



ĐẠI HỌC QUỐC GIA TP.HCM
TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT
KHOA TÀI CHÍNH - NGÂN HÀNG

ĐỀ TÀI CUỐI KỲ

DỰ ĐOÁN XU HƯỚNG GIÁ CỦA ETHEREUM (ETH) BẰNG MÔ HÌNH ARIMA

THỰC HIỆN BỞI:

Họ và tên: Bùi Nguyễn Thùy Như

MSSV: K194141737

Môn học: 211CN0801

GIẢNG VIÊN HƯỚNG DẪN:

NCS Ths. Ngô Phú Thanh

Tháng 1/2022

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN	2
1.1 Giới thiệu Ethereum (ETH)	2
1.2 Lý do chọn đề tài	3
CHƯƠNG 2: PHƯƠNG PHÁP NGHIÊN CỨU	5
2.1 Lý do chọn mô hình ARIMA	5
2.2 Quy trình thực hiện và phân tích dự báo bằng mô hình ARIMA	5
CHƯƠNG 3: DỮ LIỆU	6
3.1 Khai báo thư viện	6
3.2 Xử lý dữ liệu	6
CHƯƠNG 4: XÂY DỰNG MÔ HÌNH	10
4.1 Phân tích xu hướng giá	10
4.2 Phân rã dữ liệu chuỗi thời gian	11
4.3 Kiểm định tính dừng	12
4.3.1 Kiểm định ADF	12
4.3.2 Kiểm định KPSS	13
4.4 Sửa lỗi tính dừng cho dữ liệu	15
4.4.1 Sử dụng biểu đồ ACF & PACF	16
4.4.2 Sử dụng kiểm định ADF, KPSS	17
4.5 Mô hình ARIMA	18
4.6 Chẩn đoán mô hình	20
4.7 Kiểm tra tự động tương quan trong phần dư	21
CHƯƠNG 5: DỰ BÁO	22
5.1 Huấn luyện dữ liệu	22
5.2 Dự báo	23
5.3 Dự báo nâng cao	24
CHƯƠNG 6: KẾT LUẬN VÀ KHUYẾN NGHỊ	26
THAM KHẢO	27
PHỤ LỤC	28

CHƯƠNG 1: TỔNG QUAN

1.1 Giới thiệu Ethereum (ETH)

Trong bối cảnh của cuộc Cách mạng Công nghiệp 4.0 phát triển mạnh mẽ trên toàn thế giới nói chung và ở Việt Nam nói riêng, tiền kỹ thuật số đang trở thành cơn sốt đầu tư mới khi các đồng tiền điện tử này đang ngày một tăng trưởng về cả thị phần lẫn giá trị giao dịch trên thị trường. Việc giao dịch tiền kỹ thuật số theo đó cũng trở nên dễ dàng và phổ biến tại Việt Nam.

Cuộc cách mạng tiền kỹ thuật sự bắt đầu với sự xuất hiện tiền mật mã (crypto currency) dẫn đến kỷ nguyên mới của thời đại số, với sự xuất hiện của loại tiền này đã gây ảnh hưởng mạnh mẽ đến thị trường tài chính hiện tại. Thậm chí, thật không khoa trương khi nhiều nhà nghiên cứu cho rằng thị trường tài chính toàn cầu đã hoàn toàn thay đổi khi hàng ngàn tỷ USD đã được phân bổ vào thị trường tiền số thế hệ mới, các giao dịch tài chính được tự động hoá bởi các đoạn mã lập trình sẵn, một thế giới mà tất cả tài sản được số hoá và được một cách ẩn danh, thị trường không còn bị điều khiển mà trở nên phi tập trung với sự đóng góp của thành phần tham gia thị trường.

Với nhiều tính năng và ưu điểm tuyệt vời như vậy, trong những năm gần đây chúng ta đã chứng kiến sự phát triển vượt bậc của thị trường tiền mã hóa. Nhất định phải kể đến, Ethereum - một đồng tiền phát triển mạnh mẽ kể từ năm 2017, giá trị vốn hóa chỉ sau Bitcoin, chiếm lĩnh vị trí thứ 2 trên thị trường.

Lý do khiến nhiều nhà đầu tư hứng thú với Ethereum so với các đồng khác, đặc biệt là Bitcoin chính là Ethereum và Bitcoin đều là tiền kỹ thuật số, nhưng nói chung mục đích sử dụng của chúng khác nhau. Đặc điểm của Ethereum là đồng coin này không được thiết kế như một giải pháp thanh toán thay thế, mà là để thúc đẩy các lập trình viên sáng tạo và vận hành ứng dụng trong mạng Ethereum.

Ethereum cho phép quy đổi tiền bạc và tài sản một cách nhanh chóng và tiết kiệm chi phí hơn nhiều so với việc phải phụ thuộc vào một chuỗi trung gian. Ethereum còn được đánh giá cao hơn so với Bitcoin vì Ethereum không chỉ là đồng tiền điện tử đơn thuần. Hơn thế, mạng Ethereum còn giúp tạo ra những thị trường online, và những giao dịch có thể được lập trình được biết với cái tên hợp đồng thông minh (smart contracts).

Nói một cách đơn giản, Bitcoin là một mạng lưới thanh toán có thể được dùng để chuyển giao giá trị giữa hai người ở bất kỳ đâu trên thế giới. Hiện tại, Bitcoin được sử dụng chủ yếu như một tài sản đầu tư. Trong khi đó Ethereum nhằm mục đích tạo ra hạ tầng cho một mạng Internet không thuộc về bất kỳ một thẩm quyền đơn lẻ nào.

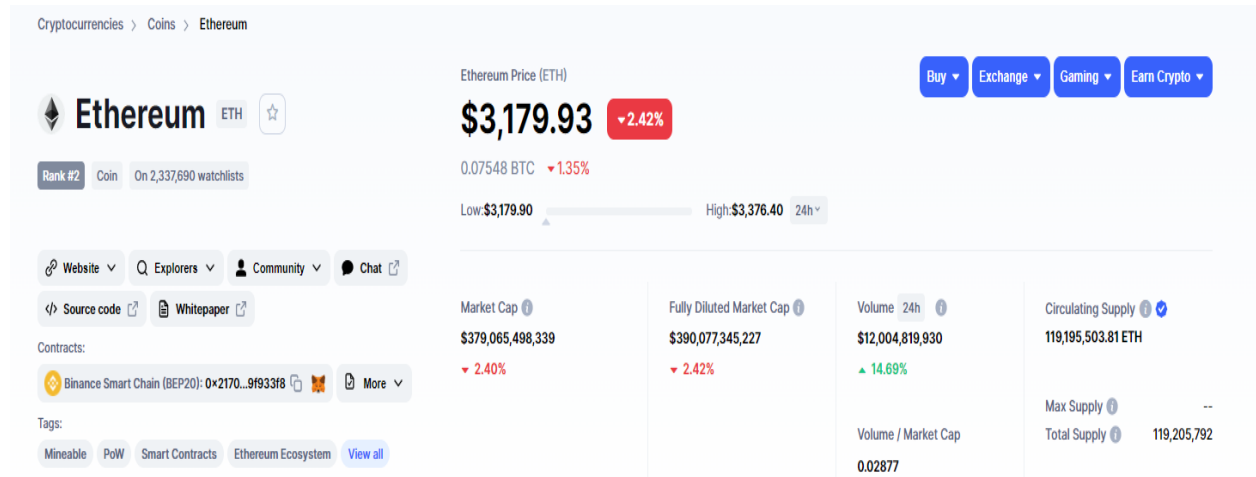
Ngoài ra khả năng mở rộng (scalability) của Ethereum cũng là một trong những ưu điểm lớn nhất của mạng lưới Ethereum nói chung ảnh hưởng đến đồng Ethereum hiện nay. Mạng lưới này trước kia vận hành bằng giao thức bằng chứng công việc (proof-of-work), tương tự như Bitcoin. Điều này có nghĩa là những người đào tiền ảo bằng máy tính chuyên dụng sẽ phải cạnh tranh giải các thuật toán phức tạp để xác nhận tính hợp lệ của giao dịch.

Phương thức trên dẫn tới sự chỉ trích nhằm vào cả Bitcoin và Ethereum từ những người lo ngại về lượng điện năng khổng lồ mà hệ thống của hai tiền ảo này tiêu thụ.

Tuy nhiên, Ethereum đã trải qua một đợt nâng cấp mang tính cách mạng tên Ethereum 2.0. Đợt nâng cấp này sẽ đưa mạng lưới Ethereum chuyển sang vận hành theo mô hình “bằng chứng cổ phần” (proof of stake). Mô hình này dựa vào “những người nắm giữ cổ phần” (stakers), là những người đã sở hữu tiền ảo Ethereum, để xử lý các giao dịch mới.

Các nhà nghiên cứu và đầu tư cho rằng việc nâng cấp sẽ giúp mạng lưới Ethereum vận hành ở quy mô lớn, xử lý được nhiều giao dịch hơn với tốc độ nhanh hơn, và hỗ trợ được những ứng dụng với hàng triệu người dùng.

Đợt nâng cấp này cũng khiến Ethereum trở nên đầy tiềm năng hơn bao giờ hết và đẩy giá Ethereum tăng.



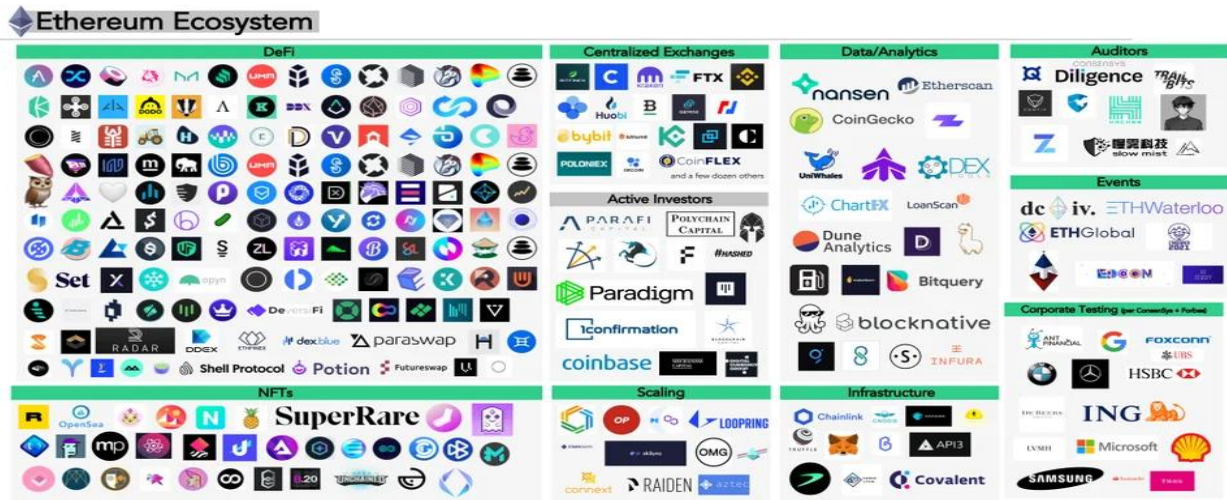
Hình 1.1: Thông tin về Ethereum (Nguồn ảnh: coinmarketcap.com ngày 18/1/2022)

Có thể thấy được, hiện nay tổng vốn hóa thị trường của Ethereum tính đến ngày 18/1/2022 là 379,151,160,889 USD với khối lượng giao dịch là 12,000,401,408 USD, chứng tỏ sự nhộn nhịp và sức hút của Ethereum với nhà đầu tư.

