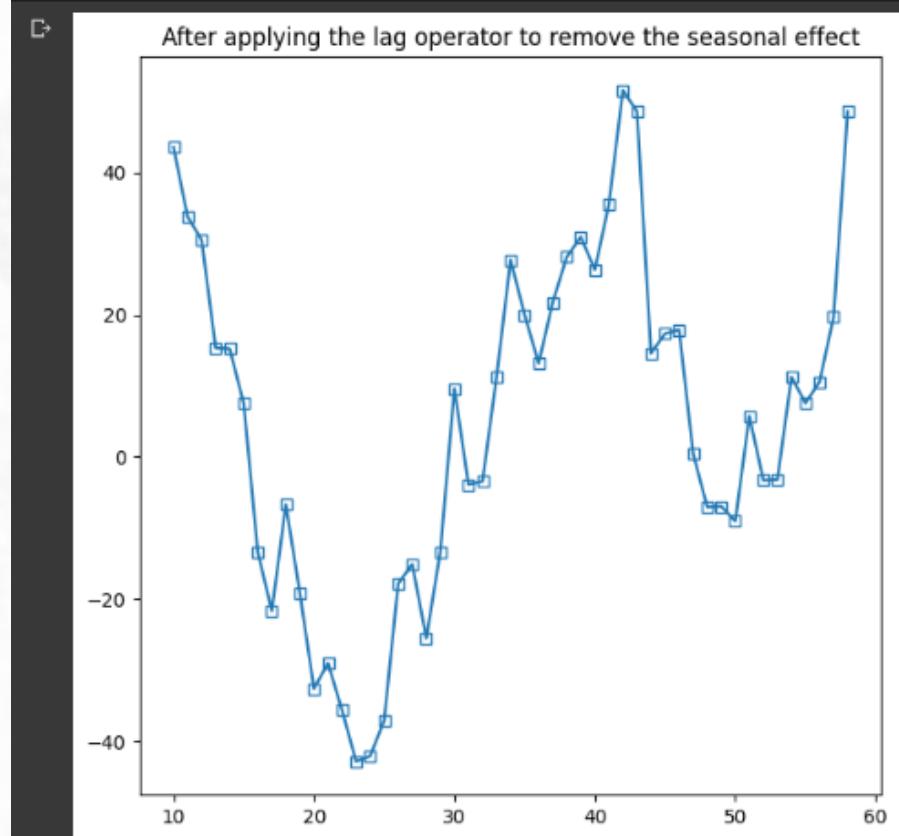
Trích lọc các thành phần cấu thành nên chuỗi bao gồm: xu hướng (trend), mùa vụ (seasonal), phần dư (residual)



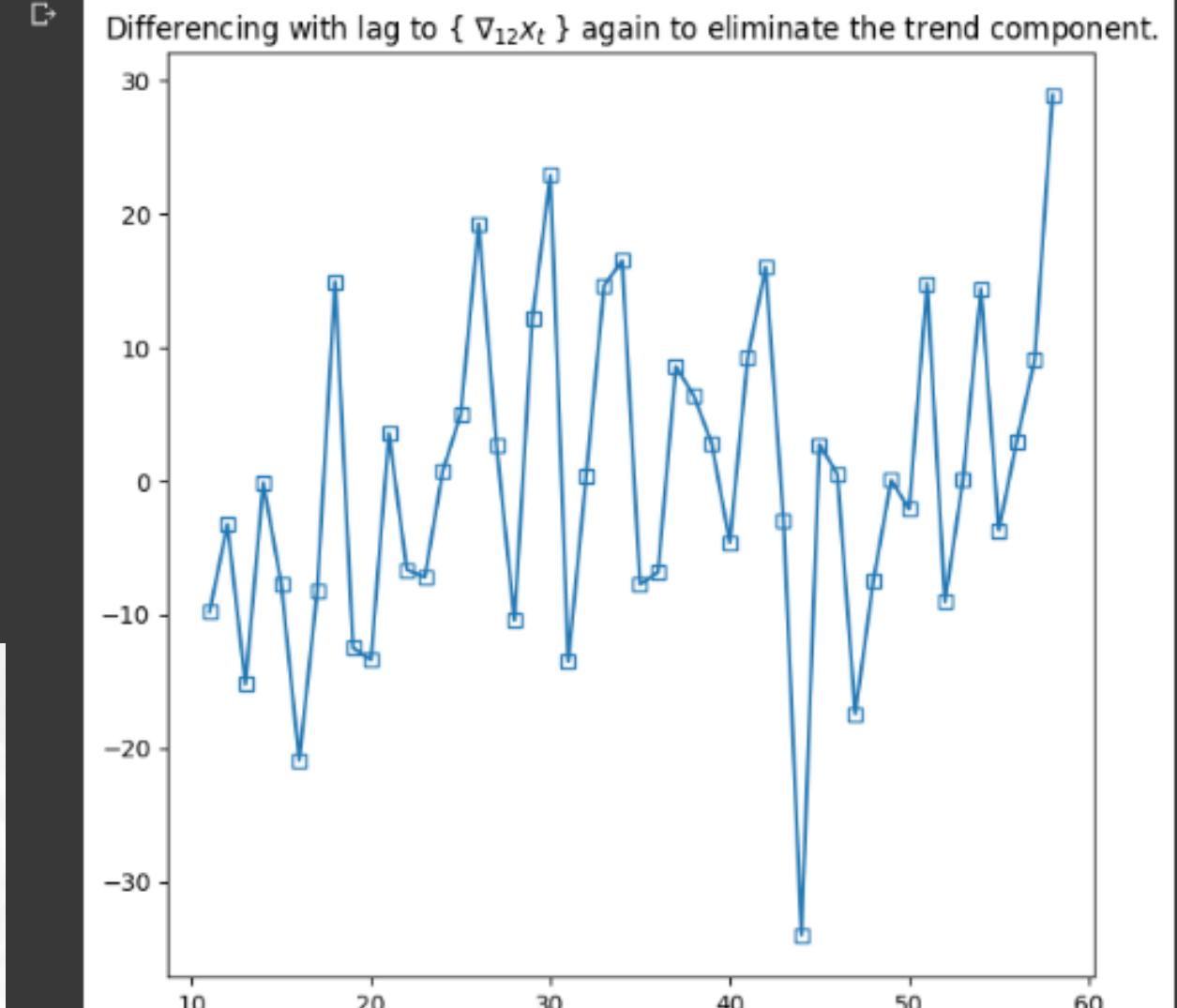
DIFFERENCING

```
[281] # create a differenced series
def difference(dataset, lag=1):
    diff = list()
    for i in range(lag, len(dataset)):
        value = dataset[i] - dataset[i - lag]
        diff.append(value)
    return diff
```

```
X = df["Return"]
lag=10
index=df.index[lag:]
diff_12 = pd.Series(difference(X, lag), index=index)
plt.plot(diff_12, marker='s', mfc='none')
plt.title("After applying the lag operator to remove the seasonal effect")
plt.show()
```



```
index_2 = diff_12.index[1:]
diff_12_1 = pd.Series(difference(diff_12.values, 1), index=index_2)
plt.plot(diff_12_1, marker='s', mfc='none')
plt.title("Differencing with lag to { $v_{12}x_t$ } again to eliminate the trend component.")
plt.show()
```



