ĐẠI HỌC BÁCH KHOA HÀ NỘI
KHOA TOÁN-TIN

—————o0o—————

HỆ HỖ TRỢ QUYẾT ĐỊNH
BÁO CÁO CUỐI KÌ

Xây Dựng Các Mô Hình Cho Bài Toán Dự Báo Giá Tiền Ảo Cryptocurrencies

Giảng viên hướng dẫn: TS. Trần Ngọc Thăng
Bộ môn: TOÁN - TIN
Sinh viên thực hiện: Đồng Văn Sỹ Hoàng
MSSV: 20227231
Mã Lớp: 158242

Hà Nội, 6/2025

Lời mở đầu

Trong những năm gần đây, sự phát triển mạnh mẽ của công nghệ blockchain đã kéo theo sự bùng nổ của thị trường tiền điện tử (cryptocurrencies). Với đặc điểm phi tập trung, minh bạch và tính thanh khoản cao, các loại tiền ảo như Bitcoin, Ethereum, v.v. đã thu hút sự quan tâm lớn từ các nhà đầu tư, các tổ chức tài chính, cũng như giới nghiên cứu khoa học dữ liệu. Tuy nhiên, thị trường tiền ảo luôn biến động mạnh và chịu ảnh hưởng bởi nhiều yếu tố, khiến việc dự đoán giá trở nên phức tạp và đầy thách thức.

Trước thực tế đó, việc áp dụng các mô hình phân tích dữ liệu và học máy để xây dựng công cụ dự đoán giá tiền ảo là một hướng đi tiềm năng. Trong khuôn khổ môn học Hệ Hỗ Trợ Quyết Định, em thực hiện đề tài "Xây Dựng Các Mô Hình Cho Bài Toán Dự Báo Giá Tiền Ảo Cryptocurrencies" nhằm tìm hiểu và ứng dụng những mô hình thống kê cơ bản nhưng hiệu quả – hồi quy tuyến tính,hồi quy phi tuyến, mạng nơ ron – để phân tích dữ liệu và dự báo xu hướng giá của một số đồng tiền điện tử phổ biến.

Báo cáo trình bày quá trình xây dựng mô hình từ khâu thu thập dữ liệu, xử lý, phân tích, huấn luyện mô hình đến đánh giá kết quả. Qua đó, em không chỉ rèn luyện kỹ năng ứng dụng lý thuyết vào thực tiễn mà còn nhận diện được những hạn chế và tiềm năng mở rộng của phương pháp tiếp cận này.

Báo cáo này sẽ được trình bày gồm 4 chương chính như sau:

- **Chương 1: Tiền xử lý dữ liệu**
  Trình bày bài toán nghiên cứu, thu thập và xử lý dữ liệu đầu vào nhằm chuẩn bị cho quá trình xây dựng mô hình dự đoán.

- **Chương 2: Đánh giá mô hình**
  Trình bày các phương pháp đánh giá, phân tích độ chính xác và sai số dự báo.

- **Chương 3: Xây dựng và cải tiến mô hình**
  Trình bày quá trình xây dựng các mô hình phù hợp cho bài toán đặt ra.Đề xuất các hướng cải tiến mô hình nhằm nâng cao chất lượng dự báo, kết hợp thêm các yếu tố và kỹ thuật xử lý dữ liệu nâng cao.

- **Chương 4: Đóng gói và thử nghiệm**
  Trình bày quá trình đóng gói mô hình, thử nghiệm với 1 trường hợp cụ thể

Dù đã rất cố gắng trong quá trình nghiên cứu và soạn thảo báo cáo nhưng vẫn không tránh khỏi những hạn chế không tránh khỏi. Vì vậy, rất mong nhận được sự đóng góp, chỉ bảo và ý kiến của các quý thầy cô.

Bảng 1: BẢNG ĐÁNH GIÁ YÊU CẦU

STT	Loại yêu cầu	Yêu cầu	Điểm chữ	Điểm số	Check	Minh chứng
1	Xử lý dữ liệu (2 điểm)	Mô tả bài toán, đầu vào, đầu ra, yêu cầu xử lý	A	1	X	Trang 9
2		Đánh nhãn & Tiền xử lý dữ liệu	A	1	X	Trang 10
3		Thống kê dữ liệu mẫu	A	1	X	Trang 11
4		Xử lý mất cân bằng dữ liệu (cho bài toán phân lớp) hoặc Chuyển đổi dữ liệu (cho bài toán hồi quy)	A	1	X	Trang 12-15
5	Đánh giá mô hình (1 điểm)	Đề xuất và lựa chọn các tiêu chí đánh giá (về độ chính xác, tốc độ, khả năng ứng dụng,...)	A	1	X	Trang 16
6		Thống kê và phân tích lỗi	A	1	X	Trang 17
7	Cải tiến mô hình (4 điểm)	Kiến trúc mô hình 1	A	1	X	Trang 20-21
8		Kiến trúc mô hình 2	A	1	X	Trang 22-24
9		Kiến trúc mô hình 3	A	1	X	Trang 25-26
10		Kiến trúc mô hình 4	A	1	X	Trang 27-30
11		Kiến trúc mô hình 5	A	1	X	Trang 31-34
12		Kiến trúc mô hình 6	A	1	X	Trang 35-38
13		Kiến trúc mô hình 7	A	1	X	Trang 39-41
14		Kiến trúc mô hình 8	A	1	X	Trang 42-45
15		Có sử dụng mô hình tiên tiến trong 3 năm trở lại đây (chỉ ra paper liên quan)	B	0.75	X	Trang 45-46

Tiếp tục ở trang sau

Bảng 2 – Tiếp theo từ trang trước

STT	Loại yêu cầu	Yêu cầu	Điểm chữ	Điểm số	Check	Minh chứng
16	Đóng gói mô hình (3 điểm)	Có khả năng ứng dụng vào một ngữ cảnh cụ thể (ứng dụng vào bài toán nghiệp vụ nào, ai là người sử dụng)	A	1	X	Trang 48
17		Các chỉ số đánh giá mô hình đủ điều kiện để ứng dụng vào thực tế	A	1	X	Trang 49
18		Đóng gói giao diện demo chương trình	A	1	X	Trang 50
19		Làm slide báo cáo	A	1	X	–
20		Thuyết trình trên lớp	A	1	X	–
Tổng điểm/20				19.75	-	
Tổng điểm/10				10		

Bảng 3: BẢNG MÔ TẢ CHI TIẾT MÔ HÌNH

STT	**Tên mô hình**	**Điều kiện dừng**	**Phương pháp tối ưu siêu tham số**	**Siêu tham số mô hình**	**Kết quả đánh giá (Test Set)**
1	Linear Regression	Mặc định	-	- (Baseline)	RMSE: 1974.59 MAE: 1398.98 R2: 0.9891 MAPE: 1.98%
2	Lasso Regression	Sai số ≤ 0.0001, Số lần lặp = 10000	LassoCV	Alpha = 0.1	RMSE: 1970.59 MAE: 1388.98 R2: 0.9891 MAPE: 1.98%
3	Polynomial Regression	Mặc định	GridSearchCV	Degree (bậc đa thức) = 2	RMSE: 1982.89 MAE: 1407.12 R2: 0.9892 MAPE: 2.01%
4	ARIMA	Mặc định	AutoArima	p (AR) = 5 d (Sai phân) = 1 q (MA) = 0	RMSE: 1971.3 MAE: 1382.27 R2: 0.989 MAPE: 1.98%
5	SVR	Mặc định	GridSearchCV	C (điều chuẩn): 100 Epsilon: 0.01 Gamma: 0.01 Kernel: linear	RMSE: 4050.4 MAE: 2755.55 R2: 0.9549 MAPE: 3.40%
6	Random Forest (RF)	Mặc định	Randomized SearchCV	Số lượng cây: 100 Max features: 7 Depth: 10	RMSE: 3107.3 MAE: 2356.34 R2: 0.9567 MAPE: 3.03%
7	LSTM	Max 100 epochs; Early stopping (val_loss, patience=10)	Keras Tuner	Số lớp LSTM: 1 Units : 96 Dropout: 0.5 Tốc độ học: 1.42e-4 Dense units: 32 Loss rate: 0.2	RMSE: 4015.62 MAE: 3045.34 R2: 0.9537 MAPE: 4.38%
8	GRU	Max 100 epochs; Early stopping (val_loss, patience=10)	Keras Tuner	Time steps: 60 Số lớp GRU: 3 Units/lớp: 64/64/32 Dropout: 0.2 Dense units: 16 Loss rate: 0.2	RMSE: 4704.74 MAE: 3667.24 R2: 0.9365 MAPE: 4.88%

Mục lục

Danh sách bảng

Danh sách hình vẽ

Chương 1

Tiền xử lý dữ liệu

1.1 Tổng quan bài toán

1.1.1 Mô tả bài toán

Bài toán đặt ra là xây dựng các mô hình hồi quy để dự báo giá đóng cửa (**Close**) của tiền ảo dựa trên các đặc trưng liên quan như giá cao nhất, giá thấp nhất, giá mở cửa, khối lượng giao dịch và các chỉ số trong phiên.

1.1.2 Đầu vào

Tập dữ liệu bao gồm 1994 dòng và 12 cột với các biến quan trọng:

- **Name**: Tên của loại tiền ảo.
- **timeOpen**: giờ mở cửa.
- **timeHigh**: giờ giá cao nhất
- **timeLow**: giờ giá thấp nhất.
- **timeClose** : giờ đóng cửa
- **High**: Giá cao nhất trong ngày.
- **Low**: Giá thấp nhất trong ngày.
- **Open**: Giá mở cửa.
- **Close**: Giá đóng cửa (biến mục tiêu).
- **Volume**: Khối lượng giao dịch.
- **Marketcap** : Vốn hóa thị trường.

1.1.3 Đầu ra

Mô hình hồi quy tuyến tính, hồi quy phi tuyến và mạng nơ ron để dự báo giá đóng cửa của tiền ảo trong tương lai dựa trên các biến đầu vào được thu thập trên tập dữ liệu lịch sử, có thể đưa ra chương trình dự báo cho người dùng trong ngữ cảnh cụ thể

1.2 Tiền xử lý dữ liệu

Trước khi đi vào xây dựng mô hình thì chúng ta cần phải đi qua bước tiền xử lý dữ liệu, chuẩn bị dữ liệu cần thiết cho các mô hình. Dữ liệu được thu thập từ Website Coin Market Cap

Ta tiến hành xử lý dữ liệu, các cột thời gian không cần thiết đến định dạng giờ phút nên chúng ta sẽ bỏ đi. Cột thời gian TimeClose là giờ đóng cửa sẽ được đổi tên thành cột Date để ghi nhận giá của ngày đó.

	timeOpen	timeClose	timeHigh	timeLow	name	open	high	low	close	volume	marketCap	timestamp
	2018-12-3	2018-12-3	2018-12-3	2018-12-3	2781	3866.839	3868.743	3725.867	3742.7	4.66E+09	6.53E+10	2018-12-31T23:59:59.999Z
	2018-12-3	2018-12-3	2018-12-3	2018-12-3	2781	3822.385	3901.909	3797.219	3865.953	4.77E+09	6.75E+10	2018-12-30T23:59:59.999Z
	2018-12-2	2018-12-2	2018-12-2	2018-12-2	2781	3932.492	3963.759	3820.409	3820.409	4.99E+09	6.67E+10	2018-12-29T23:59:59.999Z
	2018-12-2	2018-12-2	2018-12-2	2018-12-2	2781	3653.132	3956.136	3642.632	3923.919	5.63E+09	6.85E+10	2018-12-28T23:59:59.999Z
	2018-12-2	2018-12-2	2018-12-2	2018-12-2	2781	3854.689	3874.417	3645.448	3654.834	5.13E+09	6.38E+10	2018-12-27T23:59:59.999Z
	2018-12-2	2018-12-2	2018-12-2	2018-12-2	2781	3819.667	3893.36	3769.864	3857.298	5.33E+09	6.73E+10	2018-12-26T23:59:59.999Z
	2018-12-2	2018-12-2	2018-12-2	2018-12-2	2781	4081.03	4089.562	3760.02	3815.491	6.16E+09	6.66E+10	2018-12-25T23:59:59.999Z
	2018-12-2	2018-12-2	2018-12-2	2018-12-2	2781	4000.332	4271.792	4000.332	4078.599	7.24E+09	7.11E+10	2018-12-24T23:59:59.999Z
0	2018-12-2	2018-12-2	2018-12-2	2018-12-2	2781	4020.995	4085.724	3976.406	3998.98	6.15E+09	6.97E+10	2018-12-23T23:59:59.999Z
1	2018-12-2	2018-12-2	2018-12-2	2018-12-2	2781	3898.084	4014.183	3855.739	4014.183	5.61E+09	7E+10	2018-12-22T23:59:59.999Z
2	2018-12-2	2018-12-2	2018-12-2	2018-12-2	2781	4133.704	4198.43	3850.946	3896.544	7.21E+09	6.79E+10	2018-12-21T23:59:59.999Z
3	2018-12-2	2018-12-2	2018-12-2	2018-12-2	2781	3742.195	4191.229	3728.975	4134.441	8.93E+09	7.21E+10	2018-12-20T23:59:59.999Z
4	2018-12-1	2018-12-1	2018-12-1	2018-12-1	2781	3706.825	3949.323	3687.23	3745.951	6.81E+09	6.53E+10	2018-12-19T23:59:59.999Z
5	2018-12-1	2018-12-1	2018-12-1	2018-12-1	2781	3544.762	3701.349	3487.169	3696.059	5.91E+09	6.44E+10	2018-12-18T23:59:59.999Z