1.2 Lý do chọn đề tài

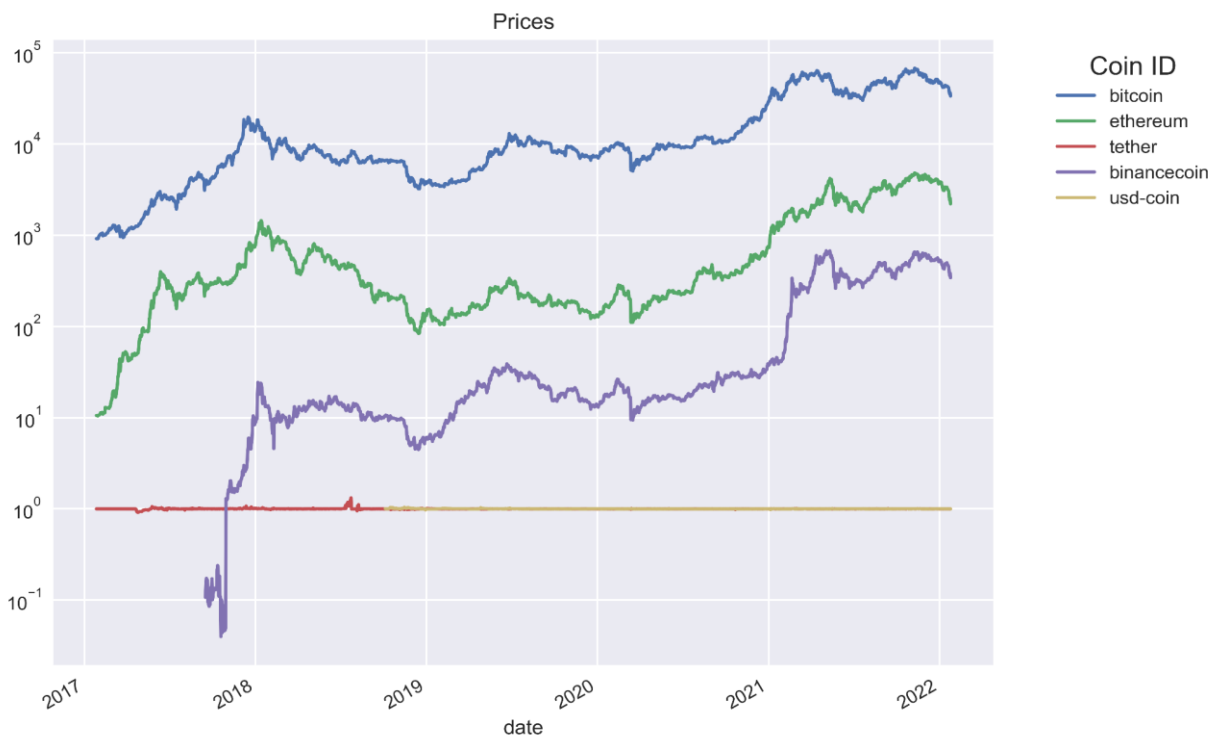
Không thể phủ nhận sự hấp dẫn ở thời điểm hiện tại của các đồng tiền mã hóa nói chung và Ethereum nói riêng đối với nhà đầu tư khi tổng vốn hóa thị trường tiền mã hóa hiện nay đạt hơn 2.000 tỷ USD (theo coinmarketcap.com).

Lí giải cho cơn sốt đầu tư tiền điện tử này có thể đến từ việc lãi suất chạm đáy trên toàn thế giới và các nỗ lực kích thích tài chính khổng lồ đã khiến nhiều người có thêm các khoản tiền dôi dư trong thời kỳ đại dịch. Ngoài ra trong thời kỳ Covid19 kéo dài, khi người dân bị hạn chế về vấn đề việc làm, đầu tư vào tiền mã hóa trở nên một giải pháp kiếm thêm thu nhập từ những đồng tiền nhàn rỗi. Ethereum với những tiềm năng và hệ sinh thái khổng lồ của mình đã thu hút các nhà đầu tư, đặc biệt là những nhà đầu tư mới gia nhập thị trường.



Hình 1.2: Hệ sinh thái của Ethereum (Nguồn: fiahub.com)

Ngoài ra "con sói" trong thế giới nghệ thuật NFT (Non-Fungible Token) đang khiến giới đầu tư và tài chính chú ý. NFT là một loại tài sản kỹ thuật số, sử dụng công nghệ chuỗi khối (blockchain) để khẳng định giá trị độc bản, không thể sao chép và tính sở hữu của chủ sở hữu tác phẩm này. Đa số NFT hiện nay dựa trên nền tảng của Ethereum và được trao đổi bằng Ethereum coin.



Hình 1.3: Biến động giá cả trong vòng 5 năm của top 5 coin

Có thể thấy xu hướng giá của Ethereum biến động tương tự như Bitcoin tăng trưởng cao và dựa vào Tether (stable coin) để neo giá

Dù vậy, nhiều người vẫn còn hoài nghi về đồng tiền này khi vào đầu tháng 1/2022, thị trường chứng kiến nhiều đợt lao dốc của các đồng tiền mã hóa. Vì vậy nghiên cứu và dự báo đồng tiền Ethereum là một chủ đề cấp thiết cũng như giúp nhà đầu tư, đặc biệt là nhà

đầu tư mới, chưa có kinh nghiệm có thể đánh giá tiềm năng, chuyên biến về giá để đưa ra quyết định một cách tối ưu nhất.

CHƯƠNG 2: PHƯƠNG PHÁP NGHIÊN CỨU

2.1 Lý do chọn mô hình ARIMA

Box & Jenkins (1970) lần đầu tiên giới thiệu mô hình ARIMA (autoregressive integrated moving average) trong phân tích chuỗi thời gian, được hiểu là phương pháp Box-Jenkins.

Mô hình ARIMA không dựa trên những biến ngoại sinh để dự báo mà dựa trên dữ liệu quá khứ để giải thích những biến chuyển của hiện tại. Một số nhận định cho rằng mô hình ARIMA sẽ phù hợp để lựa chọn dự báo trong ngắn hạn. Mô hình ARIMA hay được sử dụng để dự báo kinh tế và tài chính, đồng thời tính hiệu quả trong dự báo cũng được rất nhiều bài nghiên cứu chứng minh

2.2 Quy trình thực hiện và phân tích dự báo bằng mô hình ARIMA

**Bước 1: Nhận dạng mô hình*

Để sử dụng mô hình ARIMA(p,d,q) trong dự báo cần nhận dạng ba thành phần p,d và q của mô hình. Thành phần d của mô hình được nhận dạng thông qua kiểm định tính dừng của chuỗi thời gian. Nếu chuỗi thời gian dừng ở bậc 0 ta ký hiệu I(d=0), nếu sai phân bậc 1 của chuỗi dừng ta ký hiệu I(d=1), nếu sai phân bậc 2 của chuỗi dừng ta ký hiệu I(d=2),... Để kiểm định tính dừng của chuỗi, sử dụng kiểm định nghiệm đơn vị Dickey-Fuller cải biên (ADF).

**Bước 2: Ước lượng các tham số và lựa chọn mô hình*

Các tham số của mô hình sẽ được ước lượng bằng auto_arima qua thư viện có sẵn của python. Nó sử dụng một từng bước (thuật toán Hyndman-Khandakar) để duyệt qua không gian của các mô hình một cách hiệu quả. Chức năng auto_arima xử lý sự khác biệt dữ liệu để làm cho dữ liệu ở trạng thái tĩnh (cho dù d = 0), chọn siêu tham số và chọn mô hình tốt nhất theo AIC. Quá trình lựa chọn mô hình là quá trình thực nghiệm và so sánh các tiêu chí R-squared hiệu chỉnh, AIC và Schwarz cho đến khi ta chọn được mô hình tốt nhất cho việc dự báo.

Quy trình thực hiện và phân tích dự báo bằng mô hình ARIMA

Như đã nêu trong cuốn sách Kinh thánh Dự báo: Nguyên tắc và Thực hành, có một cách tiếp cận chung để phù hợp với mô hình ARIMA: tiền xử lý, cho đến khi dữ liệu trở nên tĩnh; nguồn cấp dữ liệu cho một hàm tính toán mô hình ARIMA; so sánh các mô hình; kiểm tra kết quả (phần dư); nếu không đủ tốt, hãy lặp lại, nếu không, hãy sử dụng mô hình kết quả để thực hiện dự báo.

**Bước 3: Xây dựng mô hình và kiểm định mô hình*

Để đảm bảo mô hình là phù hợp, sai số của mô hình phải là nhiễu trắng (white noise). Ta có thể sử dụng biểu đồ tự tương quan ACF hoặc kiểm định các cách để đưa ra kết luận phù hợp.

**Bước 4. Dự báo*

Nếu mô hình là phù hợp, mô hình sẽ được sử dụng vào việc dự báo. Dự báo bao gồm 2 phần chính đó là: dự báo trong mẫu và dự báo ngoài mẫu. Các tiêu chí được sử dụng để so sánh hiệu quả dự báo là RMSE, MAE.

CHƯƠNG 3: DỮ LIỆU

3.1 Khai báo thư viện

- Các thư viện để trích xuất biểu đồ được trực quan:

```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

```
import matplotlib.pyplot as plt
import warnings
import seaborn as sns
plt.style.use('seaborn')
# plt.style.use('seaborn-colorblind') #alternative
plt.rcParams['figure.figsize'] = [8, 4.5]
plt.rcParams['figure.dpi'] = 300
warnings.simplefilter(action='ignore', category=FutureWarning)
```

- Các thư viện để tính toán, kiểm định và dự báo:

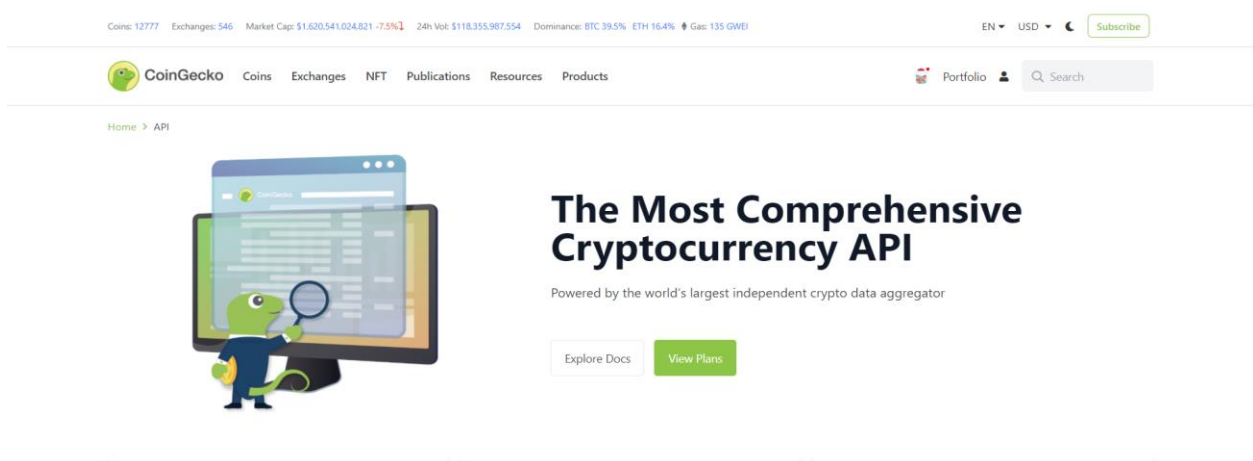
```
import numpy as np
import pandas as pd
import requests # API call
from statsmodels.tsa.stattools import adfuller, kpss
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plt.rcParams.update({'font.size': 14})
# Figure out order for ARIMA model
import pmdarima as pm
# Ignore harmless warnings
import warnings
warnings.filterwarnings('ignore')
# ARIMA model
from statsmodels.tsa.arima.model import ARIMA
import plotly.graph_objects as go
from datetime import timedelta
import statsmodels.api as sm
from statsmodels.stats.diagnostic import acorr_ljungbox
import scipy.stats as scs
```

3.2 Xử lý dữ liệu

Nghiên cứu này sử dụng giá đóng cửa của Ethereum trong vòng 5 năm (1825 ngày) theo ngày để hiển thị ARIMA tự động.

Dữ liệu được truy cập từ API của trang coingecko.com thông qua code Python. Và trang web này không yêu cầu phải có API Key.

<https://www.coingecko.com/en/api>



The screenshot shows the CoinGecko website. At the top, there's a navigation bar with links for Coins, Exchanges, NFT, Publications, Resources, and Products. Below this is a hero section with the title "The Most Comprehensive Cryptocurrency API" and a subtext "Powered by the world's largest independent crypto data aggregator". There are two buttons: "Explore Docs" and "View Plans". To the left of the text is an illustration of a green alien holding a magnifying glass over a computer screen displaying a list of cryptocurrencies.

- Đầu tiên nhập các thư viện cần thiết. Khai báo URL API của trang coingecko.com.
- Sử dụng requests để lấy dữ liệu của coin theo USD.
- Khởi tạo dataframe để chứa dữ liệu.

Input:

```
import pandas as pd
import requests
import matplotlib.pyplot as plt

# make the API call
API_URL = 'https://api.coingecko.com/api/v3'

r_coins_d = requests.get(API_URL + '/coins/markets?vs_currency=usd&order=market_cap_desc&per_page=10&page=1&sparkline=false')
d_coins = r_coins_d.json()

# create the markets dataframe
df_coins_markets = pd.DataFrame(d_coins)
```

Output:

df_coins_markets										
	id	symbol	name	image	current_price	market_cap	market_cap_rank	fully_diluted_valuation	total_volu	
0	bitcoin	btc	Bitcoin	https://assets.coingecko.com/coins/images/1/la...	41379.000000	782506280367	1	8.679314e+11	187564461	
1	ethereum	eth	Ethereum	https://assets.coingecko.com/coins/images/279/...	3112.920000	370737534624	2	NaN	127548397	
2	tether	usdt	Tether	https://assets.coingecko.com/coins/images/325/...	0.998076	78039793330	3	NaN	47273296	
3	binancecoin	bnb	Binance Coin	https://assets.coingecko.com/coins/images/825/...	460.390000	77407762375	4	7.740776e+10	1735388	
4	cardano	ada	Cardano	https://assets.coingecko.com/coins/images/975/...	1.440000	46222870203	5	6.486633e+10	4562434	
5	usd-coin	usdc	USD Coin	https://assets.coingecko.com/coins/images/6319/...	0.996135	45655543957	6	NaN	26483677	
6	solana	sol	Solana	https://assets.coingecko.com/coins/images/4128/...	135.780000	42587809158	7	NaN	1339258	
7	ripple	xrp	XRP	https://assets.coingecko.com/coins/images/44/...	0.740286	35255190223	8	7.396744e+10	2082797	
8	terra-luna	luna	Terra	https://assets.coingecko.com/coins/images/8284/...	75.630000	27055289639	9	7.558347e+10	19223421	
9	polkadot	dot	Polkadot	https://assets.coingecko.com/coins/images/1217/...	24.470000	26315288099	10	NaN	1192136	

10 rows x 26 columns

Dữ liệu lúc đầu chính là 26 trường dữ liệu của các id coin tương ứng như các trường về id, name, current_price, market_cap_rank,...

- Sau đó tạo một dictionary trong dataframe đã tạo. Chọn top 5 coin đầu tiên (phục vụ mục đích xuất ra hình ảnh 1.3). Thiết lập URL theo ngày và trong vòng 1825 ngày tương ứng với 5 năm phục vụ nghiên cứu.

Dữ liệu coingecko.com cung cấp chỉ gồm giá đóng cửa, giá mở cửa, volume và tổng vốn hóa nên chỉ API lấy về giá đóng cửa, volume và tổng vốn hóa. Sau đó thì kết hợp lại thành một dataframe.

Input:

```
df_market_dict = dict() # create a dict of dataframes

# iterate through the unique coin market ids
for coin_id in df_coins_markets.id.unique()[:5]: # first 5
    print(f'Coin ID: {coin_id}')

    # call the API
    r_market_d = requests.get(API_URL + f'/coins/{coin_id}/market_chart?vs_currency=usd&days=1825&interval=daily')

    # extract the JSON from the response
    d_market = r_market_d.json()

    # separately extract each key from the json
    prices = pd.DataFrame(d_market['prices'], columns=['date', 'prices']).set_index('date')
    market_caps = pd.DataFrame(d_market['market_caps'], columns=['date', 'market_caps']).set_index('date')
    total_volumes = pd.DataFrame(d_market['total_volumes'], columns=['date', 'total_volumes']).set_index('date')

    # combine the separate dataframes
    df_market = pd.concat([prices, market_caps, total_volumes], axis=1)

    # convert the index to a datetime dtype
    df_market.index = pd.to_datetime(df_market.index, unit='ms')

    # add the market_data dataframe to the dict, with the coin_id as the key
    df_market_dict[coin_id] = df_market

display(df_market.head())
```

Output:

Coin ID: bitcoin

	prices	market_caps	total_volumes
date			
2017-01-26	915.360000	1.476203e+10	5.834937e+07
2017-01-27	918.287500	1.481118e+10	4.902932e+07
2017-01-28	920.245000	1.484453e+10	2.188221e+07
2017-01-29	914.142500	1.474799e+10	2.028832e+07
2017-01-30	920.991146	1.486010e+10	3.016603e+07

Coin ID: ethereum

	prices	market_caps	total_volumes
date			
2017-01-26	10.555256	9.318515e+08	6.557327e+06
2017-01-27	10.509979	9.281788e+08	5.497368e+06
2017-01-28	10.554963	9.324866e+08	5.053699e+06
2017-01-29	10.430443	9.218105e+08	3.291647e+06
2017-01-30	10.570411	9.345139e+08	5.394044e+06

Coin ID: tether

	prices	market_caps	total_volumes
date			
2017-01-26	1.0	14951591.0	1812940.0
2017-01-27	1.0	14951591.0	1354660.0
2017-01-28	1.0	14951591.0	1575460.0
2017-01-29	1.0	14951591.0	748419.0
2017-01-30	1.0	14951591.0	678966.0

Coin ID: binancecoin

	prices	market_caps	total_volumes
date			
2017-09-16	0.107251	1.072506e+07	1.051223
2017-09-17	0.154041	1.540413e+07	14.678587
2017-09-18	0.173491	1.734912e+07	6.001767
2017-09-19	0.168334	1.683342e+07	3.878927
2017-09-20	0.166628	1.666279e+07	40.687619

Coin ID: usd-coin

	prices	market_caps	total_volumes
date			
2018-10-05	1.006242	0.0	31264.420430
2018-10-06	1.001530	0.0	20254.712255
2018-10-07	1.001177	0.0	49324.690669
2018-10-08	1.001906	0.0	47076.728142
2018-10-09	1.001983	0.0	55542.215509

Output là dataframe bao gồm prices (close price), market cap, total volume của top 5 đồng coin xếp hạng đầu theo coingecko.com

- Tiếp đến là tạo một dataframe chỉ chứa giá, volume, tổng vốn hóa của Ethereum để phục vụ nghiên cứu. Đồng thời sử dụng rename để đổi tên cột prices thành close cho dễ phân biệt. Sau đó đổi dữ liệu từ theo ngày sang theo tuần. Và sử dụng dataframe df_eth để chỉ có một cột giá đóng cửa để tính xuyên suốt bài nghiên cứu

```
df = df_market_dict['ethereum']

df = df.rename(columns = {'prices':'close'})

df = df.resample('W-FRI').ffill()
df_eth = df[['close']]
```

- Kiểm tra dữ liệu có đầy đủ hay bị missing value:

Input:

```
df_eth.isnull().any()
```

Output:

```
close    False
dtype: bool
```

Dữ liệu không có missing value hay NA, có thể sử dụng để bắt đầu chạy mô hình

- Kiểm tra dữ liệu cuối cùng sử dụng để chạy mô hình:

Input:

```
print(f'Downloaded {df_eth.shape[0]} rows of data.')
df_eth
```

Output:

Downloaded 262 rows of data.

	close
date	
2017-01-27	10.509979
2017-02-03	10.937686
2017-02-10	11.275641
2017-02-17	12.703484
2017-02-24	13.071871
...	...
2021-12-31	3714.945456
2022-01-07	3416.826009
2022-01-14	3256.758866
2022-01-21	3015.588778
2022-01-28	2203.522001

262 rows × 1 columns

Dữ liệu đã được xử lý tốt để bắt đầu chạy mô hình.

CHƯƠNG 4: XÂY DỰNG MÔ HÌNH

4.1 Phân tích xu hướng giá

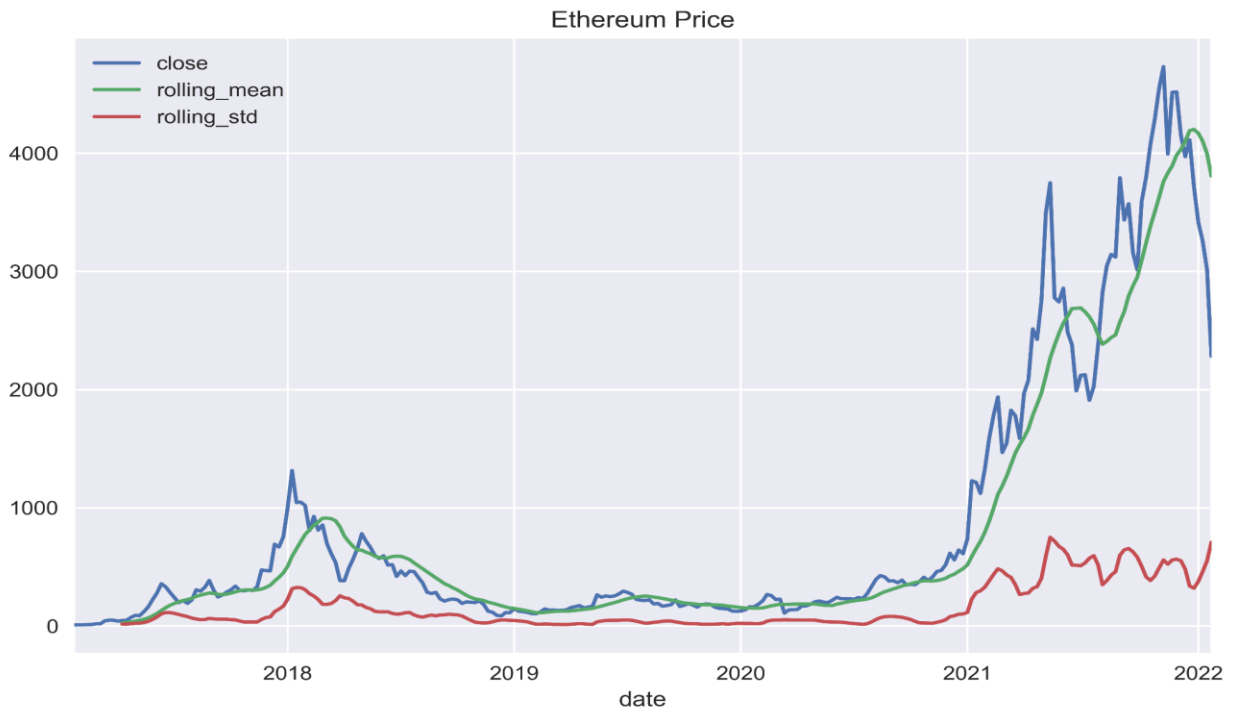
- Vẽ biểu đồ lịch sử giá so sánh cùng với giá trị trung bình và độ lệch chuẩn được tính bằng dữ liệu của, biểu đồ này sử dụng để làm mịn dữ liệu giá và phân tích xu hướng của giá Ethereum.

Input:

```
df_rolling = df_eth
WINDOW_SIZE = 12
df_rolling['rolling_mean'] = df_rolling.close.rolling(window=WINDOW_SIZE).mean()
df_rolling['rolling_std'] = df_rolling.close.rolling(window=WINDOW_SIZE).std()
df_rolling.plot(title='Ethereum Price')

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

Output:



Dựa theo đường `rolling_mean` có thể thấy dữ liệu của giá Ethereum có xu hướng đi lên và giá tăng theo thời gian. Tuy nhiên dựa vào đường `rolling_std` cho thấy khoảng từ năm 2021 đến 2022 có sự biến động khá lớn khi độ lệch chuẩn tăng đột biến vào khoảng thời gian này. Biểu đồ còn có dấu hiệu của dữ liệu có tính mùa vụ.