**Loại bỏ thành phần mùa vụ
(seasonal)**

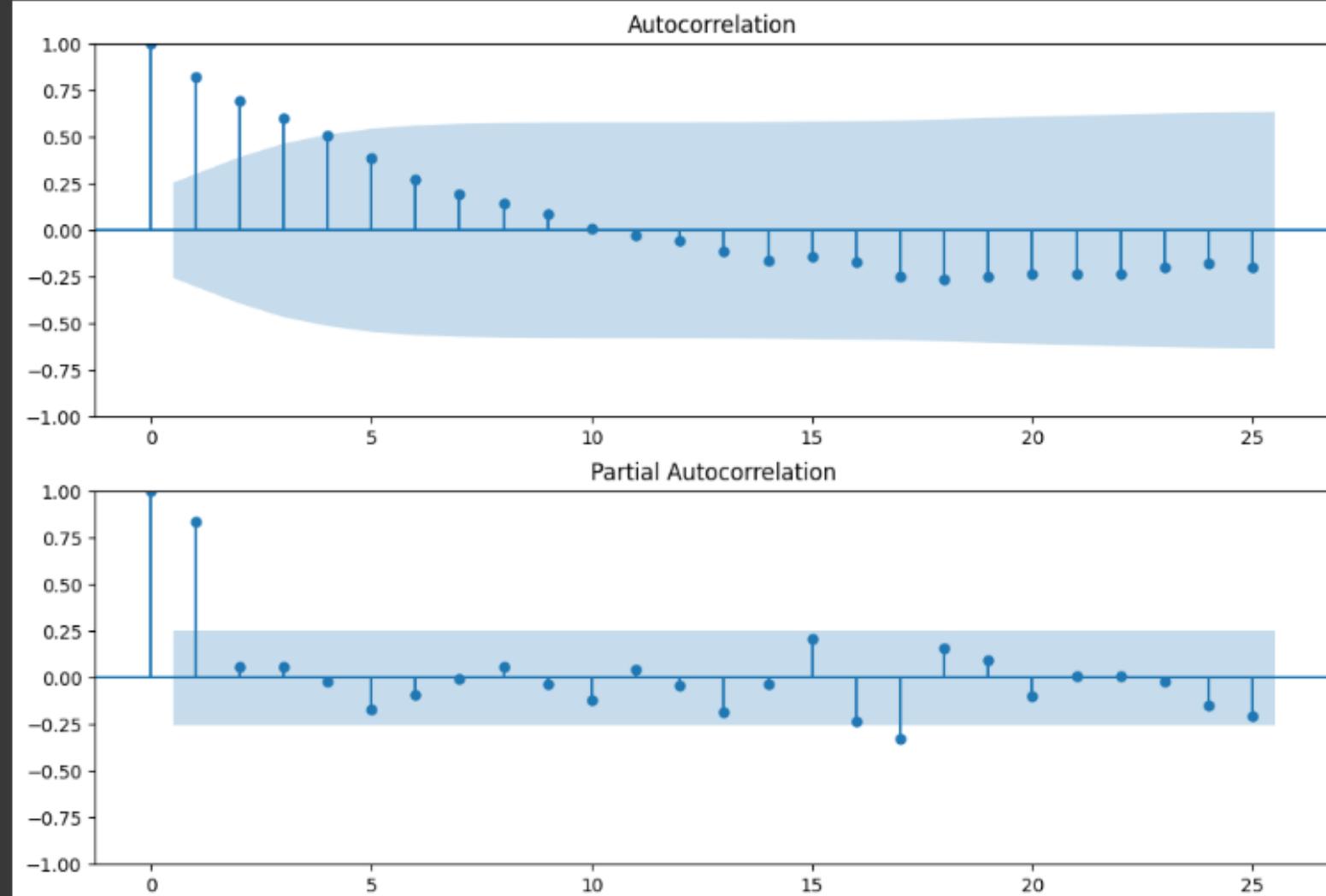
**Loại bỏ thành phần xu
hướng (trend)**

Sử dụng tham số lag=10

FITTING MODEL

Fitting model

```
from statsmodels.graphics.tsaplots import plot_acf, acf
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(df["Return"], lags=25, ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(df["Return"], lags=25, ax=ax2)
```



```
[285] rho = acf(df["Return"], nlags = 40, fft = False)
print(rho[12], rho[24], rho[36])
```

-0.05178456796310908 -0.17327643558914907 -0.04901626273175999

Time Series Analysis

ARIMA(p,d,q)

AutoRegressive Integrated Moving Average

Auto-Regressive (AR): p

xác định số lượng giá trị quá khứ của chuỗi thời gian, cho biết mức độ tương quan giữa các giá trị quá khứ và giá trị hiện tại

Moving average (MA): q

số lượng sai số quá khứ, cho biết mức độ tương quan giữa các sai số quá khứ và giá trị hiện tại, q cao => tương quan xa

Integrated (I): d

số lần tích phân được áp dụng cho chuỗi thời gian để làm cho nó dừng (stationary).

d>0: loại bỏ xu hướng và biến động không ổn định.

K12	A	B	C	D	E	F	Lag-time			MA_2	MA_3	MA_4	...
							t-1	t-2	t-3				
1													
2	Time	t											
3	1/1/1999	10.4	45.3	36.6	33.7					27.9	30.8	31.5	
4	1/2/1999	45.3	36.6	33.7	31.8					41.0	38.5	36.85	
5	1/3/1999	36.6	33.7	31.8	30.8					35.2	34.0	33.225	
6	1/4/1999	33.7	31.8	30.8	29.4					32.8	32.1	31.425	
7	1/5/1999	31.8	30.8	29.4	28					31.3	30.7	30	