Hình 1.1: Bộ dữ liệu đầu vào trước khi xử lý

timeOpen	Date	timeHigh	timeLow	name	Open	High	Low	Close	Volume	Marketcap	timestamp
12/31/2018	12/31/2018	12/31/2018	12/31/2018	2781	3866.839	3868.743	3725.867	3742.7	4.66E+09	6.53E+10	12/31/2018
12/30/2018	12/30/2018	12/30/2018	12/30/2018	2781	3822.385	3901.909	3797.219	3865.953	4.77E+09	6.75E+10	12/30/2018
12/29/2018	12/29/2018	12/29/2018	12/29/2018	2781	3932.492	3963.759	3820.409	3820.409	4.99E+09	6.67E+10	12/29/2018
12/28/2018	12/28/2018	12/28/2018	12/28/2018	2781	3653.132	3956.136	3642.632	3923.919	5.63E+09	6.85E+10	12/28/2018
12/27/2018	12/27/2018	12/27/2018	12/27/2018	2781	3854.689	3874.417	3645.448	3654.834	5.13E+09	6.38E+10	12/27/2018
12/26/2018	12/26/2018	12/26/2018	12/26/2018	2781	3819.667	3893.36	3769.864	3857.298	5.33E+09	6.73E+10	12/26/2018
12/25/2018	12/25/2018	12/25/2018	12/25/2018	2781	4081.03	4089.562	3760.02	3815.491	6.16E+09	6.66E+10	12/25/2018
12/24/2018	12/24/2018	12/24/2018	12/24/2018	2781	4000.332	4271.792	4000.332	4078.599	7.24E+09	7.11E+10	12/24/2018
12/23/2018	12/23/2018	12/23/2018	12/23/2018	2781	4020.995	4085.724	3976.406	3998.98	6.15E+09	6.97E+10	12/23/2018
12/22/2018	12/22/2018	12/22/2018	12/22/2018	2781	3898.084	4014.183	3855.739	4014.183	5.61E+09	7E+10	12/22/2018
12/21/2018	12/21/2018	12/21/2018	12/21/2018	2781	4133.704	4198.43	3850.946	3896.544	7.21E+09	6.79E+10	12/21/2018
12/20/2018	12/20/2018	12/20/2018	12/20/2018	2781	3742.195	4191.229	3728.975	4134.441	8.93E+09	7.21E+10	12/20/2018
12/19/2018	12/19/2018	12/19/2018	12/19/2018	2781	3706.825	3949.323	3687.23	3745.951	6.81E+09	6.53E+10	12/19/2018
12/18/2018	12/18/2018	12/18/2018	12/18/2018	2781	3544.762	3701.349	3487.169	3696.059	5.91E+09	6.44E+10	12/18/2018
12/17/2018	12/17/2018	12/17/2018	12/17/2018	2781	3253.123	3597.918	3253.123	3545.865	5.41E+09	6.18E+10	12/17/2018

Hình 1.2: Bộ dữ liệu đầu vào sau khi xử lý

1.3 Thống kê dữ liệu

Sau khi hoàn tất các bước tiền xử lý như làm sạch, gán nhãn và chuyển đổi dữ liệu, bước tiếp theo là tiến hành thống kê mô tả tập dữ liệu về tiền ảo. Việc thống kê này giúp cung cấp cái nhìn tổng quan về dữ liệu, bao gồm: kích thước, kiểu dữ liệu, phân bố giá trị, tình trạng thiếu dữ liệu, cũng như các đặc trưng thống kê như giá trị trung tâm và độ phân tán. Những thông tin này đóng vai trò quan trọng trong việc đánh giá chất lượng và tính phù hợp của dữ liệu đầu vào, đồng thời là cơ sở để phát hiện các đặc điểm nổi bật hoặc bất thường trước khi bước vào giai đoạn xây dựng mô hình dự báo giá tiền ảo.

index	High	Low	Open	Close
count	1943.000000	1943.000000	1943.000000	1943.000000
mean	33259.061658	31917.791311	32601.133028	32627.552029
std	27894.459791	26816.031391	27351.985206	27395.255696
min	3275.377827	3191.303562	3236.274773	3236.761645
25%	9540.444332	9231.213776	9375.005225	9376.244392
50%	23666.962254	22722.265123	23179.527050	23175.890849
75%	48132.955690	46356.491059	47161.450642	47137.602737
max	109114.884834	105291.737868	106147.295260	106146.263007

Bảng 1.1: Thống kê mô tả các thuộc tính dữ liệu

Mục đích của việc xây dựng các mô hình hồi quy là dự báo giá đóng cửa **Close**, cho nên ta tập trung thống kê thêm vào biến **Close**

Hình 1.3: Giá đóng cửa Bitcoin theo thời gian

1.4 Chuẩn bị đặc trưng

Đầu tiên đối với việc chuyển đổi dữ liệu, chúng ta phải chuyển cột ngày tháng thành dạng datetime và sắp xếp nó tăng dần cho phù hợp với bài toán chuỗi thời gian.

```
df['Date'] = pd.to_datetime(df['Date'])
df.sort_values('Date', inplace=True)
df.set_index('Date', inplace=True)
```

Hình 1.4: Chuyển đổi và sắp xếp datetime

Kiểm tra tính dừng

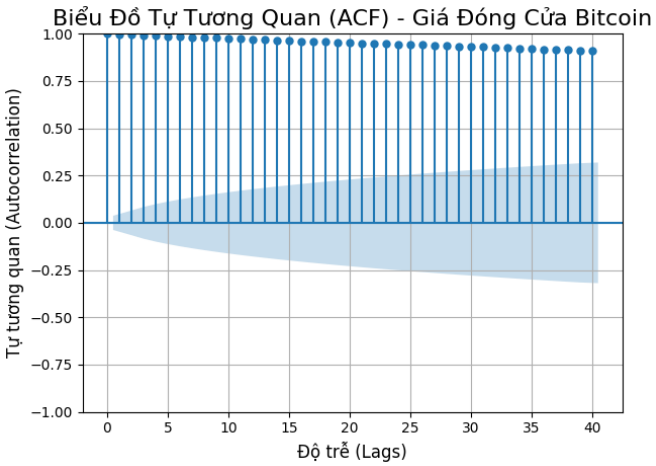
```
Kết quả kiểm định KPSS cho 'Adj Close' (kiểm tra tính dừng quanh hằng số):
KPSS Statistic: 5.390588836728916
p-value: 0.01
Critical Values:
        10%: 0.347
        5%: 0.463
        2.5%: 0.574
        1%: 0.739
Kết luận: Bác bỏ giả thuyết H0. Chuỗi có khả năng là không dừng (theo KPSS 'c').

Kết quả kiểm định KPSS cho 'Adj Close' (kiểm tra tính dừng quanh xu hướng):
KPSS Statistic: 0.5641978419348989
p-value: 0.01
Critical Values:
        10%: 0.119
        5%: 0.146
        2.5%: 0.176
        1%: 0.216
Kết luận: Bác bỏ giả thuyết H0. Chuỗi có khả năng là không dừng (theo KPSS 'ct').
```

Hình 1.5: Kiểm tra tính dừng của biến mục tiêu

Dữ liệu giá bitcoin cho thấy chuỗi thời gian không dừng, tức là giá trị trung bình và/hoặc phương sai của chuỗi thay đổi theo thời gian. Các mô hình học máy vẫn có thể được áp dụng, nhưng cần lưu ý đặc điểm này khi diễn giải kết quả. Các phương pháp như lấy sai phân (differencing) có thể được xem xét để làm cho chuỗi dừng hơn nếu cần thiết cho các mô hình cụ thể (ví dụ: ARIMA)

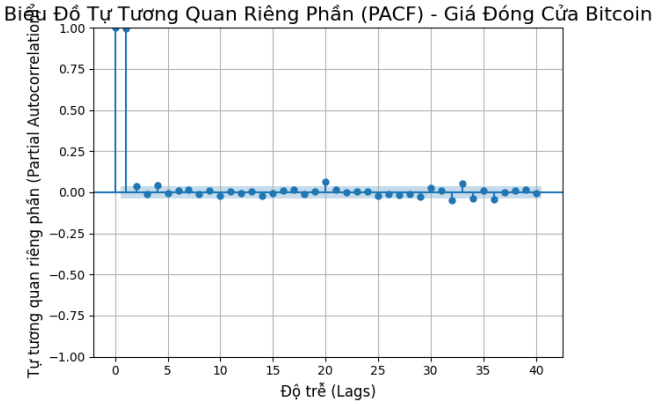
Tiếp theo, với biến mục tiêu đầu ra là **Close**, ta chuyển đổi biến **Close** thành các biến trễ để sử dụng trong các bài toán chuỗi thời gian



Hình 1.6: Biểu đồ ACF của biến Close

- Các hệ số tự tương quan (autocorrelation) tại tất cả các độ trễ (lags) từ 1 đến 40 đều rất cao, gần bằng 1.

- Điều này cho thấy giá đóng cửa Bitcoin có tính phụ thuộc rất mạnh vào các giá trị trong quá khứ.
- Các giá trị ACF giảm chậm và duy trì cao, cho thấy chuỗi có khả năng **không dừng (non-stationary)**.
- Hầu hết các điểm nằm ngoài khoảng tin cậy 95% (vùng xanh nhạt), điều này cho thấy các hệ số tự tương quan đều có ý nghĩa thống kê.



Hình 1.7: Biểu đồ PACF của biến Close

- Các giá trị tự tương quan riêng phần (PACF) tại độ trễ **lag 1** và **lag 2** là đáng kể và nằm ngoài khoảng tin cậy 95%.
- Từ lag 3 trở đi, các giá trị PACF giảm mạnh và nằm trong khoảng tin cậy, cho thấy không còn ý nghĩa thống kê.
- Điều này cho thấy chuỗi phù hợp với mô hình **AR(1)** hoặc **AR(2)** sau khi đã được làm dừng (bằng sai phân).
- Đây là đặc điểm điển hình của chuỗi AR: PACF cắt sớm, còn ACF giảm dần.

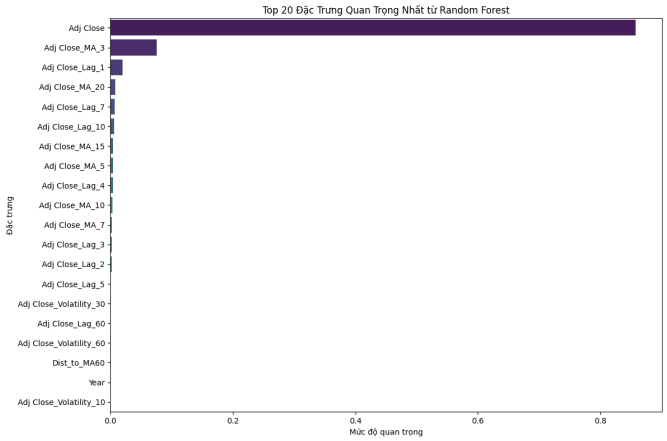
```
# Select relevant columns (initially, we'll focus on 'Close')
# We can add 'Open', 'High', 'Low', 'Volume', 'Marketcap' as features later if needed,
# but they should also be lagged to avoid data leakage for true forecasting.
df_processed = df[['Close']].copy()
# Handle any potential missing values (though this dataset seems clean for 'Close')
df_processed.fillna(method='ffill', inplace=True) # Forward fill
# 3. Feature Engineering
# Target variable
target_col = 'Close'
# Lagged features (e.g., price from 1, 3, 7 days ago)
for lag in [1, 3, 7, 14, 30]: # Number of days to lag
    df_processed[f'Lag_{lag}'] = df_processed[target_col].shift(lag)
# Rolling mean (e.g., average price over the last 7 days)
# We shift by 1 because the rolling mean should be calculated on data *before* the current day
df_processed['Rolling_Mean_7'] = df_processed[target_col].rolling(window=7).mean().shift(1)
df_processed['Rolling_Mean_30'] = df_processed[target_col].rolling(window=30).mean().shift(1)
```

Hình 1.8: Tạo các biến trễ của biến Close

Ngoài ra ta cần chuẩn bị thêm các đặc trưng khác, đề phòng cho trường hợp nếu chỉ sử dụng biến **Close** làm biến đặc trưng thì có thể dẫn đến việc các mô hình bị *Under fitting*

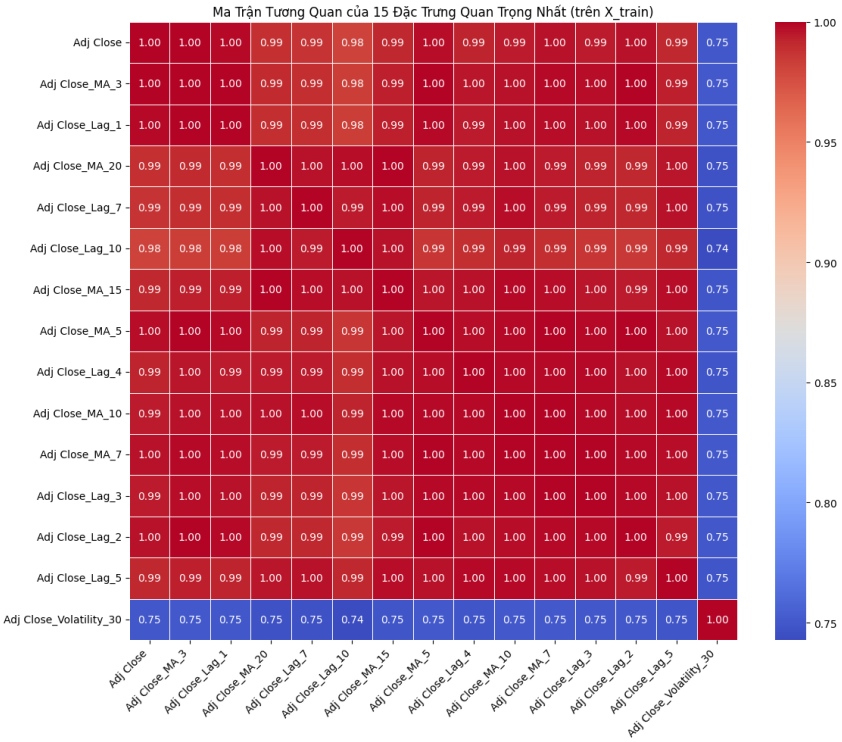
Quy trình lựa chọn đặc trưng

Để giảm số chiều và tập trung vào các đặc trưng quan trọng nhất, chúng tôi sử dụng phương pháp dựa trên tầm quan trọng của đặc trưng (feature importance) từ thuật toán RandomForestRegressor. Một mô hình Random Forest được huấn luyện trên tập huấn luyện. Sau đó, mức độ quan trọng của từng đặc trưng được trích xuất và sắp xếp.