4.2 Phân rã dữ liệu chuỗi thời gian

Phân rã dữ liệu chuỗi thời gian chính là phân tách từng phần của dữ liệu thành các yếu tố khác nhau nhằm hiểu rõ hơn về bản chất dữ liệu.

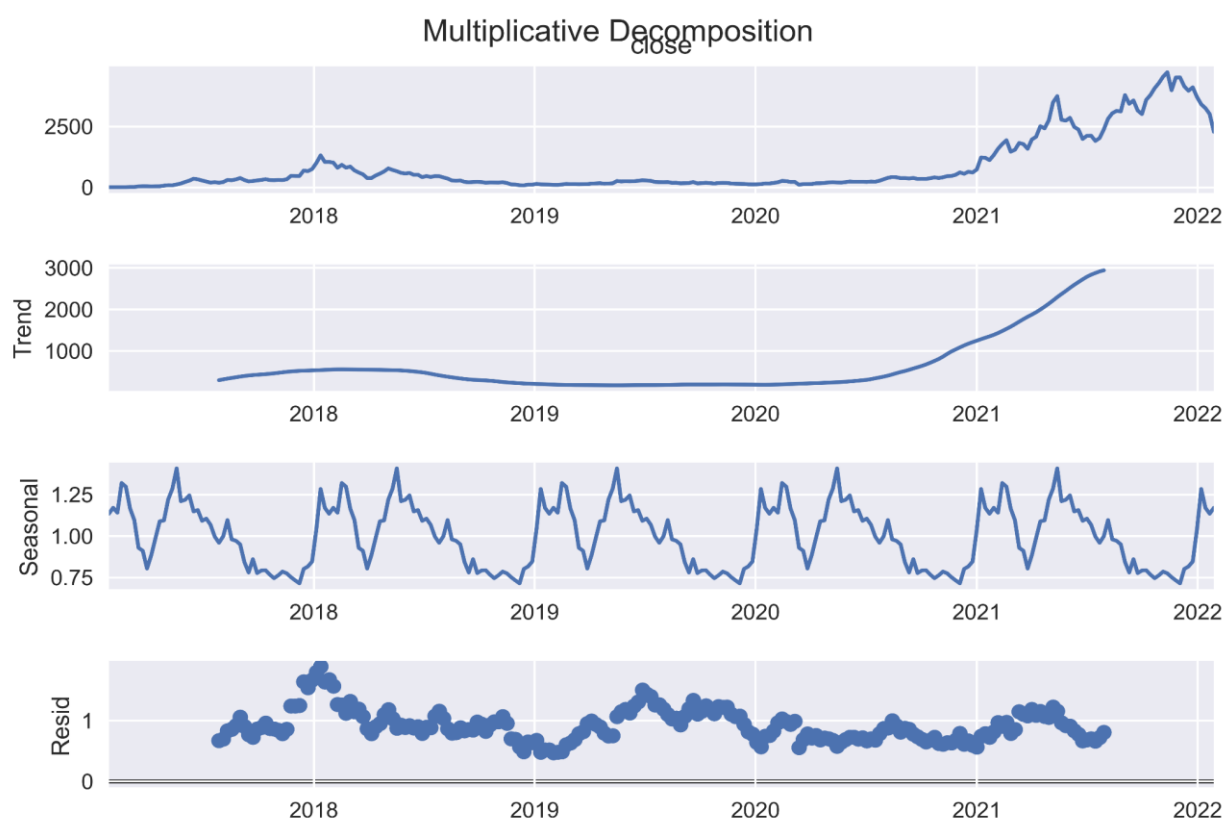
- Sử dụng gói `seasonal_decompose` trong thư viện `statsmodels` để thực hiện phân rã dữ liệu chuỗi thời gian.
- Bằng cách sử dụng mô hình tổng hợp (multiplicative model) để phân rã chuỗi dữ liệu thời gian.

Input:

```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition_results = seasonal_decompose(df_eth.close,
                                          model='multiplicative')

decomposition_results.plot() \
    .suptitle('Multiplicative Decomposition', fontsize=14)
plt.tight_layout()
plt.show()
```

Output:



Dựa vào biểu đồ trend có thể thấy có một xu hướng tăng tổng thể của dữ liệu.

Dựa vào biểu đồ Seasonal có thấy dữ liệu có tính thời vụ và quy luật thời vụ lặp lại hằng năm rất rõ ràng.

4.3 Kiểm định tính dừng

Theo Gujarati (2003) một chuỗi thời gian là dừng khi giá trị trung bình, phương sai, hiệp phương sai (tại các độ trễ khác nhau) giữ nguyên không đổi cho dù chuỗi được xác định vào thời điểm nào đi nữa. Chuỗi dừng có xu hướng trở về giá trị trung bình và những dao động quanh giá trị trung bình sẽ là như nhau. Nói cách khác, một chuỗi thời gian không dừng sẽ có giá trị trung bình thay đổi theo thời gian, hoặc giá trị phương sai thay đổi theo thời gian hoặc cả hai.

Nếu một chuỗi không dừng, chúng ta chỉ có thể nghiên cứu hành vi của nó chỉ cho riêng giai đoạn đang xem xét. Vì thế, mỗi chuỗi thời gian là một giai đoạn riêng biệt. Cho nên, chúng ta không thể khái quát hóa kết quả phân tích cho các giai đoạn khác. Đối với các mục đích dự báo, chuỗi không dừng sẽ không có giá trị ứng dụng thực tiễn.

Vì vậy kiểm định tính dừng là rất cần thiết do dữ liệu chuỗi thời gian

4.3.1 Kiểm định ADF

Kiểm định ADF (Augmented Dickey-Fuller Test) là một phương pháp kiểm định tính dừng cho dữ liệu.

Giả thuyết kiểm định:

$H_0: \beta = 0$ (Y_t là chuỗi dữ liệu không dừng)

$H_1: \beta \neq 0$ (Y_t là chuỗi dữ liệu dừng)

- Sử dụng hàm `adfuller` trong thư viện `stats` để kiểm định tính dừng

Input:

```
# ADF test
adf_test_1 = adfuller(df_eth['close'])
print(adf_test_1)
```

Output:

```
(-1.1623132204414266,
 0.6895939883116216,
 15,
 246,
 {'1%': -3.457215237265747,
  '5%': -2.873361841566324,
  '10%': -2.5730700760129555},
 3216.3306589072267)
```

- Kết quả của kiểm định ADF được lưu trong biến tuy nhiên việc sắp xếp kết quả bị rối mắt, nên tạo một hàm kiểm định ADF để kết quả hiển thị rõ ràng hơn và thuận tiện cho việc sử dụng lại sau này.

Input:

```
# Define function adf_test
def adf_test(x):
    ...
    Null Hypothesis: time series is not stationary
    Alternate Hypothesis: time series is stationary
    ...
    indices = ['Test Statistic', 'p-value',
               '# of Lags Used', '# of Observations Used']

    adf_test = adfuller(x, autolag='AIC')
    results = pd.Series(adf_test[0:4], index = indices)

    for key, value in adf_test[4].items():
        results[f'Critical Value ({key})'] = value

    return results
```

Output:

```
# Test ADF again with defined function
adf_test(df_eth['close'])

Test Statistic      -1.162313
p-value             0.689594
# of Lags Used      15.000000
# of Observations Used 246.000000
Critical Value (1%)  -3.457215
Critical Value (5%)  -2.873362
Critical Value (10%) -2.573070
dtype: float64
```

Kết quả cho thấy p-value có giá trị là $0.689594 > 0.05$ nên chấp nhận giả thuyết H_0 , kết luận đây là chuỗi dữ liệu không dừng.

Tuy nhiên vẫn nên sử dụng đồng thời kiểm định KPSS để đối chiếu kết quả về tính dừng của dữ liệu.

4.3.2 Kiểm định KPSS

Kiểm định KPSS (Kwiatkowski-Phillips-Schmidt-Shin Test) là một phương pháp kiểm định tính dừng cho dữ liệu.

Giả thuyết kiểm định:

$H_0: \beta = 0$ (Y_t là chuỗi dữ liệu dừng)

$H_1: \beta \neq 0$ (Y_t là chuỗi dữ liệu không dừng)

- Sử dụng hàm `kpss` trong thư viện `stats` để kiểm định tính dừng

Input:

```
# KPSS test
kpss(df_eth['close'])
```

Output:

```
(1.2646546200758548,
 0.01,
 10,
 {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739})
```

- Kết quả của kiểm định KPSS được lưu trong biến tuy nhiên việc sắp xếp kết quả bị rối mắt, nên tạo một hàm kiểm định KPSS để kết quả hiển thị rõ ràng hơn và thuận tiện cho việc sử dụng lại sau này.

Input:

```
def kpss_test(x, h0_type='c'):
    """
    Function for performing the Kwiatkowski-Phillips-Schmidt-Shin test for stationarity

    Null Hypothesis: time series is stationary
    Alternate Hypothesis: time series is not stationary

    Parameters
    -----
    x: pd.Series / np.array
        The time series to be checked for stationarity
    h0_type: str{'c', 'ct'}
        Indicates the null hypothesis of the KPSS test:
        * 'c': The data is stationary around a constant(default)
        * 'ct': The data is stationary around a trend

    Returns
    -----
    results: pd.DataFrame
        A DataFrame with the KPSS test's results
    """
    indices = ['Test Statistic', 'p-value', '# of Lags']

    kpss_test = kpss(x, regression=h0_type)
    results = pd.Series(kpss_test[0:3], index=indices)

    for key, value in kpss_test[3].items():
        results[f'Critical Value ({key})'] = value

    return results
```

Output:

```
kpss_test(df_eth['close'])
```

```
Test Statistic      1.264655
p-value             0.010000
# of Lags           10.000000
Critical Value (10%) 0.347000
Critical Value (5%)  0.463000
Critical Value (2.5%) 0.574000
Critical Value (1%)  0.739000
dtype: float64
```

Kết quả cho thấy p-value có giá trị là $0.010000 < 0.05$ nên chấp nhận giả thuyết H_1 , kết luận đây là chuỗi dữ liệu không dừng.

Từ kiểm định KPSS và ADF cho ra kết quả trùng khớp nên kết luận đây là dữ liệu chuỗi không dừng

4.4 Sửa lỗi tính dừng cho dữ liệu

Theo Ramanathan (2002) hầu hết các chuỗi thời gian về kinh tế là không dừng vì chúng thường có một xu hướng tuyến tính hoặc mũ theo thời gian. Tuy nhiên có thể biến đổi chúng về chuỗi dừng thông qua quá trình sai phân.

- Tạo thêm một cột dữ liệu mới chứa giá trị sai phân bậc 1 của dữ liệu có sẵn:

Input:

```
# Create difference column
df_eth['diff'] = df_eth['close'].diff(1)
print(df_eth[['close', 'diff']].head())
```

Output:

	close	diff
date		
2017-01-27	10.509979	NaN
2017-02-03	10.937686	0.427706
2017-02-10	11.275641	0.337955
2017-02-17	12.703484	1.427843
2017-02-24	13.071871	0.368387

Vì sai phân bậc 1 là lấy giá ngày hôm sau trừ đi giá hôm trước, tất cả chia cho giá hôm trước nên dòng đầu tiên của dữ liệu sai phân bậc 1 bị rỗng.

