TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT

BÁO CÁO ĐỀ TÀI

DỰ BÁO GIÁ ĐÓNG CỬA CỦA ĐỒNG BITCOIN CASH COIN BẰNG MÔ PHỎNG MONTE CARLO VÀ MÔ HÌNH ARIMA

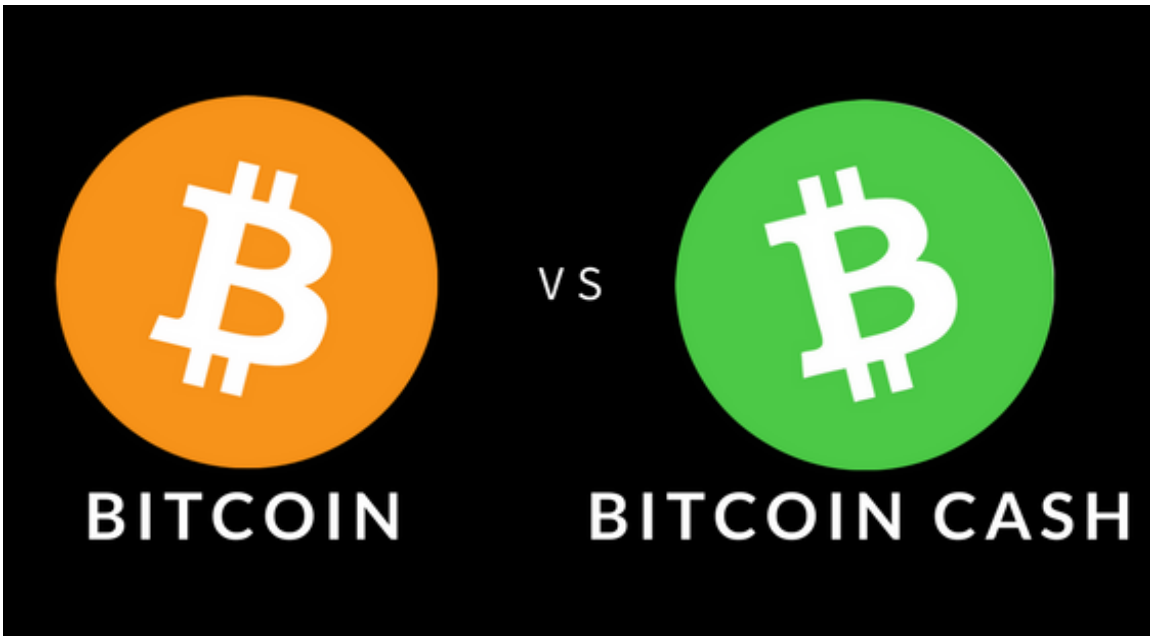
Giảng viên	***ThS Ngô Phú Thanh***
Môn	***Ứng dụng Python trong tài chính***
Sinh viên	***Lâm Nhựt Thịnh***
MSSV	***K194141751***

MỤC LỤC

PHẦN 1: TỔNG QUAN ĐỀ TÀI NGHIÊN CỨU

1. Giới thiệu đề tài nghiên cứu:

Bitcoin cash coin (BCH) là một đồng tiền điện tử được xem là "người anh em" của **Bitcoin (BTC)**. BCH là phiên bản sau phân tách (hard fork) từ nền tảng blockchain của Bitcoin, có hiệu lực từ ngày 1 tháng 8 năm 2017, dù vẫn dựa trên nền tảng Blockchain của Bitcoin nhưng đã được tăng kích thước khối **(Blocksize)** từ 1MB lên 8MB, nhằm giúp mở rộng quy mô công nghệ cơ bản của Bitcoin, cho phép nhiều giao dịch được thực hiện hơn với chi phí thấp hơn. BCH cũng có sự phát triển khá mạnh mẽ trong thời gian đầu sau khi được ra mắt (vượt qua giá của đồng Ethereum), sau này thì có dấu hiệu hạ nhiệt tuy nhiên vẫn nằm trong top 100 những đồng coin có market cap lớn nhất (hơn 7 tỷ đô tính đến đầu năm 2022, theo coinmarketcap.com).



BCH là một coin khá đặc biệt phục vụ cho tham vọng củng cố, lan rộng tầm ảnh hưởng của Bitcoin. Nhận thấy đây là một đồng tiền khá thú vị, tác giả quyết định thực hiện đề tài nghiên cứu này. Ngoài ra, sự biến động lớn của giá các đồng tiền điện tử (Cryptocurrencies) nói chung và BCH nói riêng cũng là một lý do khiến tác giả lựa chọn giá của loại tài sản này để nghiên cứu.

2. Mục tiêu nghiên cứu:

Đề tài nghiên cứu của tác giả hướng đến việc dự báo giá đóng cửa của BCH (hoặc xu hướng thay đổi giá) thông qua các mô hình ước lượng. Ngoài ra đề

tài còn hướng đến việc tìm ra các phương pháp để việc dự báo kết quả có sai số thấp nhất có thể cũng như xác định mô hình phù hợp nhất với bộ dữ liệu.

3. Đối tượng và phạm vi nghiên cứu:

3.1 Đối tượng nghiên cứu:

Giá đóng cửa theo ngày của Bitcoin Cash Coin.

3.2 Phạm vi nghiên cứu:

- *Phạm vi dữ liệu:* Dữ liệu giá của BCH trong thời gian từ ngày 01/01/2021 đến 01/01/2022. (tính cập nhật của dữ liệu được đảm bảo).
- *Phạm vi thực hiện nghiên cứu:* Từ ngày 24/12/2021 đến ngày 25/01/2022.

4. Phương pháp nghiên cứu:

Phương pháp nghiên cứu chính của đề tài là định lượng. tác giả lựa chọn 2 phương pháp dự báo là mô phỏng Monte Carlo và mô hình ARIMA - 2 phương pháp phổ biến thường được sử dụng để xử lý dữ liệu chuỗi thời gian. Ngoài ra đề tài cũng áp dụng một vài phương pháp khác như trực quan hóa, thống kê, …

PHẦN 2: DỮ LIỆU

1. Thu thập dữ liệu:

Dữ liệu được sử dụng là dữ liệu về giá của đồng Bitcoin Cash từ ngày 01/01/2021 đến ngày 01/01/2022 (bao gồm ngày giao dịch, giá mở cửa, giá đóng cửa, giá cao nhất, giá thấp nhất, khối lượng giao dịch, số lượng giao dịch). Dữ liệu được thu thập từ trang web coinapi.io thông qua API miễn phí của trang web này (hình minh họa bên dưới).

```
import requests
symbol_id = 'BITSTAMP_SPOT_BCH_USD'
period_id = '1DAY'
time_start='2021-01-01'
time_end="2022-01-01"
limit='10000'
symbol_name='BCH'
headers = {'X-CoinAPI-Key' : '43CBEEAE-257B-4F4B-82A5-DEC72E93A63E'}
response = requests.get(
f'https://rest.coinapi.io/v1/ohlcv/{symbol_id}/history?period_id={period_id}&time_start={time_start}
&time_end={time_end}&limit={limit}',
headers=headers)
```