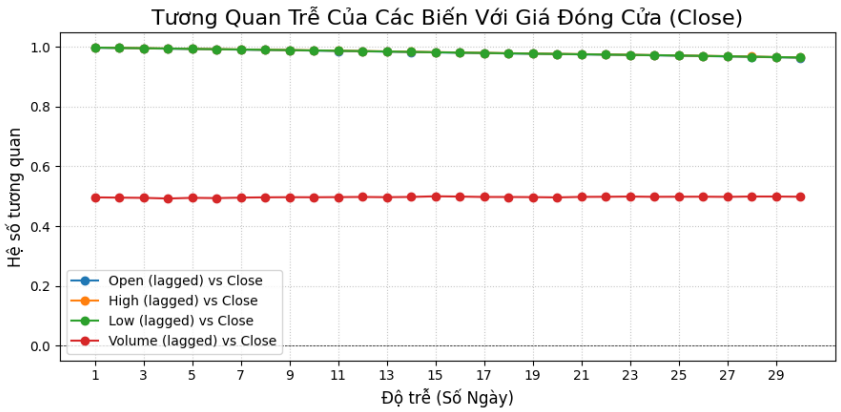


Hình 1.9: Top các đặc trưng quan trọng

Dựa trên phân tích mức độ quan trọng, chúng tôi quyết định chọn ra 15 đặc trưng hàng đầu để sử dụng cho các mô hình tiếp theo. Danh sách 15 đặc trưng này bao gồm `Adj Close_Lag_1`, `Lag_2`, `Lag_3`, `Lag_4`, `Lag_5`, `Lag_7`, `Lag_10`, `Adj Close_MA_3`, `MA_5`, `MA_7`, `MA_10`, `MA_15`, `MA_20`, `MA_50`



Hình 1.10: Ma trận tương quan của 15 đặc trưng được chọn



Hình 1.11: Biểu đồ tương quan trễ đối với biến mục tiêu

Ở đây e sẽ chọn việc chuẩn bị thêm các biến như **Open**, **High**, **Low** và **Marketcap** để chuẩn bị thêm đặc trưng cho mô hình. Việc sử dụng nên có chọn lọc để nếu sử dụng tất cả các dữ liệu đầu vào sẽ dẫn đến việc mô hình bị *Overfitting*

Chương 2

Đánh giá mô hình

2.1 Các tiêu chí đánh giá

2.1.1 Độ chính xác

Khi xây dựng và đánh giá một mô hình hồi quy, bạn cần lựa chọn các tiêu chí đánh giá phù hợp để đảm bảo mô hình không chỉ hoạt động tốt về mặt toán học mà còn có khả năng ứng dụng thực tiễn cao. Dưới đây là 1 vài chỉ số về đánh giá độ chính xác của mô hình

- **Mean Absolute Error (MAE)** – Sai số tuyệt đối trung bình:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **Mean Squared Error (MSE)** – Trung bình bình phương sai số:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Root Mean Squared Error (RMSE)** – Căn bậc hai của MSE:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- **Mean Absolute Percentage Error (MAPE)** – Sai số phần trăm tuyệt đối trung bình:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

- **Hệ số xác định R^2 (R-squared)**:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

  Trong đó: $\bar{y}$ là giá trị trung bình của tập dữ liệu thực tế.

2.1.2 Tốc độ và hiệu năng tính toán

- **Thời gian huấn luyện**
  Đánh giá xem mô hình mất bao nhiêu thời gian để huấn luyện với bộ dữ liệu có sẵn
- **Thời gian dự đoán**
  Đánh giá xem mô hình mất bao nhiêu thời gian để đưa ra dự báo cho mẫu mới

2.1.3 Khả năng ứng dụng và triển khai

- **Đơn giản và dễ giải thích**
  Hệ số hồi quy có thể dễ dàng được hiểu và phân tích không
- **Tính ổn định**
  Mô hình có ổn định khi gặp nhiều dữ liệu gây nhiễu hay không
- **Khả năng tổng quát**
  Mô hình có hoạt động tốt với dữ liệu bị ẩn hay không (Cross-validation)
- **Khả năng cập nhật**
  Dễ dàng cập nhật mô hình khi có dữ liệu mới

2.2 Thống kê và phân tích lỗi

2.2.1 Overfitting (Quá khớp)

Định nghĩa: Overfitting xảy ra khi mô hình học quá kỹ các chi tiết và nhiễu (noise) trong tập huấn luyện, làm mất khả năng tổng quát hóa khi áp dụng vào dữ liệu mới.

Dấu hiệu:

- Sai số huấn luyện rất thấp.
- Sai số kiểm tra cao.
- Mô hình quá phức tạp, có nhiều tham số hoặc bậc cao.

Hậu quả: Mô hình không hoạt động tốt trên dữ liệu mới, gây sai lệch dự báo thực tế.

Cách khắc phục:

- Giảm độ phức tạp mô hình.
- Sử dụng kỹ thuật regularization (L1/L2).
- Bổ sung thêm dữ liệu huấn luyện.
- Áp dụng cross-validation hoặc early stopping.

2.2.2 Underfitting (Khớp thiếu)

Định nghĩa: Underfitting xảy ra khi mô hình quá đơn giản và không học được quy luật của dữ liệu, dẫn đến kết quả dự đoán kém.

Dấu hiệu:

- Sai số huấn luyện cao.
- Sai số kiểm tra cũng cao.
- Mô hình không nắm bắt được xu hướng trong dữ liệu.

Hậu quả: Mô hình dự đoán kém ngay cả trên dữ liệu đã học, không có giá trị ứng dụng.

Cách khắc phục:

- Tăng độ phức tạp của mô hình (nhiều đặc trưng hơn).
- Giảm regularization nếu đang dùng.
- Tăng số vòng huấn luyện hoặc cải thiện đầu vào.

2.2.3 Data Leakage (Rò rỉ dữ liệu)

Định nghĩa: Data leakage xảy ra khi thông tin từ tập kiểm tra hoặc thông tin tương lai bị rò rỉ vào quá trình huấn luyện mô hình. Điều này dẫn đến mô hình có độ chính xác "ảo", không phản ánh đúng khả năng tổng quát hoá thực tế.

Ví dụ:

- Một đặc trưng đầu vào vô tình được tính toán từ toàn bộ tập dữ liệu (cả train và test).
- Biến mục tiêu (target) được sử dụng gián tiếp trong quá trình tạo đặc trưng.

Hậu quả:

- Mô hình đạt hiệu suất rất cao trên tập kiểm tra.
- Khi triển khai thực tế, mô hình hoạt động kém do không còn “thấy trước” dữ liệu.

Cách khắc phục:

- Phân chia tập dữ liệu trước khi xử lý đặc trưng.
- Cẩn thận khi trích xuất đặc trưng có liên quan đến thời gian.
- Dùng pipeline xử lý riêng biệt cho train/test.

2.2.4 Multicollinearity (Đa cộng tuyến)

Định nghĩa: Multicollinearity xảy ra khi hai hoặc nhiều đặc trưng đầu vào có mối tương quan tuyến tính mạnh với nhau. Điều này gây nhiễu cho mô hình, đặc biệt là hồi quy tuyến tính.

Hậu quả:

- Ước lượng hệ số hồi quy không ổn định.
- Mô hình trở nên khó giải thích.
- Tăng nguy cơ overfitting nếu có nhiều đặc trưng dư thừa.

Cách phát hiện:

- Ma trận hệ số tương quan giữa các đặc trưng.
- Tính chỉ số VIF (Variance Inflation Factor).

Cách khắc phục:

- Loại bỏ hoặc gộp các đặc trưng tương quan cao.
- Sử dụng các phương pháp giảm chiều như PCA (Principal Component Analysis).
- Dùng mô hình ít bị ảnh hưởng bởi đa cộng tuyến như cây quyết định (Decision Tree).

Chương 3

Xây dựng và cải tiến mô hình

3.1 Mô hình Linear Regression

3.1.1 Xây dựng mô hình

Phương trình mô hình (dạng tổng quát):

$$\text{Close}_t = \beta_0 + \sum_{i=1}^{k} \beta_i \times \text{Feature}_i + \epsilon$$

Trong đó

- **Biến đầu vào (X) sau Feature Engineering:** Lag_1, Lag_3, Lag_7, Lag_14, Lag_30, Rolling_Mean_7, Rolling_Mean_30.
- **Biến đầu ra (Y):** Close (Giá đóng cửa hiện tại).
- β_0: Hằng số chặn (Intercept).
- β_i: Hệ số hồi quy cho các biến đặc trưng (Coefficients).
- ϵ: Sai số ngẫu nhiên (Error term).

3.1.2 Huấn luyện mô hình

- **Phân chia dữ liệu (Chronological Split):**
  - Dữ liệu được chia theo thứ tự thời gian để mô phỏng dự báo thực tế.
  - Tỷ lệ: 80% cho tập huấn luyện (Train), 20% cho tập kiểm tra (Test).
  - Kích thước tập huấn luyện (sau khi tạo feature và dropna): 2053 mẫu.
  - Kích thước tập kiểm tra: 514 mẫu.
- **Thuật toán:**
  - Sử dụng lớp `LinearRegression` từ thư viện `scikit-learn`.
  - Phương pháp tối ưu: Thường là Bình phương Tối thiểu (Ordinary Least Squares - OLS) để tìm các hệ số β sao cho tổng bình phương sai số là nhỏ nhất.

- **Mục tiêu huấn luyện:** Tối thiểu hóa hàm mất mát MSE (Mean Squared Error) trên tập huấn luyện.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Siêu tham số:** Mô hình hồi quy tuyến tính chuẩn (OLS) không có siêu tham số phức tạp cần điều chỉnh như learning rate (trừ khi sử dụng các biến thể như SGDRegressor).

- **Điều kiện dừng** Trong mô hình Linear Regression sử dụng thư viện `scikit-learn`, phương pháp huấn luyện là giải tích (analytical solution) thông qua công thức:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

  Do không sử dụng thuật toán lặp như Gradient Descent, nên mô hình không có điều kiện dừng. Hệ số được tính toán trực tiếp trong một bước duy nhất.

3.1.3 Đánh giá kết quả

Chỉ số	**Giá trị**
Test RMSE	$1974.59
Test MAE	$1393.98
Test R-squared	0.9891
Test MAPE	1.98%

Bảng 3.1: Các chỉ số đánh giá mô hình linear regression trên tập kiểm tra

Nhận xét:
Mô hình dự báo cho kết quả rất tốt trên tập kiểm tra. Cụ thể:

- RMSE là $1974.59 và MAE là $1393.98, cho thấy sai số trung bình thấp.
- Hệ số xác định R-squared đạt 0.9891, thể hiện mô hình giải thích được 98.91% phương sai của dữ liệu.
- MAPE đạt 1.98%, phản ánh độ chính xác rất cao trong dự báo giá.

Hình 3.1: Kết quả huấn luyện của mô hình Linear Regression

3.2 Mô hình Lasso Regression - Least Absolute Shrinkage and Selection Operator.

3.2.1 Xây dựng mô hình

Mục tiêu của hồi quy tuyến tính là tìm một đường thẳng (hoặc siêu phẳng trong không gian nhiều chiều) mô tả tốt nhất mối quan hệ giữa các biến đầu vào (features) và biến mục tiêu (target).