8	1/6/1999	30.8	29.4	28	28					30.1	29.4	29.05	
9	1/7/1999	29.4	28	28	27.3					28.7	28.5	28.175	
10	1/8/1999	28	28	27.3	42.9					28.0	27.8	31.55	
11	1/9/1999	28	27.3	42.9	71					27.7	32.7	42.3	
12	1/10/1999	27.3	42.9	71	45.3					35.1	47.1	46.625	
13	1/11/1999	42.9	71	45.3	29.4					57.0	53.1	47.15	
14	1/12/1999	71	45.3	29.4	28					58.2	48.6	43.425	
15	1/13/1999	45.3	29.4	28	26.6					37.4	34.2	32.325	
16	1/14/1999	29.4	28	26.6	24.5					28.7	28.0	27.125	
17	1/15/1999	28	26.6	24.5	22.9					27.3	26.4	25.5	
18	1/16/1999	26.6	24.5	22.9	22.4					25.6	24.7	24.1	
19	1/17/1999	24.5	22.9	22.4	22.9					23.7	23.3	23.175	
20	1/18/1999	22.9	22.4	22.9	29.4					22.7	22.7	24.4	
21	1/19/1999	22.4	22.9	29.4	41.6					22.7	24.9	29.075	
22	1/20/1999	22.9	29.4	41.6	37.5					26.2	31.3	32.85	
23	1/21/1999	29.4	41.6	37.5	30.8					35.5	36.2	34.825	
24	1/22/1999	41.6	37.5	30.8	25.2					39.6	36.6	33.775	
25	1/23/1999	37.5	30.8	25.2	22.9					34.2	31.2	29.1	
26	1/24/1999	30.8	25.2	22.9	22					28.0	26.3	25.225	
27	1/25/1999	25.2	22.9	22	21.1					24.1	23.4	22.8	
28	1/26/1999	22.9	22	21.1	20.2					22.5	22.0	21.55	
29	1/27/1999	22	21.1	20.2	19					21.6	21.1	20.575	
30	1/28/1999	21.1	20.2	19	18.7					20.7	20.1	19.75	
31	1/29/1999	20.2	19	18.7	18.7					19.6	19.3	19.15	
32	1/30/1999	19	18.7	18.7	18.1					18.9	18.8	18.625	
33	1/31/1999	18.7	18.7	18.1	18.1					18.7	18.5	18.4	
34	2/1/1999	18.7	18.1	18.1	18.7					18.4	18.3	18.4	
35	2/2/1999	18.1	18.1	18.7	18.7					18.1	18.3	18.4	

Tìm điểm tối ưu của p,q,d ???