Sau khi lấy được dữ liệu thông qua API từ trang web về, tác giả sẽ tiến hành chuyển đổi dữ liệu đó thành dạng DataFrame sử dụng gói Pandas.

```
from pandas import json_normalize
dictr = response.json()
df = json_normalize(dictr)
```

Tiến hành đọc dữ liệu chúng ta sẽ được bộ dữ liệu hoàn chỉnh như sau:

time_period_start	time_period_end	time_open	time_close	price_open	price_high	price_low	price_close	volume_traded	trades_count
2021-01-00:00:00.0000000Z	2021-01-02T00:00:00.0000000Z	2021-01-01T00:03:12.6150000Z	2021-01-01T23:58:27.4330000Z	342.74	355.93	332.00	342.50	7899.661575	1827
2021-01-00:00:00.0000000Z	2021-01-03T00:00:00.0000000Z	2021-01-02T00:00:07.1560000Z	2021-01-02T23:59:40.0290000Z	342.86	368.84	334.39	354.45	19312.140340	3396
2021-01-00:00:00.0000000Z	2021-01-04T00:00:00.0000000Z	2021-01-03T00:00:47.2290000Z	2021-01-03T23:59:45.4690000Z	354.90	434.54	354.65	423.71	38108.922182	7657
2021-01-00:00:00.0000000Z	2021-01-05T00:00:00.0000000Z	2021-01-04T00:00:20.7520000Z	2021-01-04T23:59:20.3090000Z	425.20	467.67	377.00	406.28	46750.940584	9969
2021-01-00:00:00.0000000Z	2021-01-06T00:00:00.0000000Z	2021-01-05T00:01:23.6910000Z	2021-01-05T23:59:22.2090000Z	409.08	426.22	380.21	420.05	26111.978197	4527
...	...	...	...	...	...	...	...	...	...
2021-12-00:00:00.0000000Z	2021-12-28T00:00:00.0000000Z	2021-12-27T00:03:33.9000000Z	2021-12-27T23:58:47.5850000Z	452.65	475.88	448.07	465.34	5278.753228	2122
2021-12-00:00:00.0000000Z	2021-12-29T00:00:00.0000000Z	2021-12-28T00:00:03.3790000Z	2021-12-28T23:58:32.5210000Z	465.35	465.41	438.25	438.89	3727.952690	1939
2021-12-00:00:00.0000000Z	2021-12-30T00:00:00.0000000Z	2021-12-29T00:00:17.3020000Z	2021-12-29T23:58:49.2900000Z	438.96	445.51	426.62	429.78	4011.802759	1622
2021-12-00:00:00.0000000Z	2021-12-31T00:00:00.0000000Z	2021-12-30T00:03:20.9110000Z	2021-12-30T23:59:43.5430000Z	428.98	437.32	423.83	431.87	3639.521695	1546
2021-12-00:00:00.0000000Z	2022-01-01T00:00:00.0000000Z	2021-12-31T00:00:54.3480000Z	2021-12-31T23:58:07.3570000Z	431.36	437.17	419.85	430.43	6098.571487	2123

2. Xử lý dữ liệu:

Đầu tiên tác giả sẽ kiểm tra thông tin tổng thể về bộ dữ liệu như kiểu dữ liệu của các cột, số lượng dữ liệu trống (missing value).

```
: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         365 non-null    int64  
 1   time_period_start  365 non-null    object 
 2   time_period_end    365 non-null    object 
 3   time_open          365 non-null    object 
 4   time_close         365 non-null    object 
 5   price_open         365 non-null    float64
 6   price_high         365 non-null    float64
 7   price_low          365 non-null    float64
 8   price_close        365 non-null    float64
 9   volume_traded      365 non-null    float64
 10  trades_count       365 non-null    int64  
dtypes: float64(5), int64(2), object(4)
memory usage: 31.5+ KB
```

```
: df.isnull().sum().sum()
```

```
: 0
```

Như vậy bộ dữ liệu khá sạch, không có dữ liệu trống nào.
Sau đó chúng ta sẽ chỉnh lại format của dữ liệu thời gian giao dịch và bỏ đi những cột không cần thiết.

```
8]: df['time_period_start']=df['time_period_start'].apply(lambda x: x.split('T')[0])
    df=df.drop(['time_period_end','time_open','time_close','volume_traded','trades_count'
                ,'price_high','price_low','price_open'],axis=1)
    df=df.rename(columns={'time_period_start':'Date','price_close':'Close'})
    df
```

8]:

Dữ liệu về ngày giao dịch sẽ được điều chỉnh về dạng "yyyy/mm/dd", 2 cột dữ liệu được giữ lại để sử dụng là ngày giao dịch và giá đóng cửa. Các biến giữ lại cũng sẽ được đổi tên lại. Bộ dữ liệu hoàn chỉnh như sau:

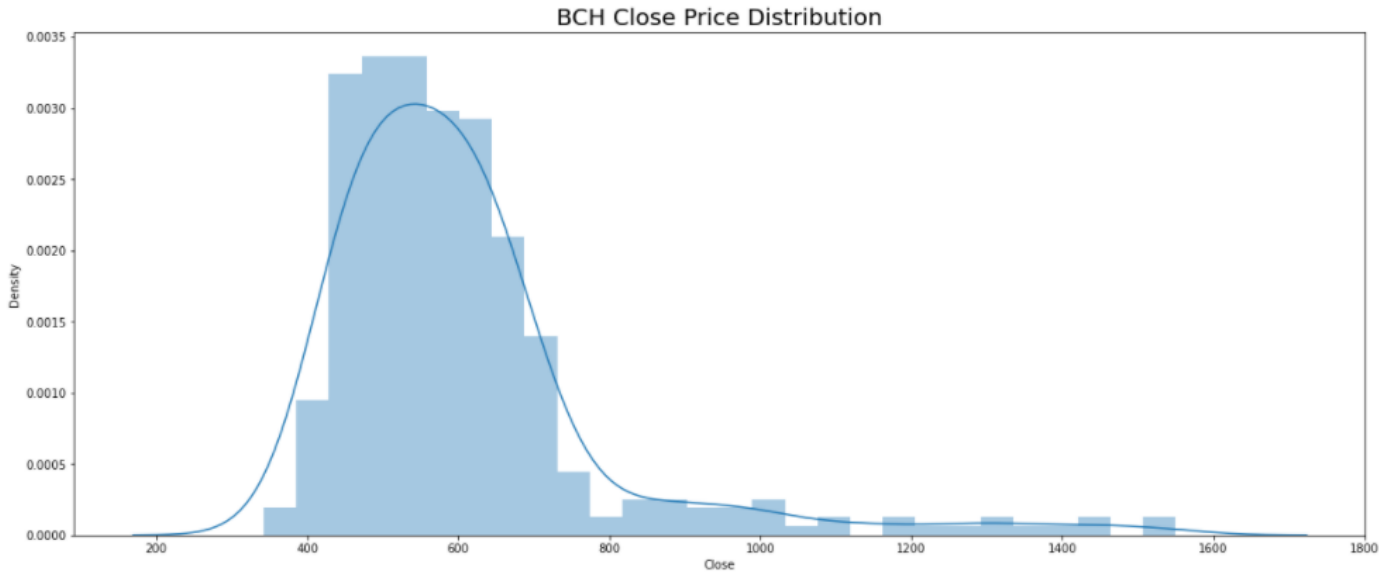
:

	Date	Close
0	2021-01-01	342.50
1	2021-01-02	354.45
2	2021-01-03	423.71
3	2021-01-04	406.28
4	2021-01-05	420.05
...	...	...
360	2021-12-27	465.34
361	2021-12-28	438.89
362	2021-12-29	429.78
363	2021-12-30	431.87
364	2021-12-31	430.43

365 rows × 2 columns

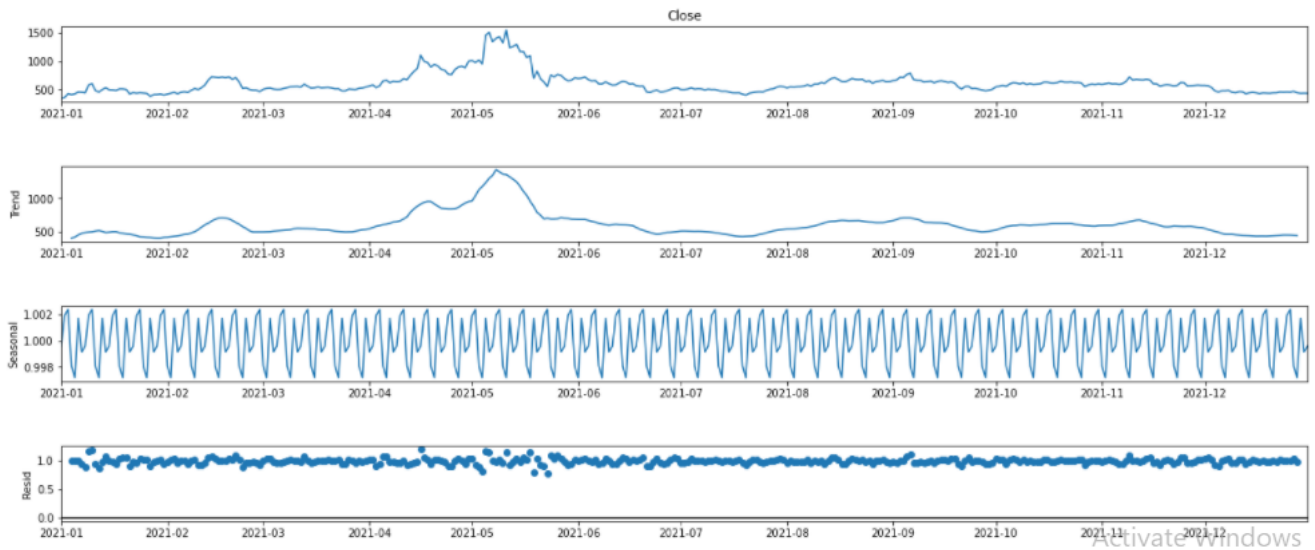
3. Một số nhìn nhận ban đầu về dữ liệu:

Hình ảnh trên là hình ảnh biểu đồ về giá đóng cửa của BCH được vẽ ra từ dữ liệu đã thu thập được. Từ hình chúng ta có thể thấy rằng giá đóng cửa của BCH trong năm 2021 biến động rất mạnh, giá cao nhất là khoảng giai đoạn tháng 05 với khoảng gần 1600$, giá thấp nhất khoảng dưới 400$. Giai đoạn tháng 05, tháng 06 cũng là giai đoạn chứng kiến sự rớt giá rất mạnh từ hơn 1400$ xuống dưới 600$. Tiếp theo chúng ta xem đến phân phối của dữ liệu:



Có thể nhận thấy rằng dữ liệu hơi lệch sang bên trái một chút vì có khá nhiều outliers ở phía bên phải của dữ liệu (chính là những lúc giá tăng cao đột biến), điều này cũng minh chứng cho sự biến động lớn trong giá đóng cửa của BCH.

KẾT QUẢ PHÂN RÃ CHUỖI DỮ LIỆU



<Figure size 1440x576 with 0 Axes>

Sau khi phân rã dữ liệu, nhận thấy trend của dữ liệu có xu hướng tăng đến tháng 05 năm 2021 sau đó giảm dần, về tổng thể chung thì không có trend tăng hay giảm quá rõ ràng. Tính mùa vụ của bộ dữ liệu diễn ra trong một chu kỳ rất ngắn, chỉ khoảng 1 tuần.

4. Phân chia tập dữ liệu:

Ở bài nghiên cứu này, bộ dữ liệu sẽ được chia thành 2 tập train - test để phục vụ cho việc dự báo và ước lượng sai lệch. Tỷ lệ chia là 80% dữ liệu dành cho tập train và 20% còn lại dành cho tập test.