- Dùng dropna để xóa dòng đầu tiên của dữ liệu bị rỗng

Input:

```
df_eth_y2 = df_eth[['close', 'diff']]
# Delete NA
df_eth_y2 = df_eth_y2.dropna()
print(df_eth_y2.head())
```

Output:

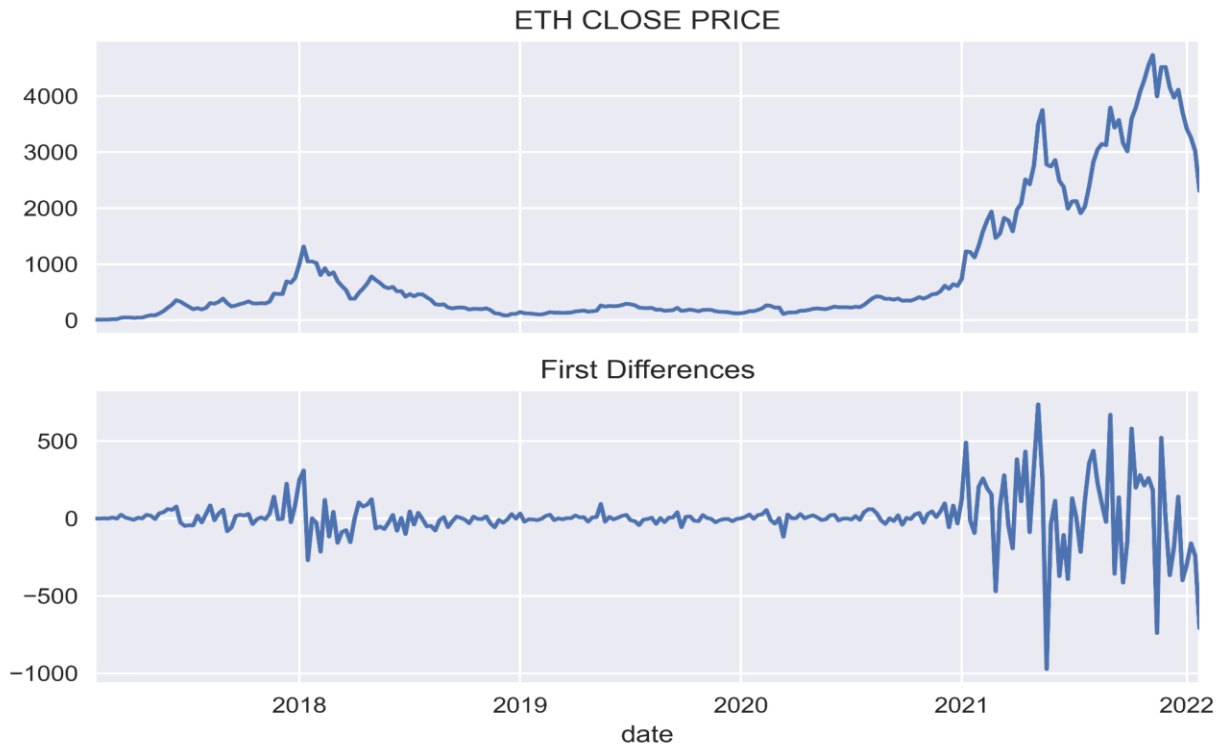
	close	diff
date		
2017-02-03	10.937686	0.427706
2017-02-10	11.275641	0.337955
2017-02-17	12.703484	1.427843
2017-02-24	13.071871	0.368387
2017-03-03	19.585749	6.513878

- Vẽ biểu đồ thể hiện dữ liệu ban đầu và dữ liệu sau khi lấy sai phân bậc 1

Input:


```
# Draw plot
fig, ax = plt.subplots(2, sharex = True)
df_eth_y2['close'].plot(ax = ax[0], title = 'ETH CLOSE PRICE')
df_eth_y2['diff'].plot(ax = ax[1], title = 'First Differences')
```

Output:



- Kiểm tra lại tính dừng của dữ liệu khi lấy sai phân bậc 1:

4.4.1 Sử dụng biểu đồ ACF & PACF

Tự tương quan (ACF - AutoCorrelation Function): Tự tương quan là một khái niệm quan trọng trong chuỗi thời gian. Hầu hết các chuỗi thời gian sẽ có sự tương quan với giá trị trễ của nó và các giá trị càng gần nhau thì tương quan càng mạnh hoặc các giá trị cùng thuộc 1 chu kỳ của chuỗi thì sẽ có tương quan cao (chẳng hạn như cùng tháng trong chu kỳ năm hay cùng quý trong chu kỳ năm). Chính vì vậy hệ số này mới có tên là tự tương quan. Hệ số tự tương quan được viết tắt là ACF và thường dùng để tìm ra độ trễ của quá trình trung bình trượt để xây dựng các mô hình như ARIMA, GARCH, ARIMAX,... và kiểm tra yếu tố mùa vụ.

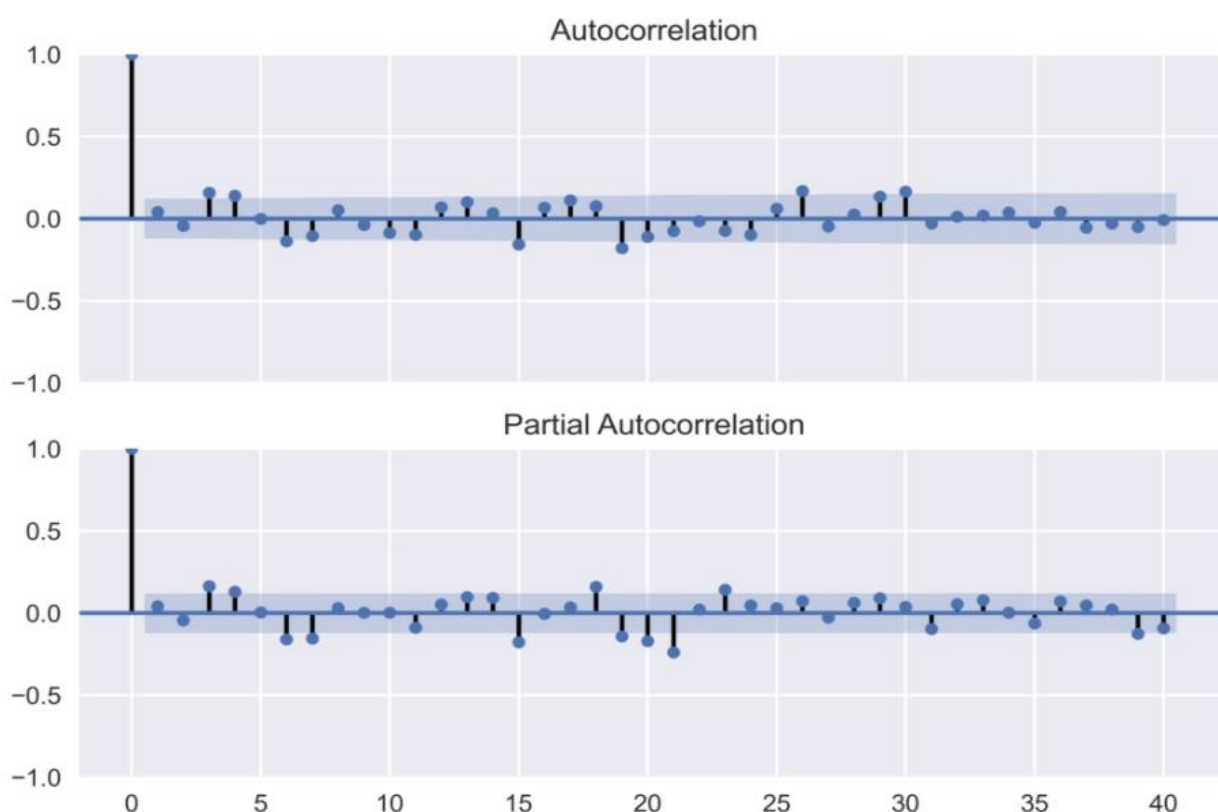
Tự tương quan riêng phần (PACF - Partial AutoCorrelation Function): Về cơ bản tương quan riêng phần cũng là chỉ số đo lường hệ số tương quan như ACF. Tuy nhiên vẫn có sự khác biệt đó là hệ số tương quan này loại bỏ ảnh hưởng của các chuỗi độ trễ trung gian

- Vẽ biểu đồ các hệ số tự tương quan ACF và PACF theo các bậc liên tiếp thông qua hàm `plot_acf` và `plot_pacf` của `statsmodels` như bên dưới:

Input:

```
# ACF & PACF plot
fig, ax = plt.subplots(2, sharex = True)
plot_acf(df_eth_y2['diff'], ax = ax[0], lags = 40, alpha = 0.05)
plot_pacf(df_eth_y2['diff'], ax = ax[1], lags = 40, alpha = 0.05)
```

Output:



Biểu đồ ACF & PACF cho thấy dữ liệu sai phân bậc 1 đảm bảo được tính dừng

4.4.2 Sử dụng kiểm định ADF, KPSS

- Gọi lại hàm `adf_test` và `kpss_test` được thiết lập từ trước cho cột `diff` để kiểm tra tính dừng

Input:

```
adf_results = adf_test(df_eth_y2['diff'])
kpss_results = kpss_test(df_eth_y2['diff'])

print('ADF test statistic: {:.2f} (p-val: {:.2f})'.format(adf_results['Test Statistic'],
                                                         adf_results['p-value']))
print('KPSS test statistic: {:.2f} (p-val: {:.2f})'.format(kpss_results['Test Statistic'],
                                                         kpss_results['p-value']))
```

Output:

```
ADF test statistic: -3.41 (p-val: 0.01)
KPSS test statistic: 0.09 (p-val: 0.10)
```

Có thể thấy p-value của kiểm định ADF < 0.05 và p-value của kiểm định KPSS > 0.05 đều chứng tỏ dữ liệu sai phân bậc 1 đảm bảo về tính dừng

Như vậy, thông qua biểu đồ ACF, PACF và kiểm định ADF, KPSS cho thấy dữ liệu sai phân bậc 1 đảm bảo về tính dừng.

- Một số nghiên cứu có thể phải lấy bậc cao hơn để đảm bảo yêu cầu về tính dừng, một cách nữa để chắc chắn dữ liệu có tính dừng ở bậc 1 chính là sử dụng `ndiffs`, `nsdiffs` của gói `pmdarima.arima`

Input:

```
from pmdarima.arima import ndiffs, nsdiffs
print(f"Suggested # of differences (ADF): {ndiffs(df_eth_y2.close, test='adf')}")
print(f"Suggested # of differences (KPSS): {ndiffs(df_eth_y2.close, test='kpss')}")
```

Output:

```
Suggested # of differences (ADF): 1
Suggested # of differences (KPSS): 1
```

Vậy kết luận bậc phù hợp của dữ liệu là bậc 1

4.5 Mô hình ARIMA

Thông qua những kiểm định ở trên, chuỗi sai phân bậc 1 đạt yêu cầu về tính dừng nên tham số d sẽ nhận giá trị là 1

Với biểu đồ ACF/PACF, ta nên chọn hệ số p và d nhỏ nhất và thường dưới 5. Có thể thu được một số kịch bản như sau:

- ARIMA(0, 1, 0)
- ARIMA(1, 1, 1)
- ARIMA(2, 1, 2)
- ARIMA(0, 1, 1)

Tuy nhiên, ta sẽ chọn kịch bản ARIMA(0, 1, 0) vì hệ số thấp nhất của nó.

- Mô hình ARIMA có thể được xây dựng qua arima thông qua package statsmodels. Điều mà chúng ta cần thực hiện chỉ là khai báo bậc của mô hình ARIMA. Khai báo ARIMA(0, 1, 0) thông qua cú pháp như sau.

Input:

```
arima = sm.tsa.arima.ARIMA(df_eth['close'], order=(0, 1, 0)).fit()
arima.summary()
```

Output:

SARIMAX Results

Dep. Variable:	close		No. Observations:	262		
Model:	ARIMA(0, 1, 0)		Log Likelihood	-1708.496		
Date:	Tue, 25 Jan 2022		AIC	3418.991		
Time:	01:12:03		BIC	3422.556		
Sample:	01-27-2017		HQIC	3420.424		
	- 01-28-2022					
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
sigma2	2.84e+04	1066.368	26.630	0.000	2.63e+04	3.05e+04
Ljung-Box (L1) (Q):	0.43	Jarque-Bera (JB):	899.49			
Prob(Q):	0.51	Prob(JB):	0.00			
Heteroskedasticity (H):	11.51	Skew:	-0.53			
Prob(H) (two-sided):	0.00	Kurtosis:	12.03			

Warnings:

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

Như vậy xét trên khía cạnh mô hình thống kê thì tất cả các hệ số ước lượng đều có ý nghĩa thống kê với mức ý nghĩa 95% bởi vì $p\text{-value} > 0.05$

Việc lựa chọn mô hình tốt nhất trong lớp các mô hình ARIMA, ta có thể căn cứ trên chỉ số AIC.