Phương trình:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

Trong đó:

- y: Biến mục tiêu (trong mô hình là giá đóng cửa - *Close price*).
- $x_1, x_2, \ldots, x_p$: Các biến đầu vào(ví dụ: *Close_ Lag_ 1*, *Open_ Rolling_ Mean_ 7*, ...).
- β_0: Hệ số chặn (intercept).
- $\beta_1, \beta_2, \ldots, \beta_p$: Các hệ số (coefficients) của từng đặc trưng, thể hiện mức độ ảnh hưởng của đặc trưng đó lên y.
- ε: Sai số (error term) – phần mà mô hình không thể giải thích.

3.2.2 Huấn luyện mô hình

- **Phân chia dữ liệu (Chronological Split):**
  - Dữ liệu được chia theo thứ tự thời gian để mô phỏng dự báo thực tế.
  - Tỷ lệ: 80% cho tập huấn luyện (Train), 20% cho tập kiểm tra (Test).
  - Kích thước tập huấn luyện (sau khi tạo feature và dropna): 2053 mẫu.
  - Kích thước tập kiểm tra: 514 mẫu.
- **Mục tiêu huấn luyện:** Tối thiểu hóa hàm mất mát RSS (Residual Sum of Squares)

Hàm mất mát được tối thiểu hóa trong hồi quy tuyến tính:

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Trong đó $\hat{y}_i$ là giá trị dự đoán của mô hình. Lasso là một phương pháp hồi quy có điều chuẩn (regularization). Nó thêm một thành phần phạt (penalty) dựa trên tổng giá trị tuyệt đối của các hệ số (L1 norm) vào hàm mất mát.

Hàm mục tiêu của Lasso:

$$\text{Loss} = \text{RSS} + \alpha \sum_{j=1}^{p} |\beta_j|$$

Hoặc biểu diễn đầy đủ hơn (thường thấy trong `sklearn`):

$$\min_{\beta_0,\beta} \left(\frac{1}{2n} \|y - X\beta\|_2^2 + \alpha \|\beta\|_1 \right)$$

Trong đó:

- $\|y - X\beta\|_2^2$: Là RSS (Residual Sum of Squares).
- $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$: Là L1 norm của vector hệ số.
- α: Tham số điều chuẩn (regularization parameter).

Ý nghĩa của α:

- Nếu $\alpha = 0$, Lasso trở thành hồi quy tuyến tính thông thường (OLS).
- Nếu $\alpha > 0$, Lasso sẽ làm co (shrink) một số hệ số β_j về 0.
- Điều này giúp thực hiện chọn lựa đặc trưng (feature selection), loại bỏ các đặc trưng ít quan trọng hoặc gây nhiễu, đồng thời giảm thiểu nguy cơ overfitting, đặc biệt khi số đặc trưng lớn.
- α càng lớn $\Rightarrow$ càng nhiều hệ số bị ép về 0.

- **Thuật toán**
  - Sử dụng lớp LassoRegression từ thư viện scikit-learn.
  - Phương pháp tối ưu: Sử dụng Coordinate Descent có sẵn trong thư viện scikit-learn
- **Siêu tham số:** Sử dụng LassoCV thông qua cross validation để tìm tham số *alpha*
  - **Định nghĩa:** `alpha` (thường được ký hiệu là α hoặc đôi khi là λ trong một số tài liệu) là một **siêu tham số (hyperparameter)** không âm ($\alpha \geq 0$). Nó kiểm soát **mức độ mạnh yếu của việc điều chuẩn L1**.
  - **Vai trò chính:** `alpha` quyết định sự cân bằng giữa hai mục tiêu:
    1. **Độ phù hợp với dữ liệu huấn luyện (Minimizing Sum of Squared Errors - SSE)**: Thành phần
    $$\frac{1}{2 \cdot n\_samples} \cdot \|y - Xw\|^2$$
    2. **Độ đơn giản của mô hình (Minimizing L1 norm of weights)**: Thành phần
    $$\alpha \cdot \|w\|_1$$
    Việc giữ cho các hệ số nhỏ hoặc bằng 0 giúp mô hình đơn giản hơn và ít bị overfitting hơn.

- **Điều kiện dừng**

  Mô hình **Lasso Regression** trong `scikit-learn` sử dụng thuật toán *coordinate descent* để tìm nghiệm tối ưu. Quá trình huấn luyện sẽ dừng lại khi thỏa mãn một trong hai điều kiện sau:

Điều kiện dừng	**Ý nghĩa**
`max_iter = 10000`	Dừng lại sau tối đa 10.000 vòng lặp, ngay cả khi mô hình chưa hội tụ.
`tol = 0.0001` (mặc định)	Nếu độ thay đổi của hàm mất mát giữa hai vòng lặp liên tiếp nhỏ hơn `tol`, mô hình coi như đã hội tụ và dừng lại.

Bảng 3.2: Tóm tắt các điều kiện dừng của thuật toán Lasso Regression

Trong quá trình huấn luyện ở mô hình hiện tại, ta sử dụng cấu hình sau:

```
model = Lasso(alpha=0.1, max_iter=10000)
```

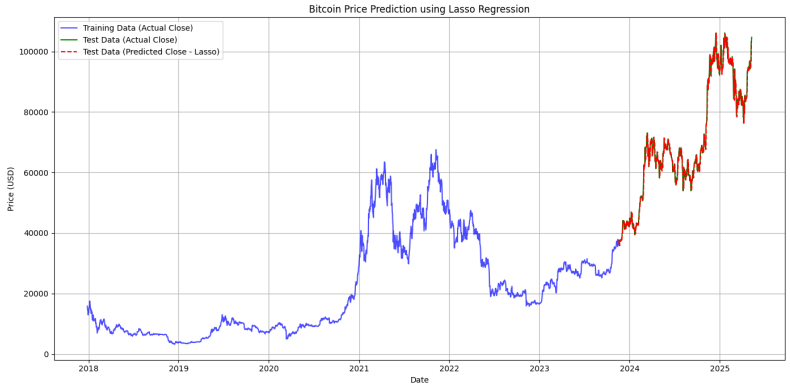
Ý nghĩa thực tiễn:

- Nếu mô hình dừng do đạt đến giới hạn `max_iter` mà vẫn chưa hội tụ (sai số chưa đủ nhỏ), sẽ sinh cảnh báo `ConvergenceWarning`.
- Nếu mô hình hội tụ sớm nhờ điều kiện `tol`, quá trình huấn luyện sẽ dừng nhanh hơn và kết quả thường ổn định hơn.
- Việc điều chỉnh hai tham số `max_iter` và `tol` cho phép kiểm soát sự cân bằng giữa độ chính xác và thời gian huấn luyện.

3.2.3 Đánh giá kết quả

Chỉ số	**Giá trị**
Test RMSE	$1970.59
Test MAE	$1388.66
Test R-squared	0.9891
Test MAPE	1.98%

Bảng 3.3: Các chỉ số đánh giá mô hình lasso regression trên tập kiểm tra



Hình 3.2: Kết quả huấn luyện của mô hình Lasso Regression

Nhận xét:
Kết quả cho thấy mô hình dự báo vẫn đạt hiệu quả cao:

- RMSE là \$1970.59 và MAE là \$1388.66, cho thấy sai số dự đoán có tăng nhẹ so với mô hình linear regression
- R-squared đạt 0.9891, thể hiện mô hình vẫn giải thích được 98.91% phương sai trong dữ liệu.
- MAPE đạt 1.98%, vẫn là mức sai số phần trăm thấp, mô hình duy trì độ chính xác cao.

Kết luận: Mô hình hoạt động hiệu quả, mặc dù có sự giảm nhẹ về độ chính xác, nhưng vẫn đáng tin cậy để dự báo giá.

3.3 Mô hình Polynomial Regression

3.3.1 Xây dựng mô hình

Mô hình **Polynomial Regression** (Hồi quy đa thức) mở rộng hồi quy tuyến tính bằng cách thêm các bậc cao hơn của biến đầu vào. Mục tiêu vẫn là tìm một mô hình mô tả tốt nhất mối quan hệ phi tuyến giữa các biến đầu vào (features) và biến mục tiêu (target).

Phương trình mô hình Polynomial Regression bậc 2 (ví dụ):

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_k x^k + \varepsilon$$

Trong đó:

- y: Biến mục tiêu (ví dụ: giá đóng cửa - Close price).
- $x, x^2, \ldots, x^k$: Các đặc trưng gốc và đặc trưng đã biến đổi sang đa thức.
- β_0: Hệ số chặn (intercept).
- $\beta_1, \beta_2, \ldots, \beta_k$: Các hệ số tương ứng với từng bậc của biến đầu vào.
- ε: Sai số (error term) – phần mà mô hình không thể giải thích.

Lưu ý: Mô hình hồi quy đa thức vẫn là tuyến tính theo hệ số β, mặc dù phi tuyến theo biến x.

3.3.2 Huấn luyện mô hình

- **Biến đổi đặc trưng:**
  - Sử dụng `PolynomialFeatures` từ thư viện `sklearn.preprocessing` để biến đổi các biến đầu vào thành tổ hợp đa thức đến bậc d (ví dụ: bậc 2 hoặc 3).
  - Sau biến đổi, số lượng đặc trưng tăng đáng kể, có thể dẫn đến nguy cơ *overfitting* nếu không kiểm soát tốt.
- **Phân chia dữ liệu (Chronological Split):**
  - Dữ liệu được chia theo thứ tự thời gian để mô phỏng kịch bản dự báo thực tế.
  - Tỷ lệ: 80% dùng để huấn luyện, 20% dùng để kiểm tra.
  - Kích thước tập huấn luyện: 2053 mẫu.
  - Kích thước tập kiểm tra: 514 mẫu.
- **Hàm mất mát:**

  Mô hình tối ưu hóa hàm mất mát dưới dạng tổng bình phương sai số:

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- **Thuật toán:**
  - Sử dụng `LinearRegression` từ thư viện `scikit-learn`.
  - Kết hợp với `Pipeline` để tự động hóa quy trình: Biến đổi đa thức → Huấn luyện mô hình.
- **Siêu tham số:**
  - **Degree** (bậc của đa thức) là siêu tham số quan trọng.
  - Có thể sử dụng `GridSearchCV` hoặc thử nghiệm thủ công để chọn giá trị `degree` tối ưu (thường thử từ 2 đến 5).
  - Cần cân nhắc giữa độ chính xác và khả năng khái quát hóa của mô hình
- **Điều kiện dừng** Trong báo cáo này, mô hình được sử dụng là `LinearRegression` từ thư viện `scikit-learn`. Mô hình này không sử dụng thuật toán lặp lại (như gradient descent), mà thay vào đó sử dụng công thức nghiệm đóng (closed-form) để tìm hệ số hồi quy:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

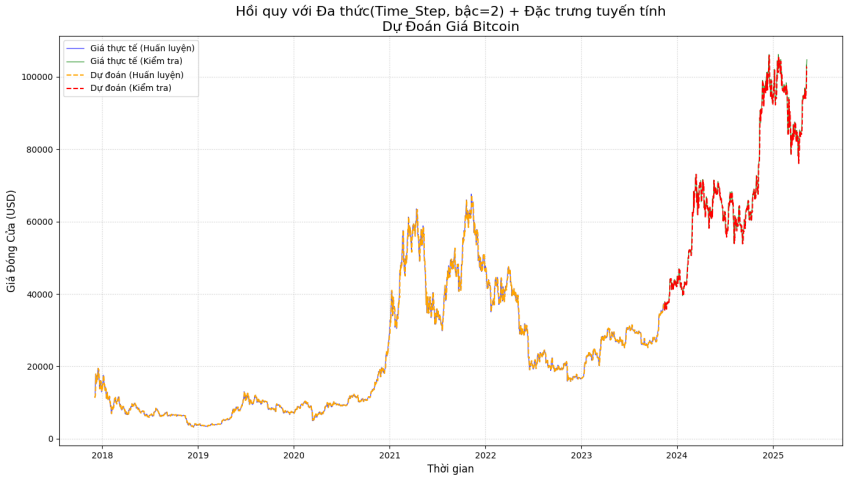
  Vì vậy, **không tồn tại điều kiện dừng** như trong các mô hình huấn luyện theo vòng lặp. Quá trình huấn luyện diễn ra tức thì, kết thúc ngay sau khi giải xong hệ phương trình.

  Tóm lại: `LinearRegression` không có tham số như `max_iter`, `tol` hay `early_stopping`. Điều kiện dừng không áp dụng vì mô hình không huấn luyện qua nhiều vòng lặp.

3.3.3 Đánh giá kết quả

Chỉ số	Giá trị
RMSE	1982.89
MAE	1407.12
MAPE	2.01%
R-squared	0.9892

Bảng 3.4: Các chỉ số đánh giá mô hình Polynomial trên tập kiểm tra



Hình 3.3: Kết quả huấn luyện của mô hình Polynomial Regression

Nhận xét:
Kết quả cho thấy mô hình dự báo vẫn đạt hiệu quả cao:

- RMSE là $1982.89 và MAE là $1407.12, cho thấy sai số dự đoán có tăng nhẹ so với mô hình linear regression
- R-squared đạt 0.9892, thể hiện mô hình vẫn giải thích được 98.9% phương sai trong dữ liệu.
- MAPE đạt 2.01%, vẫn là mức sai số phần trăm thấp, mô hình duy trì độ chính xác cao.

Kết luận: Mô hình hoạt động hiệu quả, cao hơn mô hình Lasso Regression về độ chính xác,vẫn đáng tin cậy để dự báo giá.