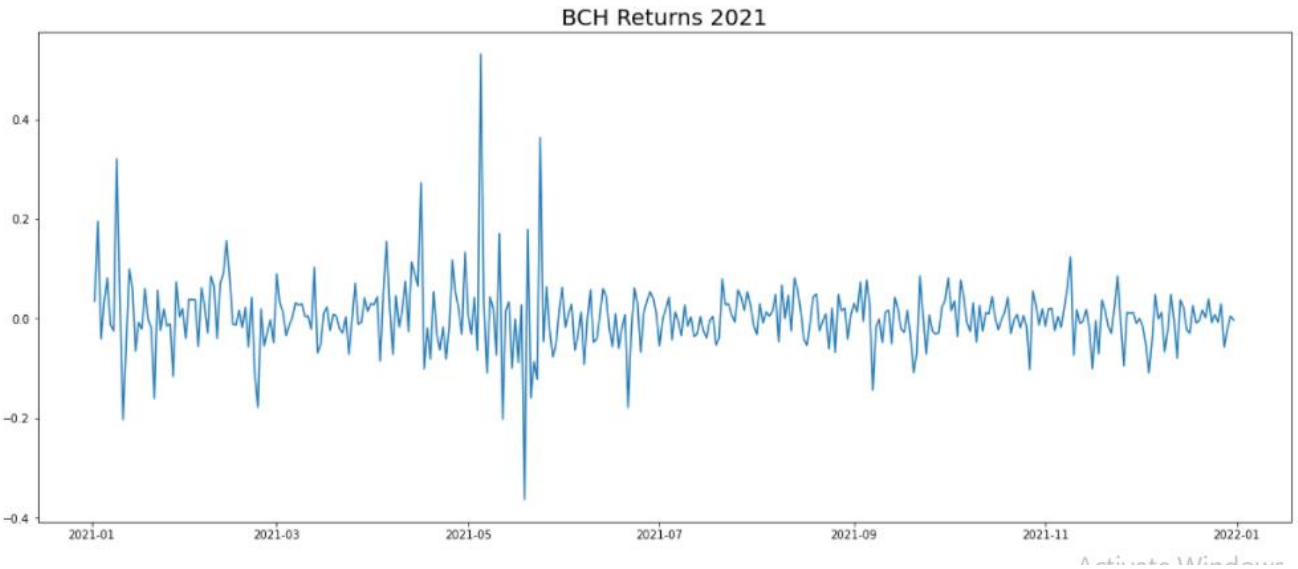
```
: #Đặt mốc để chia bộ dữ liệu ra tập train-test
splitting_landmark=int(len(df)*0.8)
```

Một mốc chia dữ liệu sẽ được đặt ra từ trước để quá trình chia dữ liệu trong suốt bài nghiên cứu trở nên thuận tiện hơn.

PHẦN 3: XÂY DỰNG MÔ HÌNH VÀ DIỄN GIẢI KẾT QUẢ

1. Phương pháp mô phỏng Monte Carlo:

Đầu tiên tác giả tiến hành tính suất sinh lời của BCH sử dụng hàm pct_change() và sau đó vẽ biểu đồ thể hiện suất sinh lời đó:



Cũng giống như biểu đồ giá, suất sinh lợi của BCH ở giai đoạn tháng 05 năm 2021 cũng có những biến động rất mạnh.

Sau đó tập dữ liệu suất sinh lợi sẽ được chia thành 2 tập train - test theo mốc chia đã được quy định từ trước.

```
]: train_mc=returns.iloc[:splitting_landmark]
   test_mc=returns.iloc[splitting_landmark:]
```

Tiếp theo, tác giả sẽ tính toán các thông số cho phương trình mô phỏng:

```
]: # Tính toán các thông số
   # T là độ dài giai đoạn mô phỏng
   # N là số điểm dữ liệu trong giai đoạn mô phỏng
   # s0 là giá trị khởi điểm cho giai đoạn mô phỏng
   # n_sim là số lần mô phỏng
   # mu là giá trị trung bình trong giai đoạn đánh giá
   # sigma là độ lệch chuẩn trong giai đoạn đánh giá
   T=len(test_mc)
   N=len(test_mc)
   s0=df['Close'][train_mc.index[-1]]
   n_sim=1000
   mu=train_mc.mean()
   sigma=train_mc.std()
```

Sau đó chúng ta sẽ tiến hành tạo hàm mô phỏng theo công thức SDE:

```
7]: # Thực hiện tạo hàm mô phỏng theo công thức SDE
    def simulate_gbm(s0,mu,sigma,n_sim,T,N,random_seed=1):
        np.random.seed(1)
        # Xác định số điểm dữ liệu mô phỏng
        dt=T/N
        # Xác định tham số ngẫu nhiên Brownian Motion
        dW=np.random.normal(scale=np.sqrt(dt),size=(n_sim,N))
        # Tạo ma trận để lưu giữ kết quả
        W=np.cumsum(dW,axis=1)
        time_step=np.linspace(dt,T,N)
        time_steps=np.broadcast_to(time_step,(n_sim,N))
        # Thiết lập công thức SDE
        s_t=s0*np.exp((mu-0.5*sigma**2)*time_steps+sigma*W)
        # Chèn giá trị khởi điểm vào đầu kết quả mô phỏng
        s_t=np.insert(s_t,0,s0,axis=1)
        return s_t
```

```
8]: # Thực hiện mô phỏng theo hàm đã tạo
    gbm_sim=simulate_gbm(s0,mu,sigma,n_sim,T,N)
```

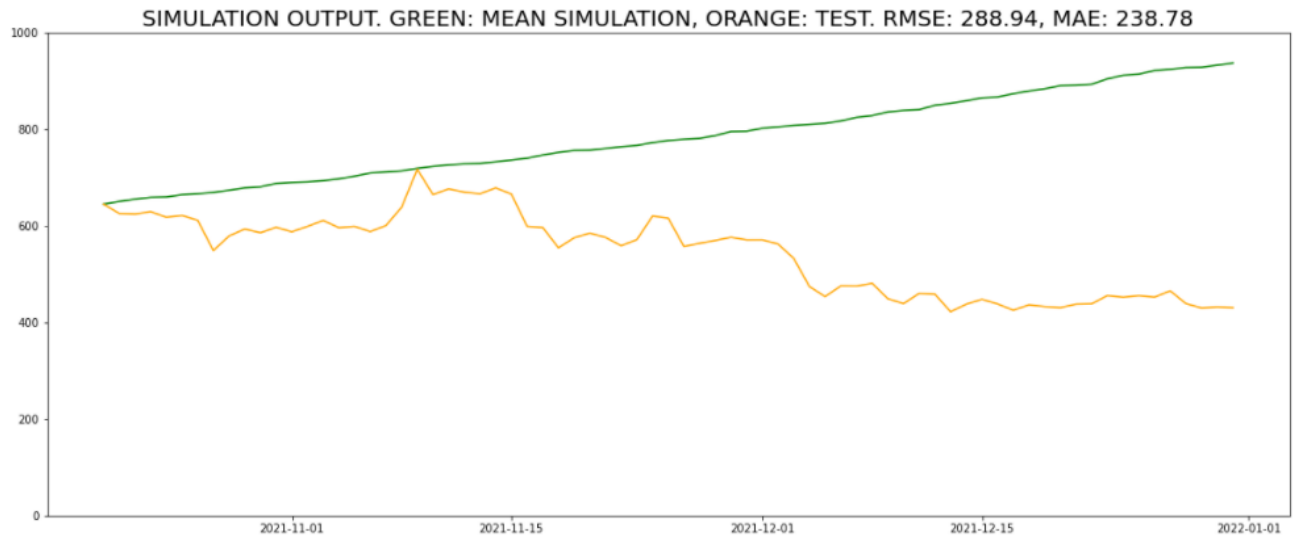
Chúng ta sẽ vẽ biểu đồ để thể hiện kết quả mô phỏng, bắt đầu với việc xác định các mốc thời gian trên biểu đồ:

```
: # Xác định các mốc thời gian trên biểu đồ
  last_train_date_mc=train_mc.index[-1]
  first_test_date_mc=test_mc.index[0]
  last_test_date_mc=test_mc.index[-1]
  indices=df['Close'][last_train_date_mc:last_test_date_mc].index
  index=[date.date() for date in indices]
  gbm_sim_df=pd.DataFrame(np.transpose(gbm_sim),index=index)
```