GRID SEARCH

```

import itertools
p = d = q = range(0, 2) # Define the p, d and q parameters to take any value between 0 and 2
pdq = list(itertools.product(p, d, q)) # Generate all different combinations of p, q and q triplets
pdq_x_QDQs = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))] # Generate all different combinations of seasonal p, q and q triplets
print('Examples of Seasonal ARIMA parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], pdq_x_QDQs[1]))
print('SARIMAX: {} x {}'.format(pdq[2], pdq_x_QDQs[2]))
```

+ Code + Text

```

[288] for param in pdq:
    for seasonal_param in pdq_x_QDQs:
        try:
            mod = sm.tsa.statespace.SARIMAX(df,
                                              order=param,
                                              seasonal_order=seasonal_param,
                                              enforce_stationarity=False,
                                              enforce_invertibility=False)
            results = mod.fit()
            print('ARIMA{}x{} - AIC:{}'.format(param, seasonal_param, results.aic))
        except:
            continue
```

```

[289] a=[]
b=[]
c=[]
wf=pd.DataFrame()
```

```

import warnings
warnings.filterwarnings('ignore')
warnings.filterwarnings("ignore") # specify to ignore warning messages
for param in pdq:
    for param_seasonal in pdq_x_QDQs:
        try:
            mod = sm.tsa.statespace.SARIMAX(df["Return"],
                                              order=param,
                                              seasonal_order=param_seasonal,
                                              enforce_stationarity=False,
                                              enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
            a.append(param)
            b.append(param_seasonal)
            c.append(results.aic)
        except:
            continue
wf['pdq']=a
wf['pdq_x_QDQs']=b
wf['aic']=c
print(wf[wf['aic']==wf['aic'].min()])