Một trong những tiêu chí thường được sử dụng để lựa chọn mô hình đó là chỉ số AIC (Akaike Information Criteria). Tiêu chí thông tin này là một công cụ ước tính lỗi dự báo và do đó đánh giá chất lượng tương đối của các mô hình thống kê trên một tập hợp dữ liệu nhất định. Giả sử có một tập hợp các mô hình được xây dựng trên cùng một bộ dữ liệu, AIC ước tính chất lượng của từng mô hình trong mối liên quan đến từng mô hình khác. Do đó, AIC cung cấp một phương tiện để lựa chọn mô hình.

- Trên python đã hỗ trợ tìm kiếm mô hình ARIMA phù hợp thông qua auto arima của thư viện pmdarima, dựa vào chỉ số AIC.

Input:

```
auto_arima = pm.auto_arima(df_eth['close'], trace = 1,
                           error_action = 'ignore',
                           suppress_warnings = True,
                           seasonal = False,
                           stepwise = True,
                           approximation = False,
                           n_jobs = -1,
                           seasonal_test = False)
```

Output:

```
Performing stepwise search to minimize aic
ARIMA(2,1,2)(0,0,0)[0] intercept : AIC=inf, Time=0.56 sec
ARIMA(0,1,0)(0,0,0)[0] intercept : AIC=3420.242, Time=0.01 sec
ARIMA(1,1,0)(0,0,0)[0] intercept : AIC=3421.789, Time=0.02 sec
ARIMA(0,1,1)(0,0,0)[0] intercept : AIC=3421.734, Time=0.07 sec
ARIMA(0,1,0)(0,0,0)[0] : AIC=3418.991, Time=0.01 sec
ARIMA(1,1,1)(0,0,0)[0] intercept : AIC=3422.395, Time=0.23 sec
```

```
Best model: ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 0.927 seconds
```

```
print(auto_arima.summary())
```

```

SARIMAX Results
=====
Dep. Variable:          y      No. Observations:          262
Model:                SARIMAX(0, 1, 0)      Log Likelihood          -1708.496
Date:                Tue, 25 Jan 2022      AIC              3418.991
Time:                01:12:04      BIC              3422.556
Sample:              0      HQIC              3420.424
                  - 262
Covariance Type:      opg
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
sigma2          2.84e+04    1066.368     26.630      0.000     2.63e+04     3.05e+04
=====
Ljung-Box (L1) (Q):              0.43      Jarque-Bera (JB):              899.49
Prob(Q):                      0.51      Prob(JB):              0.00
Heteroskedasticity (H):          11.51      Skew:              -0.53
Prob(H) (two-sided):              0.00      Kurtosis:             12.03
=====

```

Warnings:

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


Auto arima sẽ lần lượt thay đổi các tham số q, d, p của mô hình và chọn ra mô hình với các tham số có chỉ số AIC là thấp nhất. Trong trường hợp này, mô hình tối ưu mà auto ARIMA chọn cũng đồng nhất theo cách thủ công là ARIMA (0,1,0)

4.6 Chẩn đoán mô hình

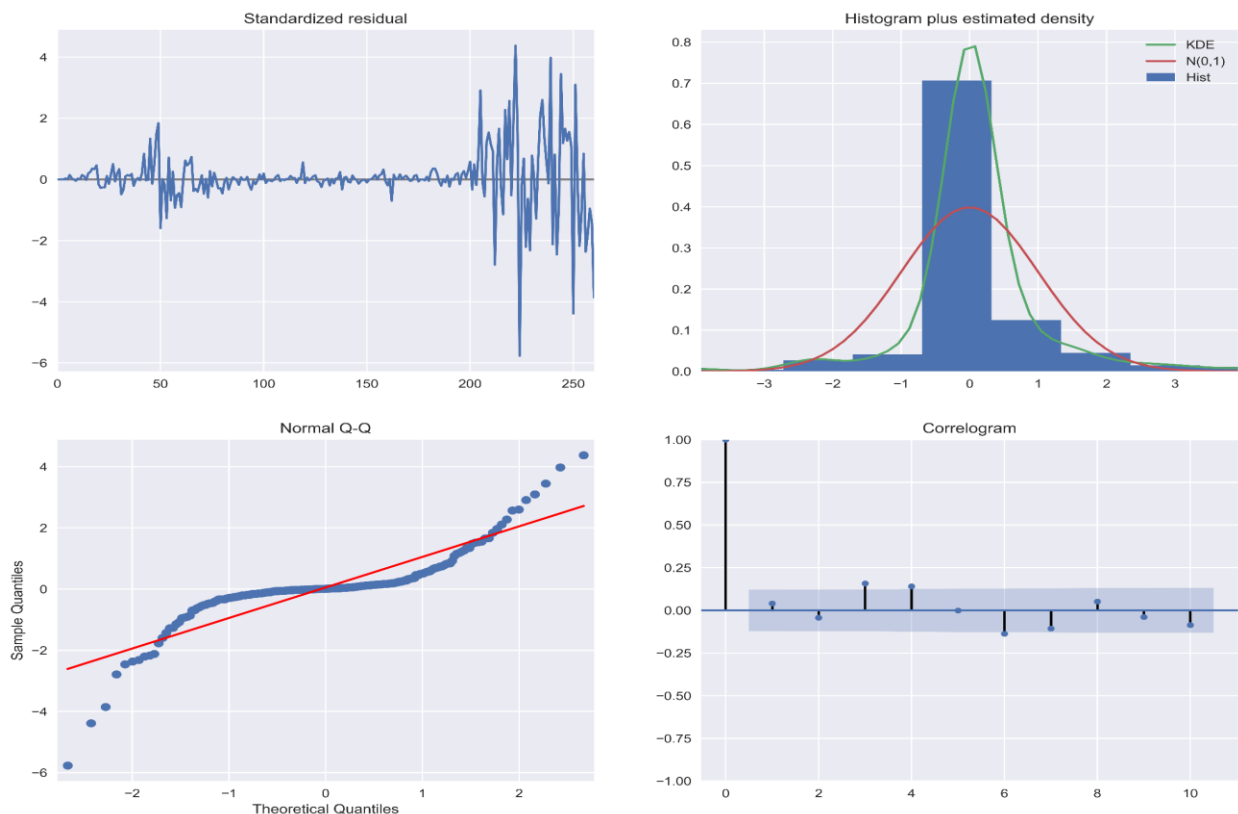
Khi lắp các mô hình ARIMA, điều quan trọng là phải chạy chẩn đoán mô hình đảm bảo rằng không có giả định nào do mô hình đưa ra bị vi phạm.

- Sử dụng `plot_diagnostics` cho phép ta nhanh chóng tạo ra chẩn đoán mô hình và điều tra xem có bất kỳ hành vi bất thường nào không

Input:

```
auto_arima.plot_diagnostics(figsize=(15, 12))
plt.show()
```

Output:



Mối quan tâm chính của ta là đảm bảo phần dư của mô hình của ta không tương quan và được phân phối bình thường với giá trị trung bình bằng 0. Nếu mô hình ARIMA không đáp ứng các đặc tính này, thì đó là một dấu hiệu tốt cho thấy nó có thể được cải thiện hơn nữa.

Trong trường hợp này, chẩn đoán mô hình của ta gợi ý rằng phần dư của mô hình được phân phối bình thường dựa trên những điều sau:

Trong biểu đồ trên cùng bên phải, ta thấy rằng đường KDE màu đỏ theo xu hướng chung với đường $N(0,1)$ (trong đó $N(0,1)$ là ký hiệu chuẩn cho phân phối chuẩn với giá trị trung bình là 0 và độ lệch chuẩn là 1). Đây là một dấu hiệu khá tốt cho thấy phần dư được phân phối chuẩn.

Biểu đồ QQ ở phía dưới bên trái cho thấy phân phối có thứ tự của phần dư (chấm màu xanh) tuân theo xu hướng tuyến tính của các mẫu được lấy từ phân phối chuẩn chuẩn với $N(0, 1)$, nó bám khá sát với phân phối đường màu đỏ, đây là một dấu hiệu mạnh mẽ cho thấy phần dư được phân phối bình thường.

Phần dư theo thời gian (ô trên cùng bên trái) có dấu hiệu của tính thời vụ. Tuy nhiên dựa vào biểu đồ tự tương quan (tức là tương quan) ở phía dưới bên phải, cho thấy rằng phần dư của chuỗi thời gian có mối tương quan thấp với các version trễ của chính nó.

Những quan sát đó giúp ta kết luận rằng mô hình của ta tạo ra sự phù hợp thỏa đáng có thể giúp ta hiểu dữ liệu chuỗi thời gian của bạn và dự báo các giá trị trong tương lai.

Mặc dù có một sự phù hợp ưng ý, một số thông số của mô hình ARIMA của ta có thể được thay đổi để cải thiện sự phù hợp với mô hình. Ví dụ: tìm kiếm lưới của ta chỉ được coi là một tập hợp giới hạn các kết hợp tham số, vì vậy ta có thể tìm thấy các mô hình tốt hơn nếu ta mở rộng tìm kiếm lưới.

4.7 Kiểm tra tự động tương quan trong phần dư

* Kiểm định Ljung-Box

H0: Không tồn tại hiện tượng tự tương quan

H1: Tồn tại hiện tượng tự tương quan

- Sử dụng `acorr_ljungbox` để kiểm định

Input

```
import statsmodels.api as sm
from statsmodels.stats.diagnostic import acorr_ljungbox
import scipy.stats as scs
```

```
import matplotlib.pyplot as plt
sm.stats.acorr_ljungbox(arima.resid, return_df=True)
```

Output:

	lb_stat	lb_pvalue
1	0.435519	0.509293
2	0.926945	0.629095
3	7.576397	0.055628
4	12.874142	0.011907
5	12.874454	0.024584
6	17.922908	0.006428
7	20.963782	0.003824
8	21.686650	0.005531
9	22.101845	0.008562
10	24.143391	0.007229

Với 3 kiểm định Ljung-box ban đầu lần lượt các độ trễ khác nhau có kết quả p-value > 0.05, chấp nhận giả thuyết H0 không tồn tại hiện tượng tương quan. Tuy nhiên vào các kiểm định Ljung-box sau đó, p-value < 0.05 chứng tỏ vẫn tồn tại hiện tượng tự tương quan.

Điều này cho thấy độ biến động của thời kì sau phụ thuộc vào các biến động thời kì trước đó.

CHƯƠNG 5: DỰ BÁO

5.1 Huấn luyện dữ liệu

- Chia dữ liệu thành tập train và tập test. Tập test sẽ là dữ liệu được thu thập trong 3 tháng gần đây tương ứng với 12 tuần.

Input:

```
# Split data into training and testing
print(df_eth.shape)
train = df_eth.iloc[:-12]
test = df_eth.iloc[-12:]
print(train.shape, test.shape)
```

Output:

```
(262, 4)
(250, 4) (12, 4)
```

test

	close	rolling_mean	rolling_std	diff
date				
2021-11-12	4732.924450	3761.137883	559.387833	182.910015
2021-11-19	3993.846595	3833.710783	524.470096	-739.077855
2021-11-26	4515.843300	3893.922662	559.701532	521.996705
2021-12-03	4519.441028	3984.211087	566.461227	3.597728
2021-12-10	4153.333311	4032.546570	552.793427	-366.107717
2021-12-17	3971.559766	4100.237395	481.239773	-181.773544
2021-12-24	4113.529932	4191.907120	339.228090	141.970166
2021-12-31	3714.945456	4201.909378	321.322405	-398.584476
2022-01-07	3416.826009	4170.435138	378.301070	-298.119447
2022-01-14	3256.758866	4102.260031	461.629949	-160.067142
2022-01-21	3015.588778	3996.217661	552.309485	-241.170089
2022-01-28	2413.355501	3818.162749	685.818859	-602.233277

- Sau khi đã tìm ra được mô hình ARIMA tốt nhất là ARIMA (0,1,0). Chúng ta sẽ áp dụng mô hình đó để huấn luyện cho tập train.