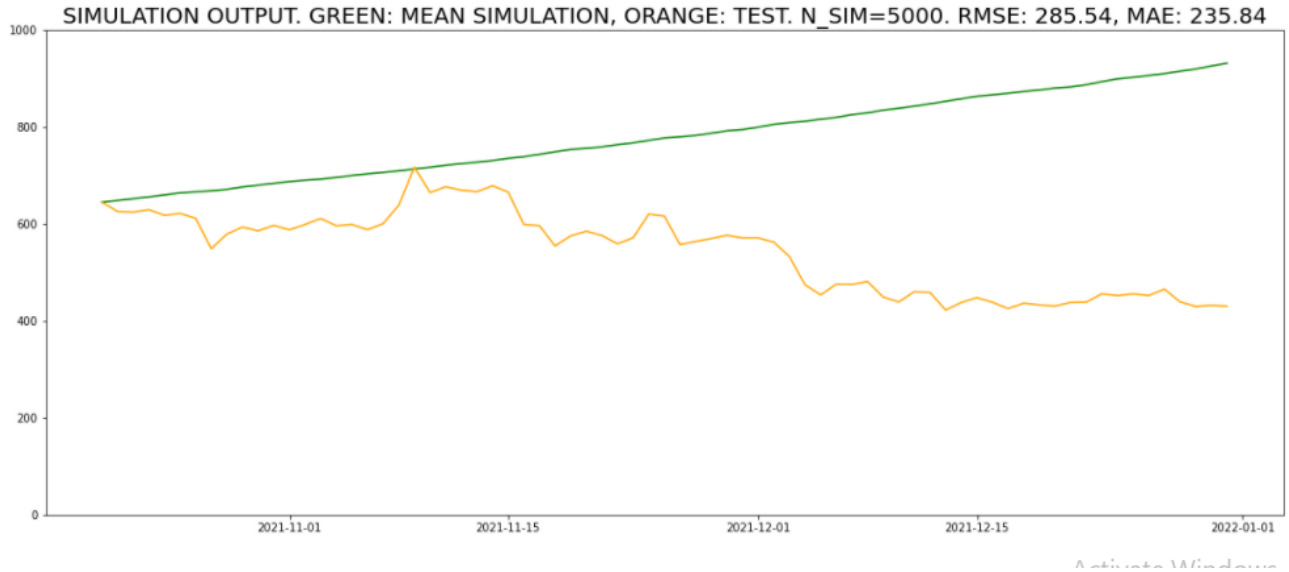
Cuối cùng chúng ta tiến hành vẽ biểu đồ và tính toán sai lệch:

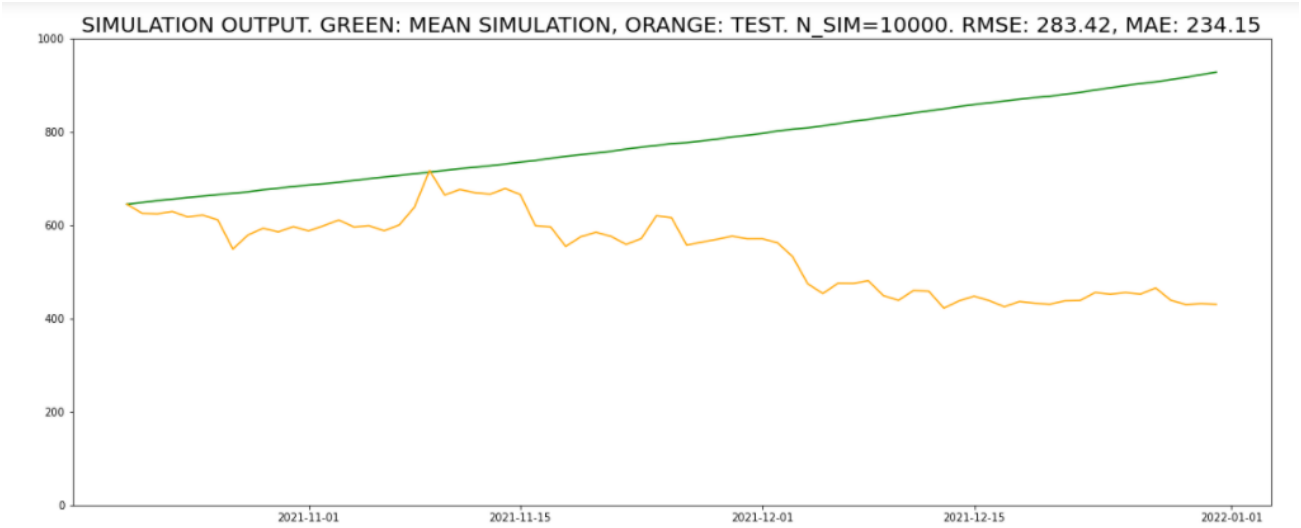
```
: ax=gbm_sim_df.plot(alpha=0.2,legend=False,figsize=(20,8),color='white')
  line_1=ax.plot(index,gbm_sim_df.mean(axis=1),color='green')
  line_2=ax.plot(index,df['Close'][last_train_date_mc:last_test_date_mc],color='orange')
  rmse = np.sqrt(mean_squared_error(df['Close'][last_train_date_mc:last_test_date_mc], gbm_sim_df.mean(axis=1)))
  mae=mean_absolute_error(df['Close'][last_train_date_mc:last_test_date_mc], gbm_sim_df.mean(axis=1))
  ax.set_title(f'Simulation Output. Green: Mean Simulation, Orange: Test. N_SIM={n_sim}. RMSE: {rmse:.2f}, MAE: {mae:.2f}'.upper()
               ,fontsize=20)
  ax.set_ylim(0,1000)
  plt.show()
```

Set color = ‘white’ để ẩn đi các đường mô phỏng để có thể nhìn rõ hơn kết quả mô phỏng trung bình và giá trị thực tế. Bên cạnh đó, RMSE và MAE cũng được kết hợp tính toán để ước lượng sai lệch.