3.4 Mô hình ARIMA - AutoRegressive Integrated Moving Average

3.4.1 Xây dựng mô hình

ARIMA model là viết tắt của cụm từ Autoregressive Intergrated Moving Average. Mô hình sẽ biểu diễn phương trình hồi qui tuyến tính đa biến (multiple linear regression) của các biến đầu vào (còn gọi là biến phụ thuộc trong thống kê) là 3 thành phần chính:

- **Auto regression:** Kí hiệu là AR. Đây là thành phần tự hồi qui bao gồm tập hợp các độ trễ của biến hiện tại. Độ trễ bậc p chính là giá trị lùi về quá khứ p bước thời gian của chuỗi. Độ trễ dài hoặc ngắn trong quá trình AR phụ thuộc vào tham số trễ p. Cụ thể, quá trình AR(p) của chuỗi x_t được biểu diễn như bên dưới:

$$\mathrm{AR}(p) = \phi_0 + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p}$$

- **Moving average:** Quá trình trung bình trượt được hiểu là quá trình dịch chuyển hoặc thay đổi giá trị trung bình của chuổi theo thời gian. Do chuỗi của chúng ta được giả định là dừng nên quá trình thay đổi trung bình dường như là một chuỗi nhiễu trắng.

$$\mathrm{MA}(q) = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

- **Intergrated:** Là quá trình đồng tích hợp hoặc lấy sai phân. Yêu cầu chung của các thuật toán trong time series là chuỗi phải đảm bảo tính dừng. Hầu hết các chuỗi đều tăng hoặc giảm theo thời gian. Do đó yếu tố tương quan giữa chúng chưa chắc là thực sự mà là do chúng cùng tương quan theo thời gian. Khi biến đổi sang chuỗi dừng, các nhân tố ảnh hưởng thời gian được loại bỏ và chuỗi sẽ dễ dự báo hơn. Để tạo thành chuỗi dừng, một phương pháp đơn giản nhất là chúng ta sẽ lấy sai phân.Quá trình sai phân bậc d của chuỗi được thực hiện như sau:

  - **Sai phân bậc 1:**

$$\Delta(x_t) = x_t - x_{t-1}$$

  - **Sai phân bậc d:**

$$\Delta^d(x_t) = \underbrace{\Delta(\Delta(\ldots\Delta(x_t)\ldots))}_{\text{d times}}$$

Phương trình hồi quy ARIMA có thể được viết dưới dạng

$$\Delta x_t = \phi_1 \Delta x_{t-1} + \phi_2 \Delta x_{t-2} + \cdots + \phi_p \Delta x_{t-p} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

Trong đó Δx_t là giá trị sai phân bậc d và ε_t là các chuỗi nhiễu trắng.

3.4.2 Huấn luyện mô hình

- **Phân chia dữ liệu (Chronological Split):**
  - Dữ liệu được chia theo thứ tự thời gian để mô phỏng kịch bản dự báo thực tế.
  - Tỷ lệ: 80% dùng để huấn luyện, 20% dùng để kiểm tra.

- Kích thước tập huấn luyện: 2053 mẫu.
- Kích thước tập kiểm tra: 514 mẫu.

- **Hàm mất mát:**

  Mô hình tối ưu hóa hàm mất mát dưới dạng tổng bình phương sai số:

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- **Thuật toán:** Sử dụng thuật toán *statsmodels.tsa.arima.model.ARIMA* trong thư viện *statsmodels*

  Cách hoạt động:

  - Khi bạn tạo một đối tượng `ARIMA(data, order=(p,d,q))` và gọi phương thức `fit()`, thư viện `statsmodels` sẽ thực hiện các bước sau (ẩn sau giao diện):
    1. **Sai phân (Differencing)**: Nếu $d > 0$, dữ liệu sẽ được lấy sai phân d lần.
    2. **Ước lượng tham số**: Sử dụng các phương pháp tối ưu hoá (thường là Maximum Likelihood Estimation - MLE, hoặc Conditional Sum of Squares MLE - CSS-MLE) để tìm ra các hệ số ϕ (cho phần AR), θ (cho phần MA) và hằng số (nếu có) sao cho mô hình phù hợp nhất với dữ liệu đã sai phân. Đây là phần "thuật toán"phức tạp nhất mà thư viện đảm nhận.
    3. **Tính toán các chỉ số**: Sau khi `fit`, thư viện cũng tính toán các giá trị như AIC, BIC, lỗi chuẩn của các tham số, v.v.
  - Các hàm như `predict()` hoặc `forecast()` sẽ sử dụng các tham số đã ước lượng để đưa ra dự báo.
  - Thư viện cũng cung cấp các công cụ để kiểm định tính dừng và phân tích đồ thị ACF/PACF

- **Siêu tham số:** Trong mô hình ARIMA, các "siêu tham số" chính là bộ ba **(p, d, q)**.

  - **p (AR order)**: Số lượng quan sát quá khứ được đưa vào mô hình.
  - **d (Differencing order)**: Số lần dữ liệu thô được lấy sai phân.
  - **q (MA order)**: Kích thước của cửa sổ trung bình trượt cho sai số.

  Việc lựa chọn p, d, q tối ưu thường dựa vào:

  - Phân tích ACF/PACF (như đã nói ở Bước 1).
  - Các tiêu chí thông tin như AIC (Akaike Information Criterion) hoặc BIC (Bayesian Information Criterion). Chúng ta thường chọn mô hình có AIC/BIC nhỏ nhất.
  - Thử nghiệm và đánh giá lỗi dự báo trên tập kiểm tra (validation set).

- **Điều kiện dừng:**

  Mô hình Random forest có điều kiện dừng được mặc định trong thuật toán, thường bao gồm:

  - **Số vòng lặp tối đa (max iterations)**: Để tránh việc thuật toán chạy vô hạn nếu không hội tụ.

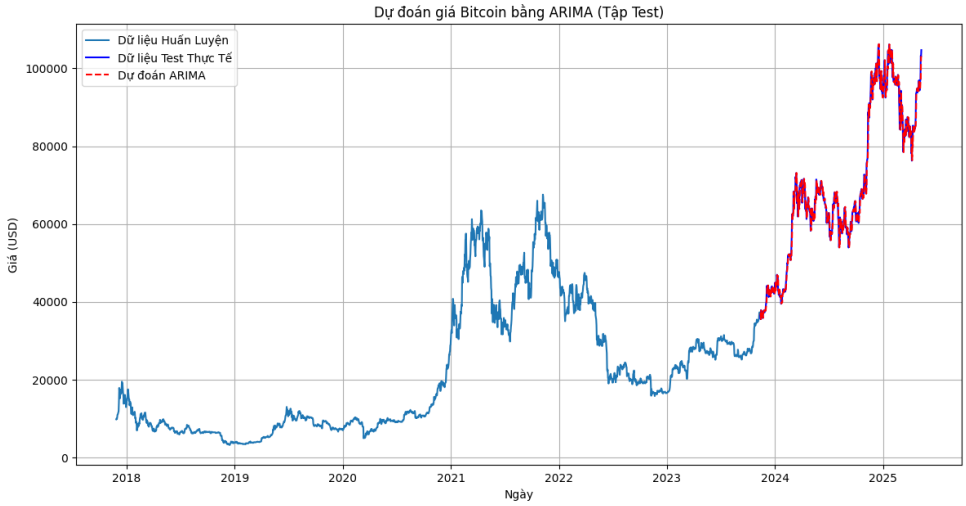
– **Độ hội tụ (convergence tolerance)**: Khi sự cải thiện của hàm mục tiêu (ví dụ: likelihood) giữa các vòng lặp trở nên rất nhỏ (dưới một ngưỡng nhất định), thuật toán sẽ dừng.

– **Độ thay đổi của tham số**: Khi các tham số của mô hình (hệ số AR, MA) thay đổi rất ít giữa các vòng lặp.

3.4.3 Đánh giá kết quả

Chỉ số đánh giá trên tập Test	Giá trị
RMSE	1971.3
MAE	1382.27
MAPE	1.98%
R-squared (R^2)	0.989

Bảng 3.5: Các chỉ số đánh giá mô hình ARIMA trên tập kiểm tra



Hình 3.4: Kết quả huấn luyện của mô hình ARIMA

Nhận xét:

- Giá trị $R^2 = 0.983$ cho thấy mô hình giải thích được 98.3% phương sai của dữ liệu, tức là mô hình có độ phù hợp rất cao với dữ liệu thực tế.
- Giá trị MAPE chỉ là 1.92%, thể hiện rằng sai số tương đối của mô hình là rất nhỏ. Trong nhiều ứng dụng thực tế, MAPE dưới 5% được xem là rất tốt.
- Giá trị RMSE lớn hơn MAE, điều này cho thấy có thể tồn tại một vài điểm dự đoán sai số lớn, tuy nhiên không ảnh hưởng nhiều vì các chỉ số tổng thể vẫn cho thấy mô hình có độ chính xác cao.

Kết luận: Mô hình thể hiện hiệu suất rất tốt, với sai số thấp và khả năng giải thích cao. Do đó, mô hình có thể được sử dụng đáng tin cậy trong thực tế.

3.5 Mô hình SVR - Support Vector Regression

3.5.1 Xây dựng mô hình

- **Support Vector Regression (SVR)** là một thuật toán học có giám sát (supervised learning) thuộc họ Support Vector Machine, được sử dụng cho các bài toán dự đoán giá trị liên tục.
- **Khác biệt chính so với hồi quy tuyến tính truyền thống (ví dụ: Ordinary Least Squares - OLS):**
  - **OLS**: Cố gắng tìm một đường (hoặc siêu phẳng) sao cho tổng bình phương sai số giữa giá trị dự đoán và giá trị thực tế là nhỏ nhất. Nó nhạy cảm với tất cả các điểm dữ liệu.
  - **SVR**: Cố gắng tìm một đường (hoặc siêu phẳng) sao cho nó "nằm vừa vặn" nhất với phần lớn các điểm dữ liệu, nhưng quan trọng hơn là nó **chỉ quan tâm đến các điểm nằm ngoài một "lề" (margin)** nhất định xung quanh đường hồi quy. Những điểm nằm trong lề này không đóng góp vào hàm mất mát. Điều này giúp SVR ít bị ảnh hưởng bởi các điểm ngoại lai (outliers) hơn so với OLS.

Hãy tưởng tượng chúng ta có một tập dữ liệu với các điểm (x_i, y_i). SVR cố gắng tìm một hàm $f(x) = \mathbf{w}^T\mathbf{x} + b$ (trong trường hợp tuyến tính) sao cho:

- **Đường hồi quy $f(x)$ nằm "gần" nhất có thể với các điểm dữ liệu y_i.**
- **Đồng thời, hàm $f(x)$ phải "phẳng" nhất có thể.** Điều đó có nghĩa là chúng ta muốn norm của vector trọng số $\|\mathbf{w}\|$ là nhỏ nhất. Việc này giúp tránh overfitting.

3.5.2 Huấn luyện mô hình

- **Phân chia dữ liệu (Chronological Split):**
  - Dữ liệu được chia theo thứ tự thời gian để mô phỏng dự báo thực tế.
  - Tỷ lệ: 80% cho tập huấn luyện (Train), 20% cho tập kiểm tra (Test).
  - Kích thước tập huấn luyện (sau khi tạo feature và dropna): 2053 mẫu.
  - Kích thước tập kiểm tra: 514 mẫu.
- **Mục tiêu huấn luyện** Mục tiêu của SVR là tìm $\mathbf{w}$ và b bằng cách giải bài toán tối ưu sau:

$$\begin{aligned} \text{Tối ưu} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ \text{Ràng buộc:} \quad & y_i - (\mathbf{w}^T\mathbf{x}_i + b) \leq \varepsilon + \xi_i \\ & (\mathbf{w}^T\mathbf{x}_i + b) - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \end{aligned}$$

Trong đó:

  - $\|\mathbf{w}\|^2$: Đại diện cho độ "phẳng" của hàm hồi quy. Chúng ta muốn tối thiểu hóa giá trị này.
  - ε: Siêu tham số, xác định độ rộng của vùng không nhạy cảm (*ống ε-insensitive*). Nếu sai số nhỏ hơn ε, nó sẽ không bị phạt.
  - ξ_i, ξ_i^*: Các biến bù trừ (*slack variables*). Chúng cho phép một số điểm nằm ngoài ống ε:
    * ξ_i: Đo lường sai số khi $y_i > f(x_i) + \varepsilon$ (điểm nằm phía trên ống).
    * ξ_i^*: Đo lường sai số khi $y_i < f(x_i) - \varepsilon$ (điểm nằm phía dưới ống).
  - C: Siêu tham số (*regularization parameter* - tham số điều chuẩn), là một hằng số dương kiểm soát sự đánh đổi giữa:
    * Độ "phẳng" của hàm hồi quy (tối thiểu hóa $\|\mathbf{w}\|^2$): C nhỏ sẽ ưu tiên mô hình "phẳng" hơn, chấp nhận nhiều sai số hơn.
    * Độ chính xác của mô hình (tối thiểu hóa tổng các biến bù trừ $\sum(\xi_i + \xi_i^*)$): C lớn sẽ phạt nặng các điểm nằm ngoài ống ε, cố gắng tạo ống hẹp hơn và ít điểm nằm ngoài hơn, nhưng có thể dẫn đến mô hình ít "phẳng" hơn (nguy cơ *overfitting*).