```

```

[2] ARIMA(1, 0, 1)x(0, 0, 0, 12)12 - AIC:401.6428140547418
ARIMA(1, 0, 1)x(0, 0, 1, 12)12 - AIC:325.2966227342697
ARIMA(1, 0, 1)x(0, 1, 0, 12)12 - AIC:341.90824934017456
ARIMA(1, 0, 1)x(0, 1, 1, 12)12 - AIC:1265.2167574109585
ARIMA(1, 0, 1)x(1, 0, 0, 12)12 - AIC:330.24818755693457
ARIMA(1, 0, 1)x(1, 0, 1, 12)12 - AIC:325.06668619568256
ARIMA(1, 0, 1)x(1, 1, 0, 12)12 - AIC:246.8601579239894
ARIMA(1, 0, 1)x(1, 1, 1, 12)12 - AIC:1290.2899162638264
ARIMA(1, 1, 0)x(0, 0, 0, 12)12 - AIC:401.1141024573767
ARIMA(1, 1, 0)x(0, 0, 1, 12)12 - AIC:323.56550404325134
ARIMA(1, 1, 0)x(0, 1, 0, 12)12 - AIC:342.0451588848962
ARIMA(1, 1, 0)x(0, 1, 1, 12)12 - AIC:1407.1938790645027
ARIMA(1, 1, 0)x(1, 0, 0, 12)12 - AIC:323.0191805986701
ARIMA(1, 1, 0)x(1, 0, 1, 12)12 - AIC:323.4818832933914
ARIMA(1, 1, 0)x(1, 1, 0, 12)12 - AIC:238.8323614593431
ARIMA(1, 1, 0)x(1, 1, 1, 12)12 - AIC:1395.391053291775
ARIMA(1, 1, 1)x(0, 0, 0, 12)12 - AIC:393.50205402692336
ARIMA(1, 1, 1)x(0, 0, 1, 12)12 - AIC:317.06379477367665
ARIMA(1, 1, 1)x(0, 1, 0, 12)12 - AIC:337.419420226926
ARIMA(1, 1, 1)x(0, 1, 1, 12)12 - AIC:1254.7022053579658
ARIMA(1, 1, 1)x(1, 0, 0, 12)12 - AIC:323.2841228019825
ARIMA(1, 1, 1)x(1, 0, 1, 12)12 - AIC:318.79188819596976
ARIMA(1, 1, 1)x(1, 1, 0, 12)12 - AIC:240.83183610066834
ARIMA(1, 1, 1)x(1, 1, 1, 12)12 - AIC:1243.3733466778478
      pdq      pdq_x_QDQs      aic
54  (1, 1, 0)  (1, 1, 0, 12)  238.832361

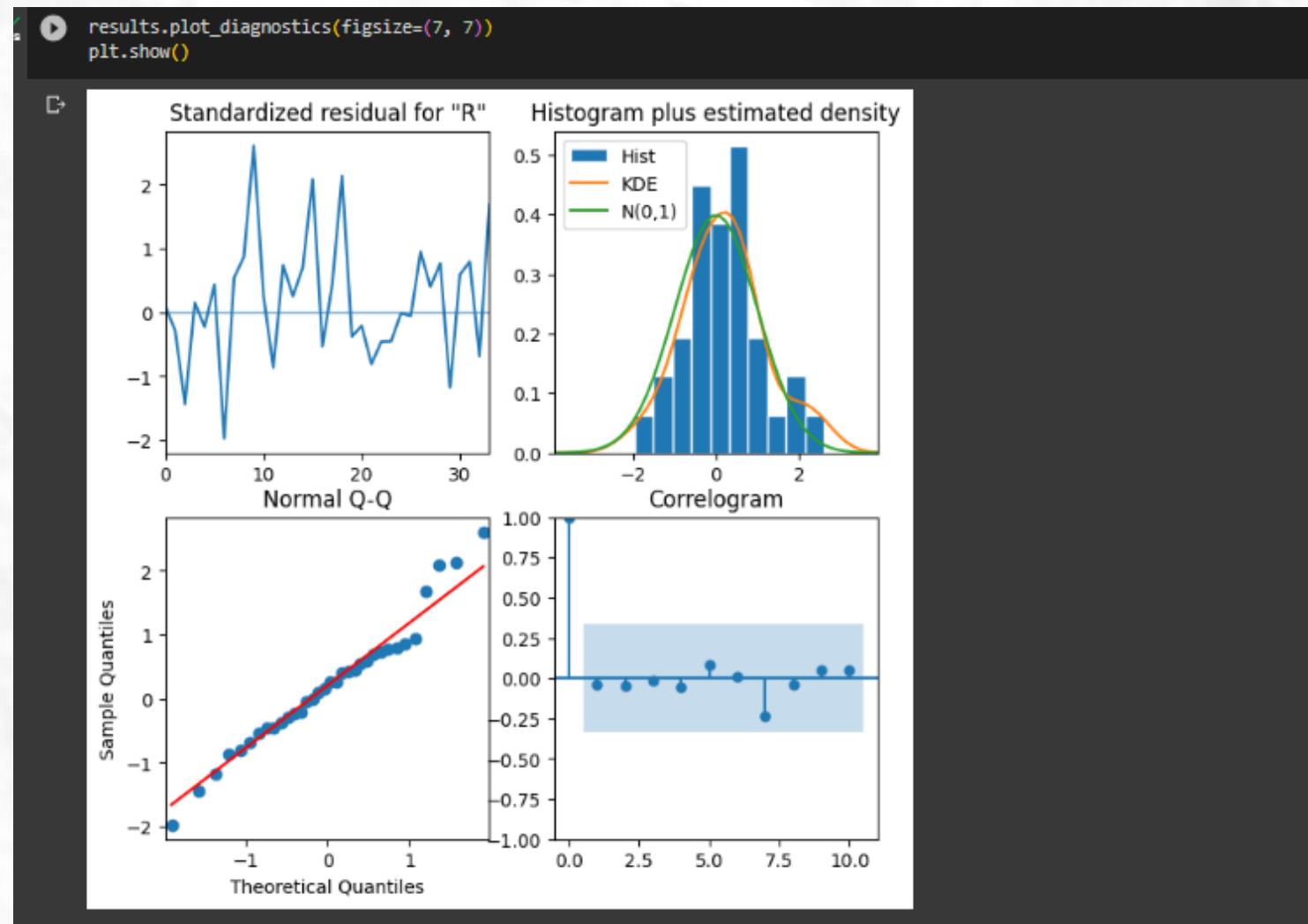
```

Điểm $pdq = (1,1,0)$ --> AIC min



SARIMAX Results						
		Dep. Variable:	Return	No. Observations:	59	
		Model:	SARIMAX(1, 1, 0)x(1, 1, 0, 12)	Log Likelihood	-116.416	
		Date:	Wed, 26 Jul 2023	AIC	238.832	
		Time:		BIC	243.322	
		Sample:	0 - 59	HQIC	240.343	
		Covariance Type:	opg			
		coef	std err	z	P> z	[0.025 0.975]
ar.L1		-0.0089	0.282	-0.032	0.975	-0.561 0.543
ar.S.L12		-0.6975	0.119	-5.875	0.000	-0.930 -0.465
sigma2		67.8727	17.490	3.881	0.000	33.592 102.153
Ljung-Box (L1) (Q): 0.04 Jarque-Bera (JB): 0.69						
Prob(Q): 0.85 Prob(JB): 0.71						
Heteroskedasticity (H): 0.48 Skew: 0.34						
Prob(H) (two-sided): 0.25 Kurtosis: 3.18						
Warnings:						