Đường màu xanh lá là giá trị mô phỏng trung bình, đường màu cam là giá trị thực tế của tập test. Chúng ta có thể nhận thấy rằng 2 đường này khá lệch nhau, tức là kết quả dự báo không đạt được sự chính xác cao. Chỉ số RMSE = 288.94 và MAE = 238.78 là rất lớn so với giá đóng cửa của một BCH coin trong tập test chỉ dao động trong khoảng từ 400$ đến hơn 600$, tác giả cũng đã thử tăng số lần mô phỏng lên và thu được 1 số kết quả như sau:





Có thể nhận thấy rằng RMSE và MAE cũng không thể cải thiện được quá nhiều. Phương pháp mô phỏng Monte Carlo không cho chúng ta những kết quả khả quan với tập dữ liệu này. Chúng ta sẽ đến với mô hình ARIMA.

2. Mô hình ARIMA:

Đầu tiên, chúng ta cũng tiến hành chia bộ dữ liệu ra thành hai tập train - test:

```
l]: # Chia tập train - test
    train_arima=df.iloc[:splitting_landmark]['Close']
    test_arima=df.iloc[splitting_landmark:]['Close']
    print(f'Train: {len(train_arima)}, Test: {len(test_arima)}')
```

```
Train: 292, Test: 73
```

Tiếp theo chúng ta sẽ kiểm tra tính dừng của tập train, kiểm định ADF có 2 giả thuyết kiểm định như sau:
H0: Chuỗi dữ liệu không dừng.
H1: Chuỗi dữ liệu dừng.

```
2]: #Kiểm tra tính dừng của dữ liệu train
result = adfuller(train_arima)
print('ADF test for train dataset')
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF test for train dataset
ADF Statistic: -1.965549
p-value: 0.301880
Critical Values:
        1%: -3.454
        5%: -2.872
        10%: -2.572
```

Có thể nhận thấy p_value đạt giá trị 0.30 lớn hơn 3 mức ý nghĩa là 1%, 5% và 10%

→ Không thể bác bỏ giả thuyết H0.

→ Chuỗi dữ liệu không dừng.

Chúng ta sẽ tiếp tục kiểm định tính dừng cho sai phân bậc 1 của train:

```
]: # Kiểm tra tính dừng của sai phân bậc 1 của train
train_diff_1 = train_arima.diff()
train_diff_1.dropna(axis=0, inplace=True)
result = adfuller(train_diff_1)
print('ADF test for train_diff_1 dataset')
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))
```

```
ADF test for train_diff_1 dataset
ADF Statistic: -6.087983
p-value: 0.000000
Critical Values:
        1%: -3.454
        5%: -2.872
        10%: -2.572
```

Có thể nhận thấy p_value đạt giá trị 0 nhỏ hơn 3 mức ý nghĩa là 1%, 5% và 10%

→ Bác bỏ giả thuyết H0.

→ Chuỗi dữ liệu dừng.

→ d=1

Sau đó chúng ta sẽ sử dụng Auto-Arima để tìm ra mô hình tốt nhất (Auto-Arima sẽ cho ra kết quả nhanh hơn nên tác giả lựa chọn).

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=3215.269, Time=0.93 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=3224.019, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=3214.095, Time=0.20 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=3214.844, Time=0.20 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=3222.089, Time=0.03 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=3216.094, Time=0.27 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=3216.095, Time=0.27 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=3216.776, Time=0.63 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=3212.198, Time=0.08 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=3214.196, Time=0.12 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=3214.197, Time=0.14 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=3212.949, Time=0.10 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=3214.860, Time=0.30 sec

Best model:  ARIMA(1,1,0)(0,0,0)[0]
Total fit time: 3.346 seconds
```

→ Mô hình tốt nhất là ARIMA(1,1,0).

Kết quả của các hệ số ước lượng trong mô hình:

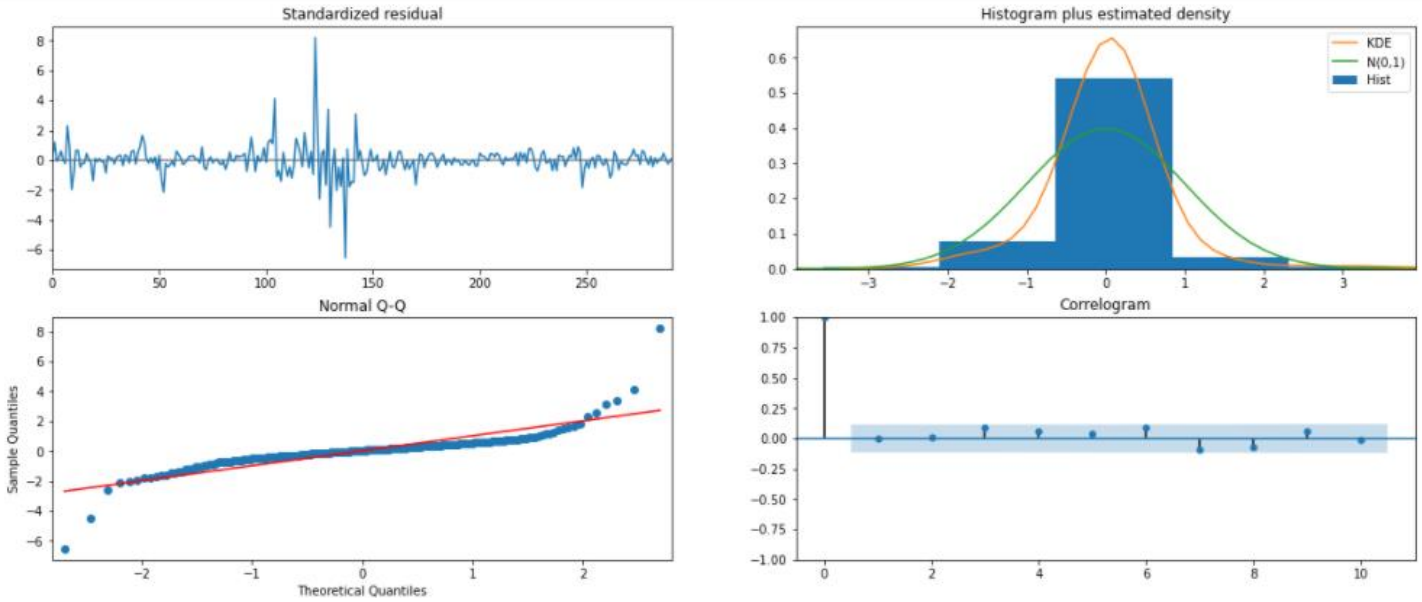
```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  292
Model:               SARIMAX(1, 1, 0)   Log Likelihood               -1604.099
Date:                Tue, 25 Jan 2022   AIC                           3212.198
Time:                        22:02:08   BIC                           3219.544
Sample:                             0   HQIC                          3215.141
                                  - 292
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.1998      0.033     -6.013      0.000      -0.265      -0.135
sigma2      3592.8423     90.218     39.824      0.000    3416.018    3769.667
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):              6418.34
Prob(Q):                              0.98   Prob(JB):                         0.00
Heteroskedasticity (H):               0.49   Skew:                             0.96
Prob(H) (two-sided):                  0.00   Kurtosis:                        25.93
===================================================================================