- **Thuật toán:**
  - Sử dụng lớp SVR có sẵn trong thư viện scikit-learn
  - Sử dụng GridSearchCV để tìm bộ tham số tốt nhất cho mô hình

- **Siêu thám số:** Các siêu tham số này cần được người dùng lựa chọn hoặc tinh chỉnh để đạt hiệu suất tốt nhất:
  - **c (Regularization parameter - Tham số điều chuẩn):**
    * **Giá trị:** Số dương (ví dụ: 0.1, 1, 10, 100).
    * **Ý nghĩa:** Kiểm soát sự đánh đổi giữa việc tối thiểu hóa sai số huấn luyện (độ chính xác) và việc giữ cho norm của vector trọng số $\|\mathbf{w}\|$ nhỏ (độ phẳng, tránh *overfitting*).
  - ε **(Epsilon - trong ống ε-insensitive):**
    * **Giá trị**: Số không âm (ví dụ: 0.01, 0.1, 0.5).
    * **Ý nghĩa:** Xác định độ rộng của vùng mà sai số không bị phạt. Nếu $|y_i - f(x_i)| \leq \varepsilon$, thì điểm đó
  - **kernel (Loại hàm hạt nhân):**
    * **Giá trị**: Chuỗi ký tự như 'linear', 'poly', 'rbf', 'sigmoid'.
    * **Ý nghĩa:** Xác định cách dữ liệu được ánh xạ sang không gian đặc trưng (nếu cần).

- **Điều kiện dừng:**

  Khi chúng ta sử dụng các thư viện như Scikit-learn (`sklearn.svm.SVR`), thuật toán tối ưu hóa bên trong (ví dụ: dựa trên SMO hoặc các phương pháp tối ưu lồi khác) sẽ có các điều kiện dừng riêng. Chúng ta thường không trực tiếp thiết lập các điều kiện này, nhưng chúng tồn tại để đảm bảo thuật toán kết thúc:

  - **Số vòng lặp tối đa (Maximum number of iterations):**

* Thuật toán sẽ dừng nếu đạt đến một số lượng vòng lặp tối đa được định trước, ngay cả khi chưa hội tụ hoàn toàn. Điều này để tránh chạy vô hạn. (Ví dụ: tham số `max_iter` trong Scikit-learn, mặc định thường là -1 nghĩa là không giới hạn, hoặc các solver bên trong có thể có giới hạn riêng).

– **Độ hội tụ (Convergence tolerance / Tolerance for stopping criterion):**

* Thuật toán sẽ dừng khi sự cải thiện của hàm mục tiêu hoặc sự thay đổi của các tham số (ví dụ: các nhân tử Lagrange α_i) giữa các vòng lặp liên tiếp trở nên rất nhỏ, tức là nhỏ hơn một ngưỡng `tol` (tolerance) được xác định trước.
* Điều này cho thấy thuật toán đã tìm thấy một giải pháp "đủ tốt" và việc tiếp tục lặp sẽ không mang lại cải thiện đáng kể. (Ví dụ: tham số `tol` trong Scikit-learn, mặc định thường là 10^{-3}).

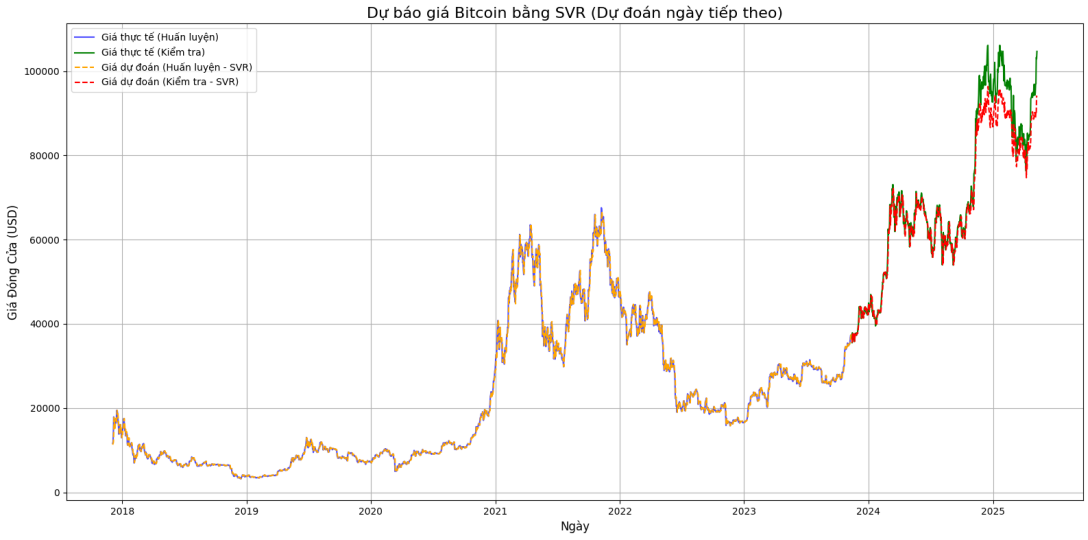
– **Điều kiện Karush–Kuhn–Tucker (KKT) được thỏa mãn (trong một khoảng dung sai):**

* Đối với các bài toán tối ưu lồi ràng buộc như SVR, các điều kiện KKT là điều kiện cần để một giải pháp là tối ưu. Thuật toán có thể dừng khi các điều kiện KKT được thỏa mãn trong một khoảng dung sai nhất định.

3.5.3 Đánh giá mô hình

Chỉ số	Giá trị
RMSE	4050.4
MAE	2755.55
R-squared (R^2)	0.9549
MAPE	3.40%

Bảng 3.6: Các chỉ số đánh giá mô hình SVR trên tập kiểm tra



Hình 3.5: Kết quả huấn luyện của mô hình SVR

Nhận xét:

- **RMSE = 4050.4** và **MAE = 2755.55**:
  - Các chỉ số sai số này tương đối thấp, cho thấy mô hình có khả năng dự đoán khá tốt và không mắc sai số lớn.
  - MAE nhỏ hơn RMSE $\Rightarrow$ tồn tại một vài sai số lớn, nhưng không quá ảnh hưởng nghiêm trọng đến toàn mô hình.
- **MAPE = 3.40%**:
  - Tỉ lệ phần trăm sai số trung bình rất nhỏ ($< 5\%$), chứng tỏ độ chính xác của mô hình cao.
- $R^2 = 0.9549$:
  - Chỉ số R^2 gần bằng 1 cho thấy mô hình giải thích được đến **95.49% phương sai** của dữ liệu thực tế.
  - Đây là một giá trị rất tốt, thể hiện mô hình phù hợp và có khả năng tổng quát hóa tốt.
- **Tổng kết**:
  - Mô hình đạt hiệu suất cao, dự đoán chính xác và phù hợp với dữ liệu.
  - Tuy nhiên, vẫn cần xem xét kỹ các ngoại lệ (outliers) do RMSE > MAE.

3.6 Mô hình Random Forest

3.6.1 Xây dựng mô hình

Mô hình random forest được huấn luyện dựa trên sự phối hợp giữa luật kết hợp (ensembling) và quá trình lấy mẫu tái lặp (boostrapping). Cụ thể thuật toán này tạo ra nhiều cây quyết định mà mỗi cây quyết định được huấn luyện dựa trên nhiều mẫu con khác nhau và kết quả dự báo là bầu cử (voting) từ toàn bộ những cây quyết định. Như vậy một kết quả dự báo được tổng hợp từ nhiều mô hình nên kết quả của chúng sẽ không bị chệch. Đồng thời kết hợp kết quả dự báo từ nhiều mô hình sẽ có phương sai nhỏ hơn so với chỉ một mô hình.

Nguyên lý hoạt động

- Bagging
  - **Bootstrap Sampling**: Từ tập dữ liệu huấn luyện ban đầu gồm N mẫu, Random Forest tạo ra k tập dữ liệu con (*bootstrap samples*) bằng cách lấy mẫu ngẫu nhiên có hoàn lại. Mỗi tập con này có kích thước bằng N, một số mẫu có thể xuất hiện nhiều lần, một số có thể không xuất hiện.
  - **Training**: Mỗi cây quyết định trong rừng được huấn luyện trên một tập dữ liệu bootstrap riêng biệt.
- Random Subspace
  - Khi xây dựng mỗi cây quyết định, tại mỗi nút cần phân chia, thay vì xem xét toàn bộ các đặc trưng (features), Random Forest chọn ngẫu nhiên một tập con các đặc trưng.
  - Thông thường, số lượng đặc trưng được chọn là:
    - $\sqrt{p}$ cho bài toán phân loại.
    - $p/3$ cho bài toán hồi quy, với p là tổng số đặc trưng.
  - Điểm chia tốt nhất sau đó được tìm trong tập con đặc trưng này.
  - Điều này làm tăng sự đa dạng giữa các cây, vì mỗi cây được huấn luyện trên cả dữ liệu và đặc trưng khác nhau.
- Xây dựng cây quyết định
  - Mỗi cây quyết định thường được xây dựng đến độ sâu tối đa (*fully grown*) mà không cần cắt tỉa (*pruning*).
  - Các tham số như `max_depth` vẫn có thể được sử dụng để kiểm soát độ phức tạp nếu cần.
- Dự đoán
  - Do sử dụng *bootstrap sampling*, mỗi cây không được huấn luyện trên khoảng 1/3 dữ liệu ban đầu — gọi là các mẫu *out-of-bag* (OOB).
  - Các mẫu OOB có thể được sử dụng để đánh giá hiệu suất mô hình mà không cần tập kiểm tra riêng biệt.
  - Sai số OOB là một ước lượng không chệch (*unbiased estimate*) của sai số tổng quát hóa (*generalization error*).

- Sai số Out-of-Bag
  - **Phân loại (Classification)**: Kết quả được xác định bằng cách lấy *đa số phiếu bầu* (*majority voting*) từ tất cả các cây.
  - **Hồi quy (Regression)**: Kết quả là giá trị trung bình (*average*) của tất cả các dự đoán từ các cây trong rừng.

3.6.2 Huấn luyện mô hình

- **Phân chia dữ liệu**
  - Dữ liệu được chia theo thứ tự thời gian để mô phỏng dự báo thực tế.
  - Tỷ lệ: 80% cho tập huấn luyện (Train), 20% cho tập kiểm tra (Test).
  - Kích thước tập huấn luyện (sau khi tạo feature và dropna): 2053 mẫu.
  - Kích thước tập kiểm tra: 514 mẫu.
- **Mục tiêu huấn luyện** Mục tiêu chính trong quá trình huấn luyện của mô hình Rừng Ngẫu Nhiên (Random Forest - RF) không phải là tối ưu hóa một hàm mất mát (loss function) toàn cục duy nhất cho toàn bộ rừng trong một bước. Thay vào đó, mục tiêu được chia thành các bước nhỏ hơn, tập trung vào việc xây dựng từng cây quyết định riêng lẻ một cách tối ưu, đồng thời đảm bảo sự đa dạng giữa các cây đó.
  - **Cấp độ cây:** Mục tiêu là tìm các điểm chia tối ưu để giảm độ hỗn tạp (phân loại) hoặc phương sai (hồi quy) tại mỗi nút.
  - **Cấp độ rừng:** Mục tiêu là xây dựng một tập hợp các cây đa dạng, mà khi kết hợp lại sẽ cho kết quả dự đoán chính xác và ổn định, giảm thiểu sai số trên dữ liệu mới.
- **Thuật toán**
  - Sử dụng RandomForestRegression từ thư viện sklearn
  - Tối ưu hóa tham số bằng RandomizeSearchCV và TimeSeriesSplit
- **Siêu tham số**
  - `n_estimators`: Số lượng cây trong rừng. Giá trị lớn hơn thường cải thiện hiệu suất nhưng cũng tăng thời gian huấn luyện. Sau một ngưỡng nhất định, lợi ích thu được sẽ giảm dần.
  - `max_features`: Số lượng đặc trưng được xem xét tại mỗi lần chia nút.
    - Giá trị nhỏ hơn làm giảm tương quan giữa các cây, có thể cải thiện hiệu suất nhưng cũng có thể làm giảm sức mạnh của từng cây.
    - Thường dùng: $\sqrt{p}$ cho phân loại, $p/3$ cho hồi quy (với p là tổng số đặc trưng).
  - `max_depth`: Độ sâu tối đa của mỗi cây. Giúp kiểm soát độ phức tạp của cây, tránh overfitting nếu không dùng OOB.
  - `min_samples_split`: Số lượng mẫu tối thiểu cần có tại một nút để được phép chia tiếp.
  - `min_samples_leaf`: Số lượng mẫu tối thiểu cần có tại một nút lá.