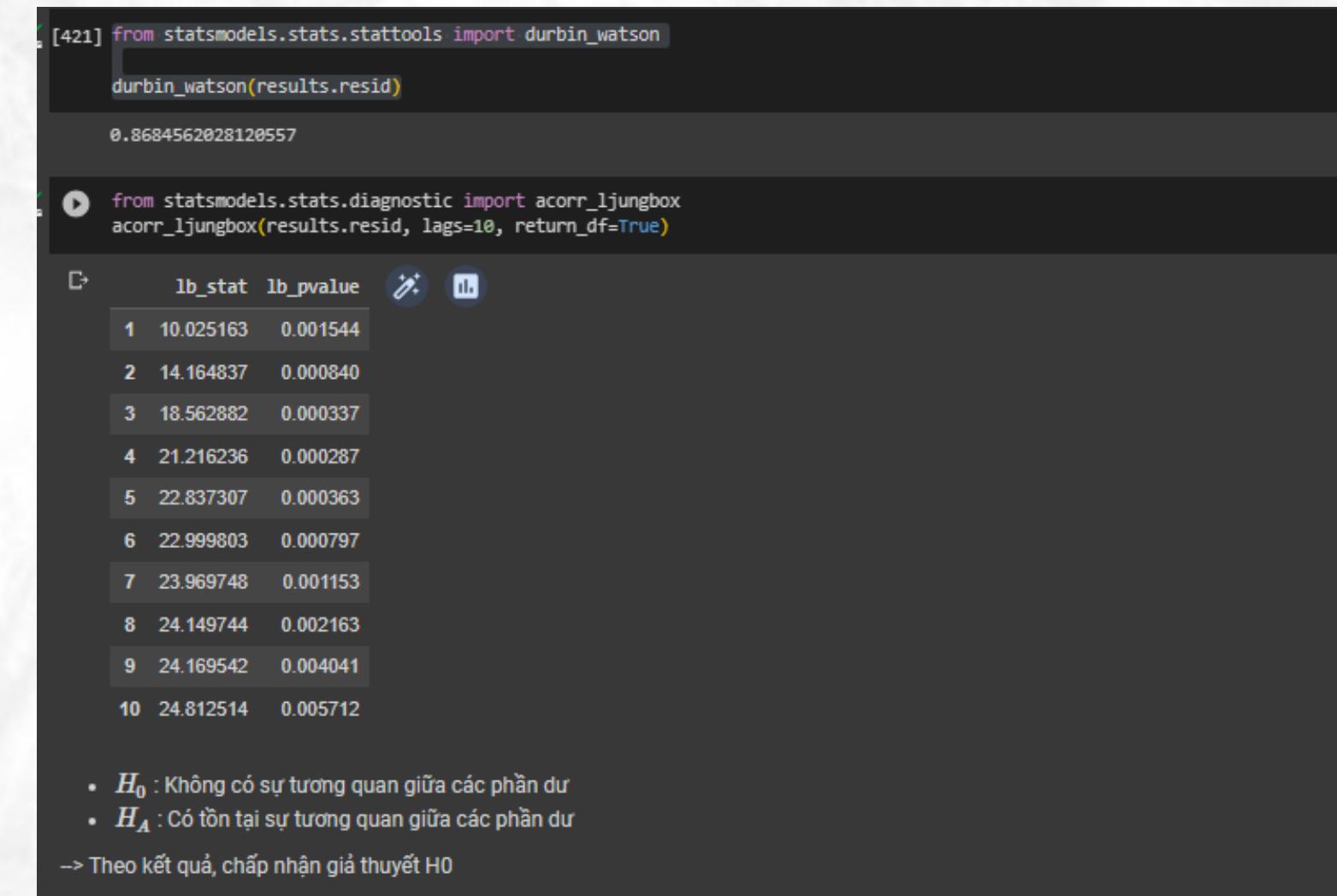
MODEL TEST



Biểu đồ **Standard residual và Correlogram** cho thấy phần dư của chuỗi thời gian có mối tương quan thấp với các phiên trễ.

Biểu đồ **QQ và histogram** dường như cho thấy residual của chuỗi tuân theo phân phối chuẩn.

Biểu đồ QQ mặc dù hai phần đuôi có chút lệch so với đường phân phối chuẩn. Tuy nhiên, đây vẫn là dấu hiệu tốt.



- **The Ljung-Box** kiểm tra có tồn tại tính tự tương quan trong một chuỗi thời gian.
- Với giá trị từ **kiểm định Durbin** cho thấy thống kê thử nghiệm này là bình thường, không vô văn đề.