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

p_value của tất cả các hệ số ước lượng đều bằng 0

→ Tất cả các hệ số đều có ý nghĩa thống kê.

Chúng ta sẽ kiểm tra đồ thị phân phối phần dư của mô hình:



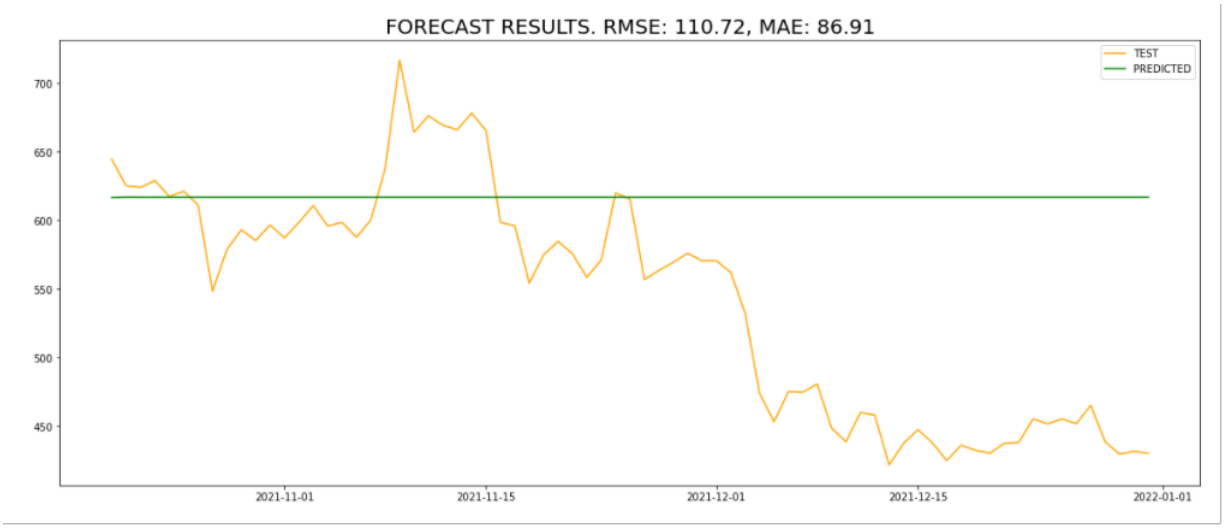
Từ distribution plot và QQ plot chúng ta có thể nhận thấy là phần dư của mô hình có tuân theo phân phối chuẩn.

```
# Kiểm định tự tương quan của phần dư
sm.stats.acorr_ljungbox(best_model.resid(),  return_df=True)
```

	lb_stat	lb_pvalue
1	0.001180	0.972601
2	0.316056	0.853826
3	2.103959	0.551112
4	3.072422	0.545780
5	3.849419	0.571293
6	5.873518	0.437507
7	8.157551	0.318907
8	8.361102	0.399015
9	9.950243	0.354528
10	10.437000	0.403025

Nhìn vào kết quả kiểm định trên, chúng ta có thể khẳng định phần dư của mô hình không xảy ra hiện tượng tự tương quan (do tất cả các lag đều không có ý nghĩa thống kê).

Tiếp theo chúng ta sẽ vẽ biểu đồ so sánh Test và Prediction:



Sai số RMSE và MAE đã có cải thiện so với mô phỏng Monte Carlo nhưng nhìn chung vẫn còn chênh lệch nhiều và đường thẳng predicted (đường màu xanh lá) không dự đoán được các xu hướng thay đổi giá. Mô hình ARIMA đã thể hiện sự hiệu quả cao hơn so với Monte Carlo, chúng ta sẽ tìm cách để tối ưu kết quả dự đoán của mô hình này.

3. Cải thiện kết quả dự báo cho mô hình ARIMA:

3.1 Seven-steps Forecast:

Với phương pháp này, chúng ta sẽ chia nhỏ để dự đoán từng giai đoạn của tập test, ý tưởng này có thể diễn giải như sau:

- Đầu tiên, chúng ta sẽ dự báo 7 ngày tiếp theo (n_periods=7), tức là 7 ngày đầu tiên của tập test.
- Sau đó chúng ta sẽ thêm 7 giá trị tương ứng của tập test của 7 ngày đó vào tập train và tiếp tục chạy mô hình ARIMA để dự đoán 7 ngày tiếp theo nữa.
- Quá trình lặp lại cho đến khi hết tập dữ liệu test.