Input:

```
model = ARIMA(train['close'], order=(0,1,0))
```

```
# Train the model
model = model.fit()
print(model.summary())
```

Output:

```

SARIMAX Results
=====
Dep. Variable:          close    No. Observations:          250
Model:                 ARIMA(0, 1, 0)    Log Likelihood          -1601.330
Date:                 Tue, 25 Jan 2022    AIC                   3204.660
Time:                 11:08:36    BIC                   3208.177
Sample:              01-27-2017    HQIC                  3206.076
                  - 11-05-2021
Covariance Type:      opg
=====
              coef    std err          z      P>|z|      [0.025    0.975]
-----
sigma2      2.257e+04    767.078     29.418     0.000     2.11e+04     2.41e+04
=====
Ljung-Box (L1) (Q):           1.16    Jarque-Bera (JB):           1571.18
Prob(Q):                     0.28    Prob(JB):                 0.00
Heteroskedasticity (H):       8.52    Skew:                    -0.09
Prob(H) (two-sided):          0.00    Kurtosis:                 15.30
=====

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

```

5.2 Dự báo

- Dự báo từ mô hình đã được học tập thông qua hàm `predict()`. Hàm này tạo ra một vòng lặp liên tiếp dự báo qua các bước thời gian.

```

# Make prediction on test set
start = len(train)
end = len(train) + len(test)
pred = model.predict(start=start+1, end=end, typ='levels')
pred.index = pd.to_datetime(test.index, format="%Y-%m-%d")

```

- In ra giá trị dự báo trung bình trong 12 tuần so sánh với giá trị thực tế

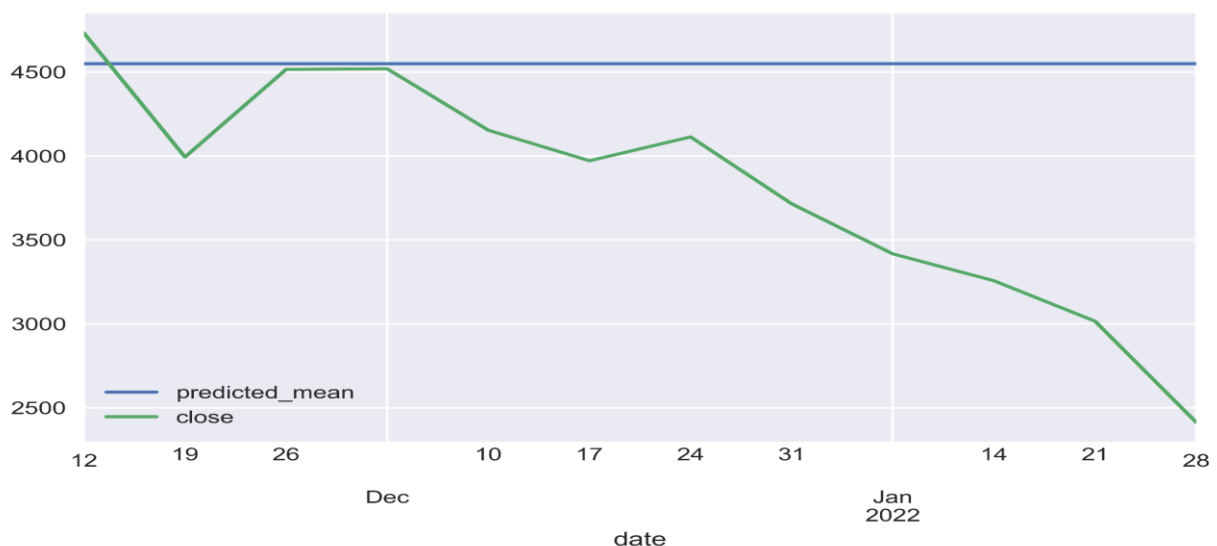
Input:

```

pred.index = pd.to_datetime(pred.index, format="%Y-%m-%d")
test.index = pd.to_datetime(test.index, format="%Y-%m-%d")
pred.plot(legend=True)
test['close'].plot(legend=True)

```

Output:



Có thể thấy giá trị dự báo trung bình ở mức xấp xỉ 4500 USD tuy nhiên xu hướng giá trong 3 tháng gần đây đang bị sụt giảm mạnh. Vì vậy cần thiết so sánh trung bình giá của tập test

- Tính trung bình giá của tập test thông qua hàm mean:

Input:

```
test['close'].mean()
```

Output:

```
3816.2438892574887
```

Có thể thấy trung bình của tập test thấp hơn nhiều so với trung bình của dự báo. Chính vì thế cần kiểm tra chất lượng của mô hình trên tập dự báo. Vì vậy cần thông qua MSE và RMSE để kiểm tra dự báo.

MSE (mean square error): Trung bình tổng bình phương sai số. Đo trung bình bình phương của các lỗi – nghĩa là chênh lệch bình phương trung bình giữa các giá trị ước tính và giá trị ước tính.

RMSE (root mean square error): Phương sai hoặc độ lệch chuẩn của chuỗi dự báo so với thực tế, một biện pháp thường được sử dụng trong những khác biệt giữa các giá trị (mẫu hoặc các giá trị dân) được dự đoán bởi một mô hình hay một ước lượng và các giá trị quan sát được.

- Tính MSE và RMSE thông qua nhập các hàm cần tính và format thành số chỉ có 2 số thập phân

Input:

```
from sklearn.metrics import mean_squared_error
from math import sqrt
import numpy as np
```

```
# MSE The Mean Squared Error of our forecasts is
mse = ((pred - test['close']) ** 2).mean()
print('MSE of our forecasts {}'.format(round(mse, 2)))
# RMSE The Root Mean Squared Error of
print('RMSE our forecasts is {}'.format(round(np.sqrt(mse), 2)))
```

Output:

```
MSE of our forecasts 975002.85
RMSE our forecasts is 987.42
```

MSE: Trung bình tổng bình phương sai số là 975002.85

RMSE: Biên độ giao động của giá trị dự báo xung quanh giá trị thực tế là 983.24.

Các chỉ số này khá cao có thể kết luận mô hình không khớp với thực tế lắm.

5.3 Dự báo nâng cao

- Dự báo mô hình bắt đầu từ ngày 01/10/2021, cho 3 tháng sắp tới

Input:

```

pred_os = model.get_prediction(start='2021-10-01')
pred_ci = pred_os.conf_int()
ax = df_eth['close']['2017:'].plot(label='Observed')
pred_os.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7, figsize=(14, 7))
ax.fill_between(pred_ci.index,
               pred_ci.iloc[:, 0],
               pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Year')
ax.set_ylabel('ETH Close Price')
plt.legend()
plt.show()

```

Output:



Có thể thấy giữ quan sát và dự báo khá khớp với nhau. Tuy nhiên từ biểu đồ cũng có thể thấy mô hình có thể bị overfitting vì khớp với nhau rất nhiều.

- Dự báo trong một thời gian dài tiếp theo (trong 3 năm)

Input:

```

# from datetime import timedelta
step = 156
day_last_train = train.index.max() + timedelta(days=1)
date_list = [day_last_train + timedelta(days=x) for x in range(step)]

```

```

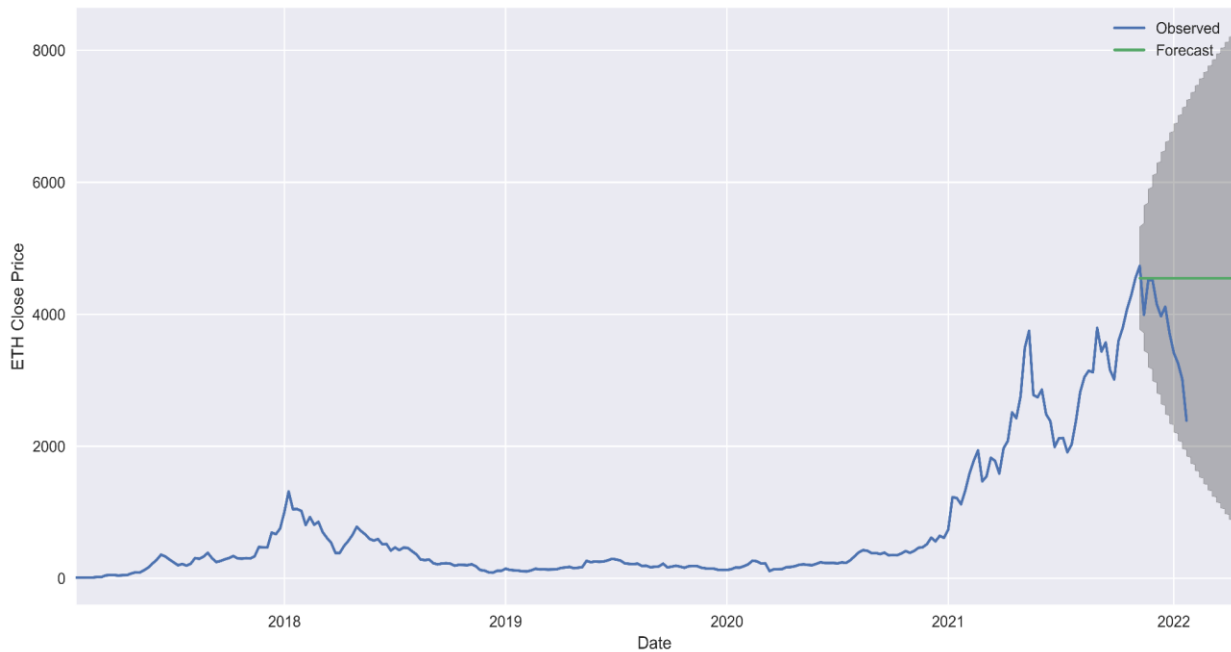
## Dự báo tiếp
pred_uc = model.get_forecast(steps=step)
pred_ci = pred_uc.conf_int()

df_mean = pred_uc.predicted_mean
df_mean.index = date_list
pred_ci.index = date_list

ax = df_eth['close'].plot(label='Observed', figsize=(14, 7))
df_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
               pred_ci.iloc[:, 0],
               pred_ci.iloc[:, 1], color='k', alpha=.25)
ax.set_xlabel('Date')
ax.set_ylabel('ETH Close Price')
plt.legend()
plt.show()

```

Output:



CHƯƠNG 6: KẾT LUẬN VÀ KHUYẾN NGHỊ

Hiện tại mô hình dự báo đang chưa có sự đánh giá chính xác khi độ chênh lệch giữa dự báo và thực tế đang khá cao. Tuy nhiên xét về xu hướng giá thì mô hình dự báo hướng tăng trưởng trong tương lai khá chính xác. Trong thời gian ngắn kế tiếp thì mô hình dự đoán xu hướng giá sẽ đi ngang hoặc có thể đi lên.

Bài toán phân tích dự báo đồng Ethereum chỉ dựa trên số liệu giá đóng cửa nên có phần chưa đánh giá tốt toàn cảnh của mô hình. Để phân tích và dự đoán chính xác thì cần dựa vào và loại bỏ lạm phát để sự phân tích và dự báo chính xác hơn.

Hơn nữa, trong công tác dự báo nên sử dụng kết hợp các mô hình, bởi vì mỗi mô hình có một ưu thế khác nhau trong quá trình dự báo và kết quả của mỗi mô hình sẽ đóng vai trò hỗ trợ, kiểm định cho mô hình còn lại. Vì vậy trong tương lai nên sử dụng thêm các mô hình học sâu như LSTM hay RNN để kết hợp cho ra dự báo chính xác nhất.

Tóm lại, mục đích của nghiên cứu là cung cấp một mô hình để các nhà đầu tư tham khảo, nhưng không nên tin tưởng hoàn toàn vào các dự đoán của mô hình. Chắc chắn là nghiên cứu vẫn còn nhiều điểm cần cải thiện về mặt mô hình hoặc có thể có một số lỗi trong code hay một số lỗi ẩn khác mà tác giả chưa phát hiện ra, chưa khắc phục hoàn toàn. Số liệu cũng không thể lấy hết vì đồng Ethereum vẫn là một đồng mới và chưa có số liệu đầy đủ. Sau cùng, tác giả sẽ cố gắng khắc phục những điểm yếu, cố gắng khác phương pháp hoàn thiện mô hình và dự đoán chuỗi thời gian.