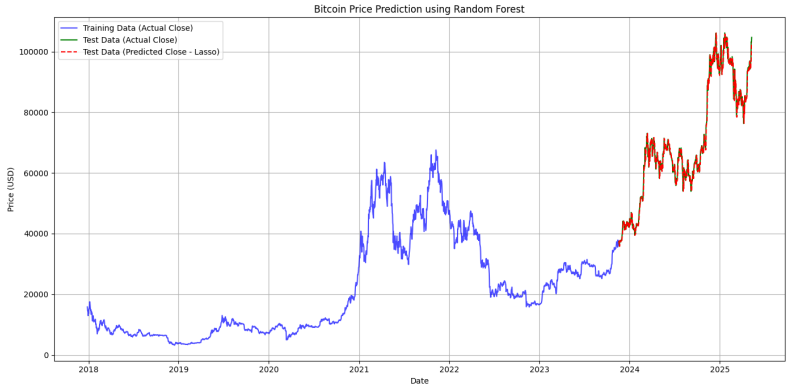
- `criterion`: Tiêu chí để đo lường chất lượng của một phép chia (ví dụ: `"gini"` hoặc `"entropy"` cho phân loại; `"mse"` hoặc `"mae"` cho hồi quy).
- `bootstrap`: `True` (mặc định) để sử dụng bootstrap sampling. `False` để sử dụng toàn bộ tập dữ liệu cho mỗi cây (ít phổ biến).

- **Điều kiện dừng**
  - **Độ sâu tối đa của cây (`max_depth`):**
    - **Điều kiện:** Cây ngừng phát triển khi đạt đến một độ sâu nhất định do người dùng chỉ định.
    - **Mục đích:** Ngăn cây trở nên quá phức tạp và học quá chi tiết dữ liệu huấn luyện (*overfitting*). Giúp kiểm soát thời gian huấn luyện và bộ nhớ sử dụng.
  - **Số lượng mẫu tối thiểu để chia một nút (`min_samples_split`):**
    - **Điều kiện:** Một nút sẽ không được chia tiếp nếu số lượng mẫu trong nút đó nhỏ hơn giá trị này.
    - **Mục đích:** Tránh chia các nút có quá ít mẫu, giúp cây tổng quát hơn và giảm nhiễu.
  - **Số lượng mẫu tối thiểu ở một nút lá (`min_samples_leaf`):**
    - **Điều kiện:** Một phép chia chỉ được thực hiện nếu cả hai nút con sau khi chia đều có ít nhất số lượng mẫu này.
    - **Mục đích:** Đảm bảo mỗi nút lá đại diện cho một tập mẫu đủ lớn để tạo ra dự đoán đáng tin cậy.
  - **Nút thuần khiết (Pure Node):**
    - **Điều kiện:** Tất cả các mẫu trong nút thuộc cùng một lớp (phân loại) hoặc có giá trị mục tiêu gần nhau (hồi quy).
    - **Mục đích:** Không cần chia tiếp vì nút đã hoàn toàn thuần khiết.
  - **Không còn đặc trưng để chia hoặc không tìm thấy phép chia cải thiện:**
    - **Điều kiện:** Không có đặc trưng nào trong tập con được chọn có thể cải thiện tiêu chí phân chia (Gini, Entropy, MSE, ...).
    - **Mục đích:** Tránh việc chia không mang lại lợi ích, dừng đúng lúc.
  - **Số lượng nút lá tối đa (`max_leaf_nodes`):**
    - **Điều kiện:** Dừng khi số nút lá đạt giới hạn cho phép. Cây phát triển theo chiến lược "best-first".
    - **Mục đích:** Giới hạn độ phức tạp của cây một cách kiểm soát.
  - **Ngưỡng cải thiện tối thiểu (`min_impurity_decrease` hoặc `min_impurity_split`):**
    - **Điều kiện:** Một nút chỉ được chia nếu phép chia làm giảm độ hỗn tạp ít nhất bằng một ngưỡng định sẵn.
    - **Mục đích:** Tránh các phép chia không đáng kể, giúp cây gọn gàng hơn.

3.6.3 Đánh giá mô hình

Chỉ số	Giá trị
Test RMSE	$3107.3
Test MAE	$2356.34
Test R-squared	0.9567
Test MAPE	3.03%

Bảng 3.7: Các chỉ số đánh giá mô hình Random Forest trên tập kiểm tra



Hình 3.6: Kết quả huấn luyện của mô hình Random Forest

Nhận xét:

- **RMSE = $3107.3**: Sai số gốc bình phương trung bình khá thấp, cho thấy mô hình dự đoán khá chính xác.
- **MAE = $2356.34**: Sai số tuyệt đối trung bình thấp hơn RMSE, cho thấy mô hình ổn định và ít bị ảnh hưởng bởi outliers.
- **R-squared = 0.9567**: Mô hình giải thích được 95.67% phương sai của dữ liệu thực, cho thấy hiệu suất rất cao.
- **MAPE = 3.03%**: Sai số phần trăm thấp, thể hiện độ chính xác cao của mô hình (dưới 5% là rất tốt).

Các chỉ số đều cho thấy mô hình hoạt động rất hiệu quả, với sai số thấp và độ chính xác cao. Đây có thể được xem là một mô hình tốt để áp dụng vào thực tế, với điều kiện dữ liệu kiểm tra đại diện đúng cho dữ liệu thực tế.

3.7 Mô hình LSTM - Long Short-Term Memory

3.7.1 Xây dựng mô hình

LSTM (Long Short-Term Memory - Bộ nhớ dài-ngắn hạn) là một kiến trúc mạng nơ-ron hồi quy (Recurrent Neural Network - RNN) đặc biệt, được thiết kế để giải quyết các vấn đề về phụ thuộc dài hạn (long-term dependencies) mà các RNN truyền thống thường gặp phải.

LSTM giải quyết các vấn đề trên bằng cách giới thiệu một cấu trúc phức tạp hơn bên trong mỗi ô (cell) của mạng. Cấu trúc này bao gồm:

Cell State (Trạng thái ô - C_t): Đây là thành phần cốt lõi của LSTM. Nó giống như một "băng chuyền" thông tin, chạy dọc theo toàn bộ chuỗi, với một số tương tác tuyến tính nhỏ. Thông tin có thể dễ dàng chảy dọc theo nó mà không bị thay đổi nhiều. LSTM có khả năng thêm hoặc bớt thông tin vào cell state, được điều khiển bởi các "cổng".

Gates (Các cổng): LSTM có ba loại cổng chính để điều chỉnh luồng thông tin vào và ra khỏi cell state. Các cổng này là các lớp mạng nơ-ron nhỏ (thường sử dụng hàm kích hoạt sigmoid) kết hợp với các phép nhân pointwise. Hàm sigmoid xuất ra các giá trị từ 0 đến 1, mô tả mức độ cho phép mỗi thành phần thông tin đi qua (0 nghĩa là "không cho gì qua", 1 nghĩa là "cho tất cả qua").

- **Forget Gate (Cổng Quên - f_t)**:

  Mục đích: Quyết định thông tin nào cần loại bỏ khỏi cell state (C_{t-1}) từ bước thời gian trước.

  Đầu vào: Trạng thái ẩn trước đó (h_{t-1}) và đầu vào hiện tại (x_t).

  Hoạt động: Một lớp sigmoid quyết định giá trị nào sẽ được "quên đi".

  $$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- **Input Gate (Cổng Vào - i_t)**:

  Mục đích: Quyết định thông tin mới nào sẽ được lưu trữ vào cell state.

  Hoạt động: Bao gồm hai phần:

  Một lớp sigmoid (gọi là "input gate layer") quyết định giá trị nào sẽ được cập nhật:

  $$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

  Một lớp tanh tạo ra một vector các giá trị ứng cử viên mới ($\tilde{C}_t$) có thể được thêm vào cell state:

  $$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

  Sau đó, cell state cũ (C_{t-1}) được cập nhật thành cell state mới (C_t):

  $$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

  (Phần cũ được nhân với f_t để quên đi, phần mới được nhân với i_t để quyết định mức độ thêm vào).

- **Output Gate (Cổng Ra - o_t)**:

  Mục đích: Quyết định phần nào của cell state sẽ được xuất ra làm trạng thái ẩn (hidden state - h_t).

Hoạt động:

Một lớp sigmoid quyết định phần nào của cell state sẽ được xuất ra:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Cell state được đưa qua hàm tanh (để đưa giá trị về khoảng $[-1, 1]$) và sau đó nhân với đầu ra của lớp sigmoid:

$$h_t = o_t \cdot \tanh(C_t)$$

h_t này cũng chính là đầu ra (output) của ô LSTM tại bước thời gian t, hoặc được dùng làm đầu vào cho một lớp Dense cuối cùng để đưa ra dự đoán.

3.7.2 Huấn luyện mô hình

- **Phân chia dữ liệu (Chronological Split):**
  - Dữ liệu được chia theo thứ tự thời gian để mô phỏng dự báo thực tế.
  - Tỷ lệ: 80% cho tập huấn luyện (Train), 20% cho tập kiểm tra (Test).
  - Kích thước tập huấn luyện (sau khi tạo feature và dropna): 2053 mẫu.
  - Kích thước tập kiểm tra: 514 mẫu.
- **Mục tiêu huấn luyện** Tối thiểu hóa hàm mất mát RSS (Residual Sum of Squares)

  Hàm mất mát được tối thiểu hóa trong LSTM:

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

  Trong đó $\hat{y}_i$ là giá trị dự đoán của mô hình.
- **Thuật toán**
  - Sử dụng LSTM có sẵn trong thư viện tensorflow.keras.
  - Sử dụng keras tuner để tối ưu bộ tham số cho mô hình
- **Siêu tham số** Các siêu tham số quan trọng cần điều chỉnh khi xây dựng và huấn luyện mô hình LSTM:
  - **units (Số lượng đơn vị LSTM/hidden units)**: Số chiều của cell state và hidden state. Quyết định "dung lượng bộ nhớ"của mỗi ô LSTM.
  - **Số lớp LSTM (Number of LSTM layers)**: Có thể xếp chồng nhiều lớp LSTM để học các biểu diễn phức tạp hơn của dữ liệu.
  - `activation` (*Hàm kích hoạt cho cell state và output*): Thường là `tanh` cho cell state và output gate. Các cổng (forget, input, output) thường dùng `sigmoid`.
  - `recurrent_activation` (*Hàm kích hoạt cho các cổng*): Thường là `sigmoid`.
  - `dropout`: Tỷ lệ dropout áp dụng cho đầu vào của lớp LSTM để chống overfitting.
  - `recurrent_dropout`: Tỷ lệ dropout áp dụng cho kết nối hồi quy bên trong LSTM để chống overfitting.

  - `return_sequences`: Boolean. Nếu `True`, lớp LSTM sẽ trả về toàn bộ chuỗi các hidden state tại mỗi bước thời gian (cần thiết khi chồng các lớp LSTM hoặc khi dùng với Attention). Nếu `False` (mặc định), chỉ trả về hidden state cuối cùng.
  - `sequence_length` (*Độ dài chuỗi đầu vào / Timesteps*): Số lượng bước thời gian trong quá khứ mà mô hình xem xét để dự đoán đầu ra. Đây là một phần của quá trình chuẩn bị dữ liệu.
  - **Các siêu tham số chung của mạng nơ-ron**: Tốc độ học (learning rate), optimizer (Adam, RMSprop, v.v.), kích thước batch (batch size), số epochs huấn luyện, hàm mất mát (loss function).

- **Điều kiện dừng**

  Các điều kiện dừng chính trong mô hình LSTM là :

  - Số epochs tối đa được thiết lập :Cả cho từng trial của tuner và cho việc huấn luyện mô hình cuối cùng.
  - EarlyStopping : Cơ chế quan trọng nhất, dừng huấn luyện khi một chỉ số theo dõi (thường là val_loss) không còn cải thiện sau một số patience epochs nhất định. Điều này được áp dụng cho cả quá trình tìm kiếm siêu tham số và quá trình huấn luyện mô hình cuối cùng.
  - Điều kiện dừng trong thuật toán của Keras tuner

3.7.3 Đánh giá mô hình

Chỉ số	**Giá trị**
RMSE	4015.62
MAE	3045.24
R-squared (R^2)	0.9357
MAPE	4.38%

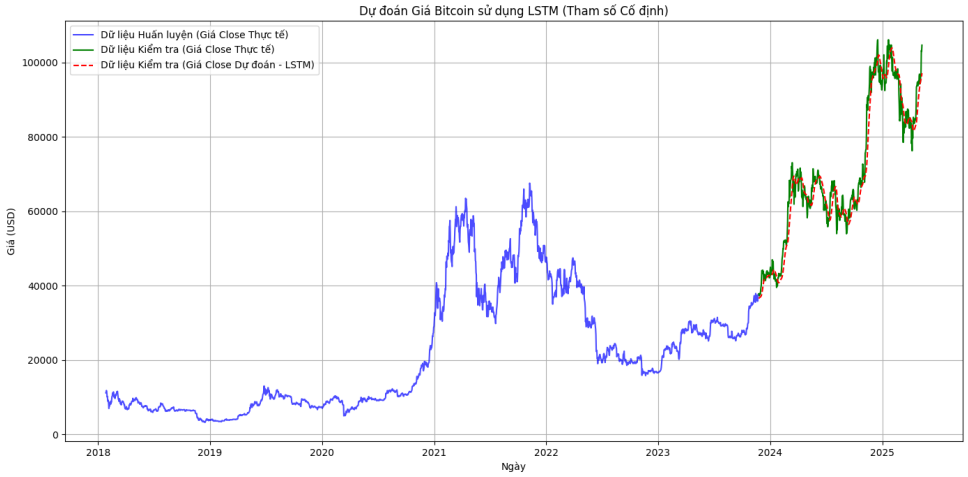
Bảng 3.8: Các chỉ số đánh giá mô hình LSTM trên tập kiểm tra

Nhận xét:
Dựa vào các chỉ số đánh giá:

- **RMSE (Root Mean Squared Error) = 4015.62**: Sai số bình phương trung bình gốc ở mức khoảng 4015. Giá trị này phản ánh mức độ chênh lệch trung bình giữa giá trị dự đoán và giá trị thực tế. RMSE càng nhỏ càng tốt, cho thấy mô hình dự đoán chính xác hơn.