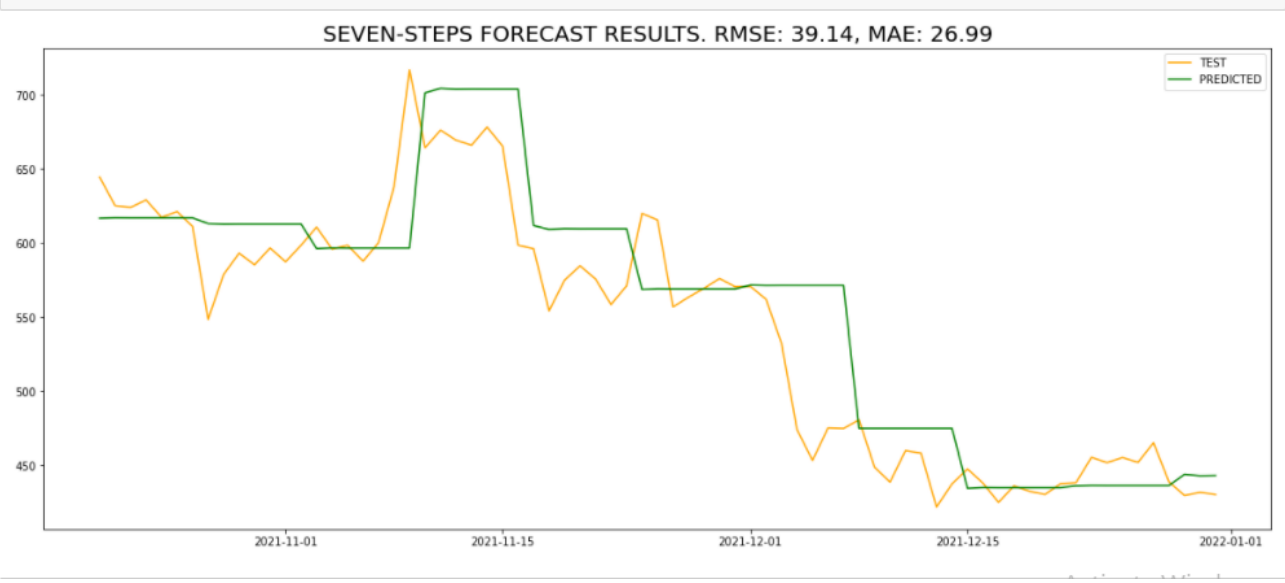
Chúng ta sẽ xây dựng hàm phục vụ cho cách dự báo này:

```
def seven_steps_forecast(DF):
    test_size=len(test_arima)
    prediction_7=[]
    while test_size>0:
        train=DF['Close'].iloc[:-test_size]
        test=DF['Close'].iloc[-test_size:]
        if test_size>=7:
            model=pmdarima.auto_arima(train,
                                      trace=1,
                                      test='adf',
```

```
                    error_action ='ignore',
                    suppress_warnings = True,
                    seasonal = False,
                    stepwise = True,
                    approximation = False,
                    n_jobs = -1,
                    seasonal_test = None)
        predicted=model.predict(n_periods=7).tolist()
        for i in predicted:
            prediction_7.append(i)
        test_size-=7
    else:
        model=pmdarima.auto_arima(train,
                    trace=1,
                    test='adf',
                    error_action ='ignore',
                    suppress_warnings = True,
                    seasonal = False,
                    stepwise = True,
                    approximation = False,
                    n_jobs = -1,
                    seasonal_test = None)
        predicted=model.predict(n_periods=test_size).tolist()
        for i in predicted:
            prediction_7.append(i)
        test_size=0
  return prediction_7
```

(Chụp không thấy hết được phần code nên tác giả copy vào đây)

Tiếp theo chúng ta sẽ vẽ biểu đồ so sánh Test và Prediction:



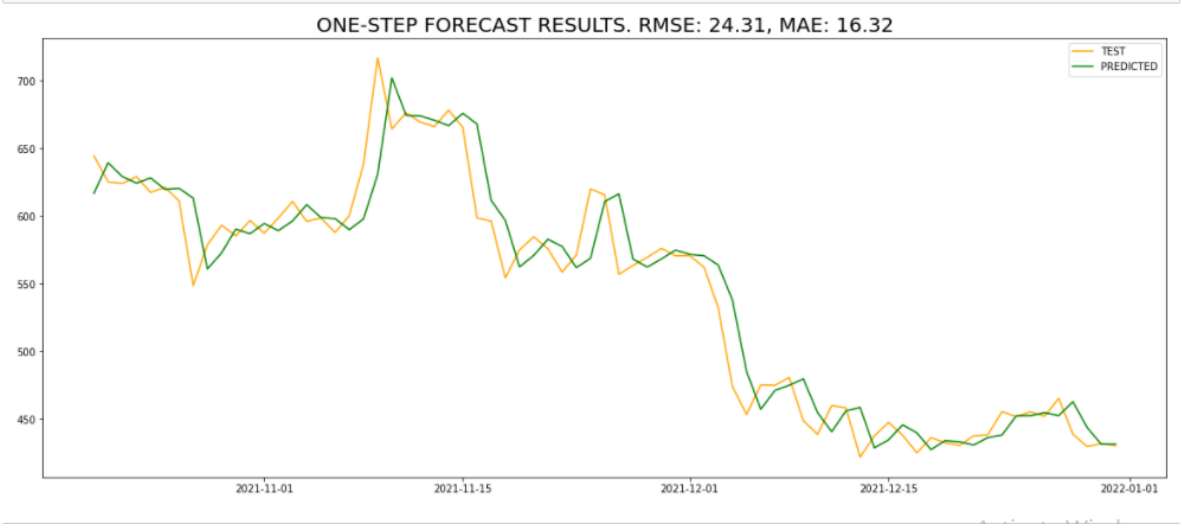
Trong trường hợp này RMSE và MAE có sự cải thiện rất rõ rệt khi chỉ còn lần lượt là 39.14 và 26.99 và bên cạnh đó đường Predicted (đường màu xanh lá cây) đã có thể bám theo được những xu hướng biến động giá của tập test.

3.2 One-step Forecast:

Tương tự như Seven-steps Forecast, phương pháp này sẽ chia nhỏ giai đoạn dự báo ra chỉ còn 1 ngày (n_periods=1)
Hàm được xây dựng như sau:

```
def one_step_forecast():
    predicted = best_model.predict(n_periods=1)
    return predicted.tolist()[0]
one_step_prediction = []
for x in test_arima:
    predicted = one_step_forecast()
    one_step_prediction.append(predicted)
    # Updates the existing model
    best_model.update(x)
```

Tiếp theo chúng ta sẽ vẽ biểu đồ so sánh Test và Prediction:



Sai lệch RMSE và MAE tiếp tục được cải thiện so với các phương pháp trước, đường Predicted cũng đã bám sát giá trị thực tế hơn.

PHẦN 4: KẾT LUẬN

Qua bài nghiên cứu, tác giả rút ra một số kết luận như sau:

- Giá của tiền điện tử nói chung và Bitcoin Cash Coin nói riêng có sự biến động rất lớn, gây khó khăn cho các mô hình tài chính truyền thống điển hình như ARIMA hay Monte Carlo để dự báo. Đặc biệt là Monte Carlo khi mô hình này cho ra kết quả không mấy khả quan.
- Việc dự báo trong ngắn hạn có thể giảm thiểu những sai số trong quá trình dự báo. Tuy nhiên dự báo dài hạn vẫn là một bài toán đau đầu cần giải quyết.
- Với những dữ liệu phức tạp và biến động khó lường như giá các đồng tiền điện tử, cần thiết những mô hình nâng cao của Machine Learning hay Deep Learning để xử lý và cho ra kết quả dự báo tốt hơn.

TÀI LIỆU 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. ARIMA documentation (statsmodels).
3. Time Series Tutorial.
4. Github.
5.