TRAIN DATA

```
# Test and Train
k=round(0.75*len(df))
pred = results.get_prediction(start = k, end = len(df), dynamic=False)
pred_ci = pred.conf_int()
pred_ci.head()

lower Return upper Return
44 129.687336 161.497450
45 127.967225 159.777340
46 129.131608 160.941722
47 117.351651 149.161766
48 114.572698 146.382812

ax = df["Return"][:].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead forecast', alpha=.7)

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='r', alpha=.3)
ax.set_xlabel('Time (years)')
ax.set_ylabel('Return')
plt.legend()
plt.show()
plt.show()
```

Nhìn chung các kết quả khá khớp với dữ liệu test.

```
df_forecast = pred.predicted_mean
df_truth = df["Return"][k:]

# Compute the mean square error
df_2 = ((df_forecast - df_truth) ** 2).mean()
print('The Mean Squared Error (MSE) of the forecast is {}'.format(round(df_2, 2)))
print('The Root Mean Square Error (RMSE) of the forecast: {:.4f}'.format(np.sqrt(sum(df_for

[426] df_pred_concat = pd.concat([df_truth, df_forecast], axis=1)
df_pred_concat.reset_index()
df_pred_concat['diff'] = df_pred_concat["Return"] - df_pred_concat['predicted_mean']
display(df_pred_concat)
Total = df_pred_concat['diff'].sum()
print(Total)
```

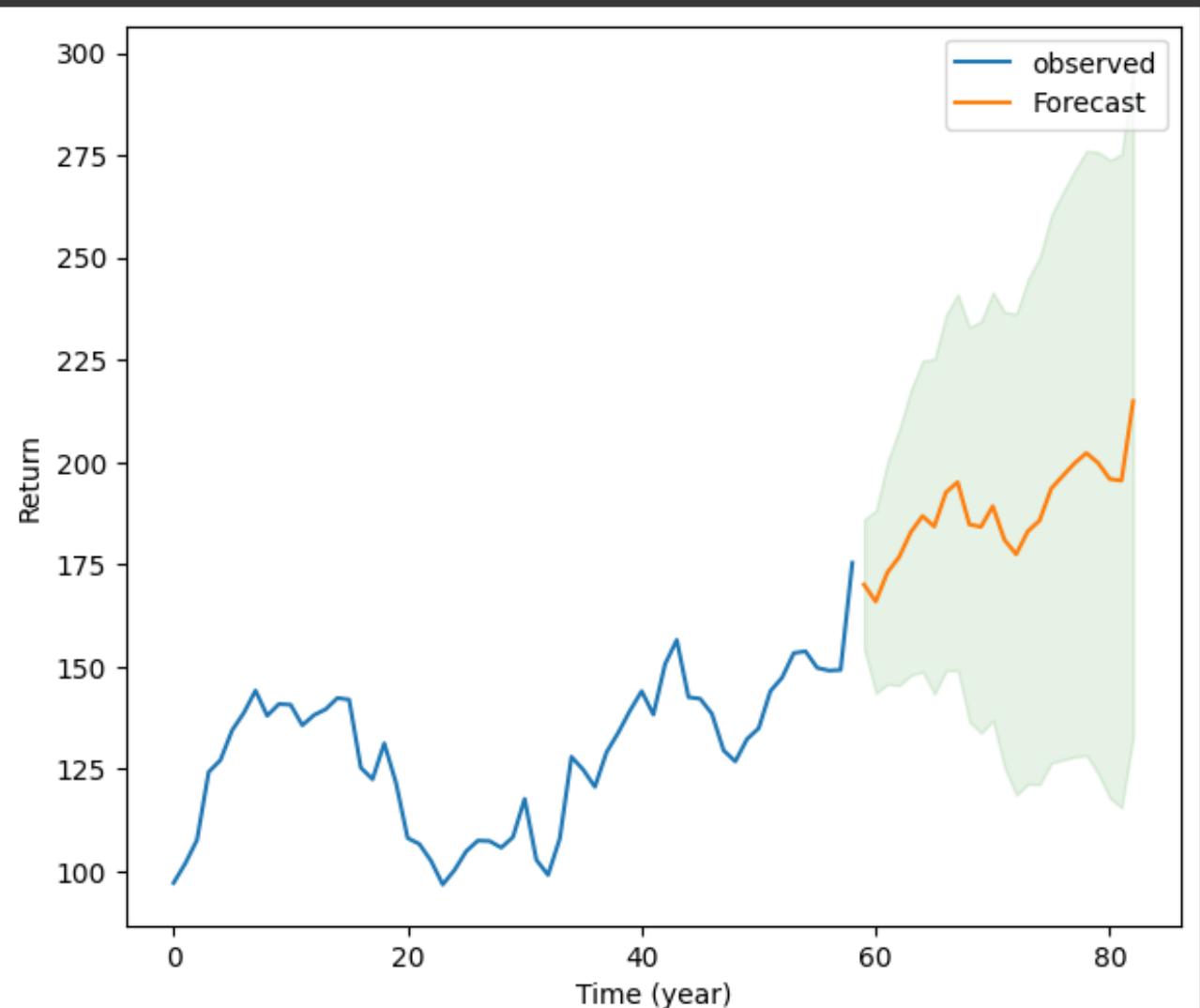
	Return	predicted_mean	diff
44	142.528571	145.592393	-3.063821
45	142.157143	143.872283	-1.715140
46	138.490000	145.036665	-6.546665
47	129.545455	133.256709	-3.711254
48	126.817647	130.477755	-3.660108
49	132.289474	132.407755	-0.118281
50	134.939130	135.375990	-0.436860
51	144.021053	136.320209	7.700843
52	147.328571	144.087092	3.241479
53	153.321429	147.109738	6.211690
54	153.750000	163.252062	-9.502062
55	149.705000	144.907771	4.797229
56	149.041667	142.601955	6.439712
57	149.194444	154.762960	-5.568516
58	175.376471	161.709857	13.666613
59	NaN	170.068578	NaN