THAM KHẢO

- [1] Phong, N.A và cộng sự (2020). Sách tham khảo "Ứng dụng Python trong tài chính", NXB Đại học Quốc Gia Tp.HCM
- [2] Khoa học dữ liệu – Khanh's blog
- [3] Quang, B. (2010). Ứng dụng mô hình ARIMA để dự báo VNINDEX.
- [4] Khôi, H. (2020). Xây dựng ứng dụng phân tích dự báo doanh thu doanh nghiệp golf

PHỤ LỤC

```
#!/usr/bin/env python
# coding: utf-8

# # Import Library

# In[1]:

get_ipython().run_line_magic('matplotlib', 'inline')
get_ipython().run_line_magic('config', "InlineBackend.figure_format =
'retina'")

# In[2]:

import matplotlib.pyplot as plt
import warnings
import seaborn as sns
plt.style.use('seaborn')
# plt.style.use('seaborn-colorblind') #alternative
plt.rcParams['figure.figsize'] = [8, 4.5]
plt.rcParams['figure.dpi'] = 300
warnings.simplefilter(action='ignore', category=FutureWarning)

# In[3]:

import numpy as np
import pandas as pd
import requests # API call
from statsmodels.tsa.stattools import adfuller, kpss
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plt.rcParams.update({'font.size': 14})
# Figure out order for ARIMA model
import pmdarima as pm
# Ignore harmless warnings
import warnings
warnings.filterwarnings('ignore')
# ARIMA model
from statsmodels.tsa.arima.model import ARIMA
import plotly.graph_objects as go
from datetime import timedelta
import statsmodels.api as sm
from statsmodels.stats.diagnostic import acorr_ljungbox
import scipy.stats as scs

# # Data Processing

# In[4]:

import pandas as pd
import requests
import matplotlib.pyplot as plt

# make the API call
API_URL = 'https://api.coingecko.com/api/v3'
```



```

r_coins_d = requests.get(API_URL +
    '/coins/markets?vs_currency=usd&order=market_cap_desc&per_page=10&page=1&spar
kline=false')
d_coins = r_coins_d.json()

# create the markets dataframe
df_coins_markets = pd.DataFrame(d_coins)

# In[5]:

df_coins_markets

# In[6]:

df_market_dict = dict() # create a dict of dataframes

# iterate through the unique coin market ids
for coin_id in df_coins_markets.id.unique()[:5]: # first 5
    print(f'Coin ID: {coin_id}')

    # call the API
    r_market_d = requests.get(API_URL +
    f'/coins/{coin_id}/market_chart?vs_currency=usd&days=1825&interval=daily')

    # extract the JSON from the response
    d_market = r_market_d.json()

    # separately extract each key from the json
    prices = pd.DataFrame(d_market['prices'], columns=['date',
'prices']).set_index('date')
    market_caps = pd.DataFrame(d_market['market_caps'], columns=['date',
'market_caps']).set_index('date')
    total_volumes = pd.DataFrame(d_market['total_volumes'], columns=['date',
'total_volumes']).set_index('date')

    # combine the separate dataframes
    df_market = pd.concat([prices, market_caps, total_volumes], axis=1)

    # convert the index to a datetime dtype
    df_market.index = pd.to_datetime(df_market.index, unit='ms')

    # add the market_data dataframe to the dict, with the coin_id as the key
    df_market_dict[coin_id] = df_market

    display(df_market.head())

# In[7]:

fig, ax = plt.subplots(figsize=(9, 7))
for k, v in df_market_dict.items():
    v.plot(y='prices', label=f'{k}', ax=ax)

ax.set_title('Prices')
ax.set_yscale('log')
ax.legend(title='Coin ID', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()

```

```

# In[8]:

df = df_market_dict['ethereum']

# In[9]:

df = df.rename(columns = {'prices':'close'})

# In[10]:

df = df.resample('W-FRI').ffill()
df_eth = df[['close']]

# In[11]:

df_eth.isnull().any()

# In[12]:

print(f'Downloaded {df_eth.shape[0]} rows of data.')

df_eth

# In[13]:

df_eth.tail(20)

# In[14]:

# View the properties of the dataframe

print(df_eth.info())
df_eth.plot(x= None, y='close', legend=True, figsize=(20,5))
df_eth[: -10]

# # Visualization of rolling mean and standard deviation

# In[15]:

df_rolling = df_eth
WINDOW_SIZE = 12
df_rolling['rolling_mean'] =
df_rolling.close.rolling(window=WINDOW_SIZE).mean()
df_rolling['rolling_std'] =
df_rolling.close.rolling(window=WINDOW_SIZE).std()
df_rolling.plot(title='Ethereum Price')

```

```

plt.tight_layout()
plt.show()

# In[16]:

from statsmodels.tsa.seasonal import seasonal_decompose
decomposition_results = seasonal_decompose(df_eth.close,
                                          model='multiplicative')
decomposition_results.plot()
Decomposition', fontsize=14)
plt.tight_layout()
plt.show()

# # Stationarity test

# In[17]:

# ADF test
adfuller(df_eth['close'])

# In[18]:

# KPSS test
kpss(df_eth['close'])

# In[19]:

# Define function adf_test
def adf_test(x):
    '''
    Null Hypothesis: time series is not stationary
    Alternate Hypothesis: time series is stationary
    '''
    indices = ['Test Statistic', 'p-value',
               '# of Lags Used', '# of Observations Used']

    adf_test = adfuller(x, autolag='AIC')
    results = pd.Series(adf_test[0:4], index = indices)

    for key, value in adf_test[4].items():
        results[f'Critical Value ({key})'] = value

    return results

# In[20]:

# Test ADF again with defined function
adf_test(df_eth['close'])

# In[21]:

```

```

def kpss_test(x, h0_type='c'):
    '''
        Function for performing the Kwiatkowski-Phillips-Schmidt-Shin test for
        stationarity

        Null Hypothesis: time series is stationary
        Alternate Hypothesis: time series is not stationary

        Parameters
        -----
        x: pd.Series / np.array
            The time series to be checked for stationarity
        h0_type: str{'c', 'ct'}
            Indicates the null hypothesis of the KPSS test:
            * 'c': The data is stationary around a constant(default)
            * 'ct': The data is stationary around a trend

        Returns
        -----
        results: pd.DataFrame
            A DataFrame with the KPSS test's results
    '''

    indices = ['Test Statistic', 'p-value', '# of Lags']

    kpss_test = kpss(x, regression=h0_type)
    results = pd.Series(kpss_test[0:3], index=indices)

    for key, value in kpss_test[3].items():
        results[f'Critical Value ({key})'] = value

    return results

# In[22]:

kpss_test(df_eth['close'])

# In[23]:

# import matplotlib.pyplot as plt
# from statsmodels.tsa.stattools import adfuller, kpss
# from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
# plt.rcParams.update({'font.size': 14})

# # Fix stationarity for data

# In[24]:

# Create difference column
df_eth['diff'] = df_eth['close'].diff(1)
print(df_eth[['close', 'diff']].head())

# In[25]:

```

```

df_eth_y2 = df_eth[['close','diff']]
# Delete NA
df_eth_y2 = df_eth_y2.dropna()
print(df_eth_y2.head())

# In[26]:

# Draw plot
fig, ax = plt.subplots(2, sharex = True)
df_eth_y2['close'].plot(ax = ax[0], title = 'ETH CLOSE PRICE')
df_eth_y2['diff'].plot(ax = ax[1], title = 'First Differences')

# In[27]:

# ACF & PACF plot
fig, ax = plt.subplots(2, sharex = True)
plot_acf(df_eth_y2['diff'], ax = ax[0], lags = 40, alpha = 0.05)
plot_pacf(df_eth_y2['diff'], ax = ax[1], lags = 40, alpha = 0.05)

# In[28]:

adf_results = adf_test(df_eth_y2['diff'])
kpss_results = kpss_test(df_eth_y2['diff'])

print('ADF test statistic: {:.2f} (p-val: {:.2f})'.format(adf_results['Test
Statistic'],
                                                         adf_results['p-
value'])))
print('KPSS test statistic: {:.2f} (p-val: {:.2f})'.format(kpss_results['Test
Statistic'],
                                                         kpss_results['p-value'])))

# In[29]:

from pmdarima.arima import ndiffs, nsdiffs
print(f"Suggested # of differences (ADF): {ndiffs(df_eth_y2.close,
test='adf')}")
print(f"Suggested # of differences (KPSS): {ndiffs(df_eth_y2.close,
test='kpss')}")

# # ARIMA Model

# In[30]:

arima = sm.tsa.arima.ARIMA(df_eth['close'], order=(0, 1, 0)).fit()
arima.summary()

# In[31]:

auto_arima = pm.auto_arima(df_eth['close'], trace = 1,

```

```

        error_action = 'ignore',
        suppress_warnings = True,
        seasonal = False,
        stepwise = True,
        approximation = False,
        n_jobs = -1,
        seasonal_test = False)

# In[32]:

print(auto_arima.summary())

# In[33]:

auto_arima.plot_diagnostics(figsize=(15, 12))
plt.show()

# # Ljung-Box's Test

# In[34]:

import statsmodels.api as sm
from statsmodels.stats.diagnostic import acorr_ljungbox
import scipy.stats as scs

# In[35]:

import matplotlib.pyplot as plt
sm.stats.acorr_ljungbox(arima.resid, return_df=True)

# # Prediction

# In[36]:

# Spilt data into training and testing
print(df_eth.shape)
train = df_eth.iloc[:-12]
test = df_eth.iloc[-12:]
print(train.shape, test.shape)

# In[37]:

test

# In[38]:

model = ARIMA(train['close'], order=(0,1,0))

```



```

# In[39]:

# Train the model
model = model.fit()
print(model.summary())

# In[40]:

# Make prediction on test set
start = len(train)
end= len(train) + len(test)
pred = model.predict(start=start+1,end=end,typ='levels')
pred.index = pd.to_datetime(test.index, format="%Y-%m-%d")

# In[41]:

pred.index = pd.to_datetime(pred.index, format="%Y-%m-%d")
test.index = pd.to_datetime(test.index, format="%Y-%m-%d")
pred.plot(legend=True)
test['close'].plot(legend=True)

# In[42]:

test['close'].mean()

# In[43]:

from sklearn.metrics import mean_squared_error
from math import sqrt
import numpy as np

# In[44]:

# MSE The Mean Squared Error of our forecasts is
mse = ((pred - test['close']) ** 2).mean()
print('MSE of our forecasts {}'.format(round(mse, 2)))
# RMSE The Root Mean Squared Error of
print('RMSE our forecasts is {}'.format(round(np.sqrt(mse), 2)))

# In[45]:

pred_os = model.get_prediction(start='2021-10-01')
pred_ci = pred_os.conf_int()
ax = df_eth['close']['2017:'].plot(label='Observed')
pred_os.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7,
figsize=(14, 7))
ax.fill_between(pred_ci.index,
pred_ci.iloc[:, 0],
pred_ci.iloc[:, 1], color='k', alpha=.2)
ax.set_xlabel('Year')

```

```

ax.set_ylabel('ETH Close Price')
plt.legend()
plt.show()

# In[46]:

# from datetime import timedelta
step = 156
day_last_train = train.index.max() + timedelta(days=1)
date_list = [day_last_train + timedelta(days=x) for x in range(step)]

# In[47]:

## Dự báo tiếp
pred_uc = model.get_forecast(steps=step)
pred_ci = pred_uc.conf_int()

df_mean = pred_uc.predicted_mean
df_mean.index = date_list
pred_ci.index = date_list

ax = df_eth['close'].plot(label='Observed', figsize=(14, 7))
df_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
               pred_ci.iloc[:, 0],
               pred_ci.iloc[:, 1], color='k', alpha=.25)
ax.set_xlabel('Date')
ax.set_ylabel('ETH Close Price')
plt.legend()
plt.show()

# In[48]:

df_market_dict['ethereum'].to_excel("K194141737_Bui Nguyen Thuy
Nhu_FinalData.xlsx")

# In[ ]:

```