7.734860056199565

→ Ta có thể nhận thấy sai lệch giữa kết quả predict và thực tế là rất nhỏ. Điều này được thể hiện chi tiết trong mte_pred_concat

PREDICTION

```
ax = df["Return"].plot(label='observed', figsize=(7, 6))
forecast.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(forecast_ci.index,
                 forecast_ci.iloc[:, 0],
                 forecast_ci.iloc[:, 1], color='g', alpha=.1)
ax.set_xlabel('Time (year)')
ax.set_ylabel('Return')
plt.legend()
plt.show()
```

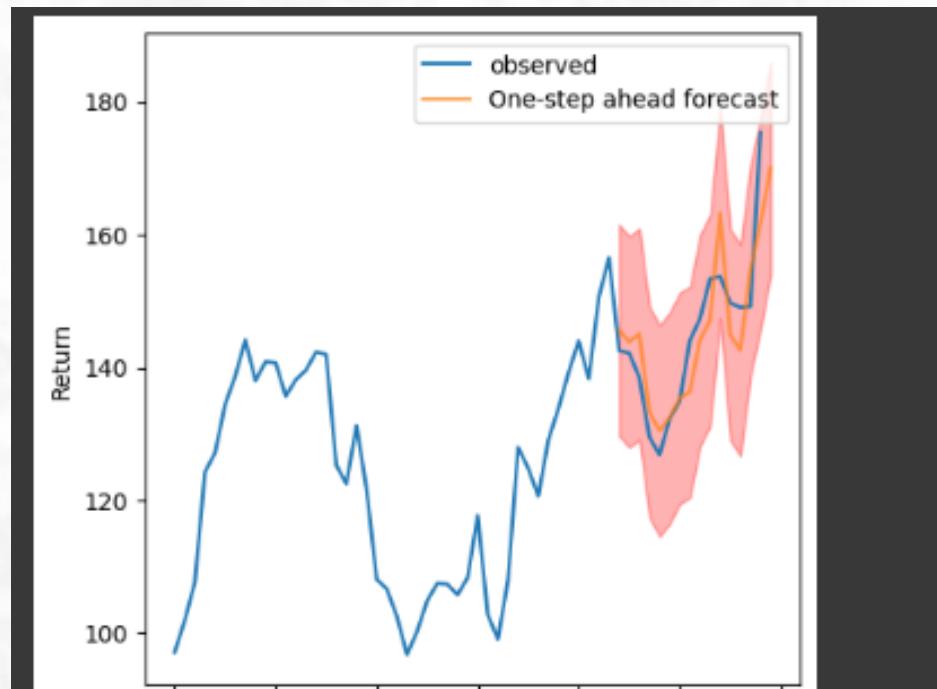


```
[430] # Get forecast of 2 year or 24 months steps ahead in future
forecast = results.get_forecast(steps= 24)
# Get confidence intervals of forecasts
forecast_ci = forecast.conf_int()
forecast_ci.head()
```

Dự báo giá trị cổ phiếu
VNM trong 2 năm tới

Nhìn chung, giá trị cổ phiếu VNM có xu
hướng tăng trong tương lai.

LIMITATION



Sai lệch giữa giá trị dự đoán và giá trị thực tế vẫn còn. Nên lựa chọn các mô hình khác để giảm độ lệch khi dự báo.

SARIMAX Results						
Dep. Variable:		Return	No. Observations:	59		
Model:	SARIMAX(1, 1, 0)x(1, 1, 0, 12)	Log Likelihood	-116.416			
Date:	Wed, 26 Jul 2023	AIC	238.832			
Time:	11:35:25	BIC	243.322			
Sample:	0 - 59	HQIC	240.343			
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
ar.L1	-0.0089	0.282	-0.032	0.975	-0.561	0.543
ar.S.L12	-0.6975	0.119	-5.875	0.000	-0.930	-0.465
sigma2	67.8727	17.490	3.881	0.000	33.592	102.153
Ljung-Box (L1) (Q):	0.04	Jarque-Bera (JB):		0.69		
Prob(Q):	0.85	Prob(JB):		0.71		
Heteroskedasticity (H):	0.48	Skew:		0.34		
Prob(H) (two-sided):	0.25	Kurtosis:		3.18		

Warnings:

Ar.L1 không có ý nghĩa thống kê, ảnh hưởng đến việc chọn mô hình



Cân nhắc các tác động khác (shocks) như war, các ngành kinh tế khác, chính sách,...