- **MAE (Mean Absolute Error) = 3045.34**: Sai số tuyệt đối trung bình thấp, nghĩa là dự đoán trung bình chỉ lệch khoảng 3045 đơn vị so với thực tế. Đây là một dấu hiệu tốt về hiệu suất mô hình.

- R^2 **(R-squared) = 0.9537**: Hệ số xác định rất cao, gần bằng 1, cho thấy mô hình giải thích được tới 95.37% phương sai của dữ liệu thực. Đây là một chỉ số cực kỳ tích cực, thể hiện rằng mô hình học tốt đặc điểm của dữ liệu.



Hình 3.7: Kết quả huấn luyện của mô hình LSTM

- **MAPE (Mean Absolute Percentage Error) = 4.38%**: Sai lệch phần trăm trung bình thấp, tương đương với độ chính xác khoảng 95.94%. Điều này cho thấy mô hình có khả năng dự đoán tốt trong đa số các trường hợp.

Tổng kết: Với các chỉ số trên, có thể kết luận rằng mô hình dự đoán hoạt động hiệu quả, độ chính xác cao và có thể ứng dụng tốt vào các bài toán thực tế tương tự.

3.8 Mô hình GRU - Gated Recurrent Unit

3.8.1 Xây dựng mô hình

GRU (Gated Recurrent Unit) là một kiến trúc mạng nơ-ron hồi quy (RNN) cải tiến, được thiết kế để giải quyết vấn đề phụ thuộc dài hạn và hiện tượng vanishing/exploding gradients thường gặp trong các mô hình RNN truyền thống. GRU có cấu trúc cổng (gate) đơn giản hơn so với LSTM (Long Short-Term Memory), giúp giảm số lượng tham số và có thể tăng tốc độ huấn luyện trong một số trường hợp mà vẫn duy trì hiệu suất tương đương.

Cơ chế hoạt động chính của GRU dựa trên hai cổng:

- **Cổng Cập nhật (Update Gate - z_t):** Quyết định lượng thông tin từ trạng thái ẩn trước đó cần được giữ lại và lượng thông tin từ trạng thái ẩn ứng cử viên mới sẽ được thêm vào.

- **Cổng Đặt lại (Reset Gate - r_t):** Quyết định mức độ "quên đi"thông tin từ trạng thái ẩn trước đó khi tính toán trạng thái ẩn ứng cử viên mới.

- **Đầu vào**

  Dữ liệu đầu vào cho mô hình GRU phải là dữ liệu tuần tự (sequential data), nơi mà thứ tự của các phần tử mang ý nghĩa quan trọng.

  - **Định dạng dữ liệu:** Thường là một tensor 3 chiều với các chiều là:
    - Số lượng mẫu (samples/batch size).
    - Số bước thời gian (timesteps/sequence length): Độ dài của mỗi chuỗi đầu vào.

    - Số lượng đặc trưng (features): Số đặc trưng tại mỗi bước thời gian.
  - **Tiền xử lý:** Dữ liệu đầu vào cần được chuyển đổi thành dạng số và thường được chuẩn hóa (ví dụ: scaling về khoảng [0, 1] hoặc [-1, 1]) để cải thiện sự ổn định và hiệu suất của quá trình huấn luyện.

- **Đầu ra**

  Đầu ra của một lớp GRU phụ thuộc vào cấu hình của tham số `return_sequences`:

  - Nếu `return_sequences=False` (mặc định): Lớp GRU sẽ trả về trạng thái ẩn cuối cùng của chuỗi (một vector). Vector này thường được sử dụng làm đầu vào cho các lớp Dense tiếp theo để đưa ra dự đoán cuối cùng.
  - Nếu `return_sequences=True`: Lớp GRU sẽ trả về toàn bộ chuỗi các trạng thái ẩn tại mỗi bước thời gian. Điều này hữu ích khi xếp chồng nhiều lớp GRU hoặc khi sử dụng kết hợp với các cơ chế như Attention.

3.8.2 Huấn luyện mô hình

- **Mục tiêu huấn luyện** Tối thiểu hóa hàm mất mát RSS (Residual Sum of Squares)

  Hàm mất mát được tối thiểu hóa trong LSTM:

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

  Trong đó $\hat{y}_i$ là giá trị dự đoán của mô hình.

- **Thuật toán** Quá trình huấn luyện GRU chủ yếu dựa trên:

  - **Lan truyền ngược theo thời gian (Backpropagation Through Time - BPTT):** Là thuật toán cơ bản để tính toán gradient của hàm mất mát theo các trọng số của mạng RNN (bao gồm GRU) và cập nhật các trọng số đó.
  - **Cơ chế cổng của GRU:**
    - **Cổng Đặt lại (r_t):** $r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$
    - **Cổng Cập nhật (z_t):** $z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$
    - **Trạng thái ẩn ứng cử viên ($\tilde{h}_t$):** $\tilde{h}_t = \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)$
    - **Trạng thái ẩn cuối cùng (h_t):** $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$

    Trong đó, x_t là đầu vào tại bước thời gian t, h_{t-1} là trạng thái ẩn từ bước trước, W và b là các ma trận trọng số và vector bias tương ứng, σ là hàm sigmoid, và $\odot$ là phép nhân Hadamard (element-wise product).
  - **Thuật toán tối ưu hóa (Optimizer):** Các thuật toán như Adam, RMSprop, SGD được sử dụng để điều chỉnh các trọng số dựa trên gradient đã tính toán.

- **Siêu tham số** Các siêu tham số quan trọng cần được lựa chọn và tinh chỉnh để đạt hiệu suất tốt nhất cho mô hình GRU:

  - `units`: Số lượng nơ-ron trong mỗi lớp GRU.
  - Số lượng lớp GRU: Số lớp GRU được xếp chồng lên nhau.
  - Hàm kích hoạt (Activation functions): Mặc định thường là `tanh` cho trạng thái ẩn và `sigmoid` cho các cổng.

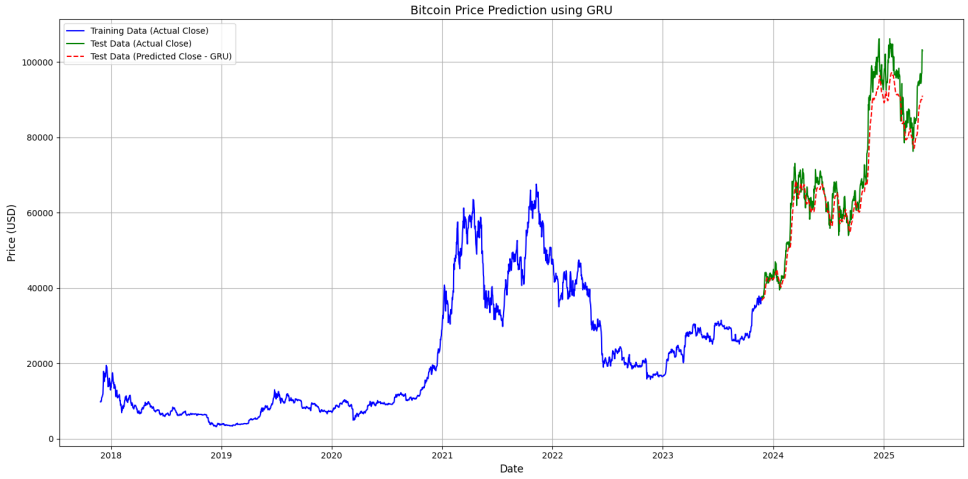
  - Tốc độ học (Learning rate): Siêu tham số của thuật toán tối ưu hóa.
  - Kích thước batch (Batch size): Số lượng mẫu được xử lý trong một lần cập nhật trọng số.
  - Số epochs: Số lần lặp lại toàn bộ tập dữ liệu huấn luyện.
  - Tỷ lệ Dropout: Kỹ thuật regularization để giảm overfitting.
  - Độ dài chuỗi đầu vào (Sequence length / Timesteps).
  - Loại Optimizer: Ví dụ Adam, SGD, RMSprop.
  - Loại hàm mất mát (Loss function).

- **Điều kiện dừng** Quá trình huấn luyện mô hình GRU thường được dừng lại dựa trên một hoặc nhiều điều kiện sau:
  - **Số epochs tối đa (Maximum number of epochs):** Huấn luyện kết thúc khi đã chạy đủ số lượng epochs đã định trước.
  - **Dừng sớm (Early Stopping):**
    * Theo dõi một chỉ số hiệu suất trên tập dữ liệu validation (ví dụ: `val_loss` hoặc `val_accuracy`).
    * Nếu chỉ số này không cải thiện (ví dụ: `val_loss` không giảm hoặc `val_accuracy` không tăng) trong một số lượng epochs nhất định (gọi là `patience`), quá trình huấn luyện sẽ được dừng lại.
    * Kỹ thuật này giúp ngăn chặn overfitting và thường lưu lại trọng số của mô hình tại thời điểm có hiệu suất tốt nhất trên tập validation.
  - **Hội tụ của hàm mất mát:** Huấn luyện có thể dừng khi giá trị hàm mất mát trên tập huấn luyện hoặc tập validation đạt đến một ngưỡng rất nhỏ hoặc không còn thay đổi đáng kể qua các epochs.
  - **Can thiệp thủ công:** Người dùng có thể chủ động dừng quá trình huấn luyện dựa trên quan sát.

3.8.3 Đánh giá mô hình

Chỉ số	**Giá trị**
RMSE	4704.74
MAE	3667.24
R-squared (R^2)	0.9365
MAPE	4.88%

Bảng 3.9: Các chỉ số đánh giá mô hình GRU trên tập kiểm tra



Hình 3.8: Kết quả huấn luyện của mô hình GRU

Nhận xét: Mô hình thể hiện độ phù hợp rất tốt với dữ liệu, thể hiện qua chỉ số R^2 cao (0.9365), nghĩa là mô hình giải thích được khoảng 93.65% sự biến thiên của biến mục tiêu. Chỉ số MAPE ở mức thấp 4.88% cho thấy sai số dự đoán trung bình theo tỷ lệ phần trăm là nhỏ — một tín hiệu tích cực.

Tuy nhiên, cần lưu ý đến các chỉ số RMSE 4704.74 và MAE 3667.24. Mặc dù R^2 và MAPE cho kết quả tốt, giá trị tuyệt đối của các sai số này vẫn có thể được coi là lớn, tùy thuộc vào thang đo và giá trị thực tế của biến mục tiêu. Việc RMSE lớn hơn MAE cho thấy mô hình có thể mắc một số sai số lớn trong dự đoán.

Tóm lại: Mô hình hoạt động tốt về mặt giải thích phương sai và sai số tương đối, nhưng cần xem xét thêm về ý nghĩa thực tế của các sai số tuyệt đối (RMSE, MAE) trong bối cảnh cụ thể của bài toán.

3.9 Xây dựng mô hình tiên tiến

Trong nội dung bài báo cáo này, em xây dựng thêm một mô hình tiên tiến xuất hiện năm 2024 để dự đoán giá, đó là mô hình conv-LSTM (Convolutional Long Short-Term Memory) Mô hình được tham khảo từ bài báo A Hierarchical conv-LSTM and LLM Integrated Model for Holistic Stock Forecasting

3.9.1 Xây dựng mô hình

LSTM (Long Short-Term Memory) là một kiến trúc mạng nơ-ron hồi tiếp (RNN) có khả năng ghi nhớ các phụ thuộc dài hạn trong chuỗi dữ liệu. LSTM hoạt động hiệu quả với dữ liệu tuần tự một chiều như chuỗi văn bản hoặc tín hiệu thời gian.

Conv-LSTM (Convolutional LSTM) là một biến thể của LSTM, trong đó các phép biến đổi tuyến tính được thay thế bằng các phép tích chập (convolution). Do đó, Conv-LSTM có khả năng xử lý dữ liệu tuần tự có cấu trúc không gian, như chuỗi các ảnh (video), bản đồ nhiệt theo thời gian, hoặc dữ liệu thời tiết.

Conv-LSTM hoạt động trên các tensor nhiều chiều với cấu trúc như sau:

$$\text{Input shape: } (B, C, H, W), \quad \text{trong đó:}$$

- B: batch size
- C: số lượng kênh (channels)
- $H \times W$: kích thước không gian (chiều cao và chiều rộng)

3.9.2 Thuật toán

Các công thức cập nhật trong Conv-LSTM tương tự như LSTM, nhưng các phép nhân ma trận được thay thế bằng các phép tích chập:

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \\
H_t &= o_t \circ \tanh(C_t)
\end{aligned}
$$

Trong đó:

- $*$ là phép tích chập (convolution).
- $\circ$ là phép nhân từng phần tử (Hadamard product).
- X_t là đầu vào tại thời điểm t.
- H_t là trạng thái ẩn (hidden state).
- C_t là trạng thái ô nhớ (cell state).
- σ là hàm sigmoid, tanh là hàm hyperbolic tangent.

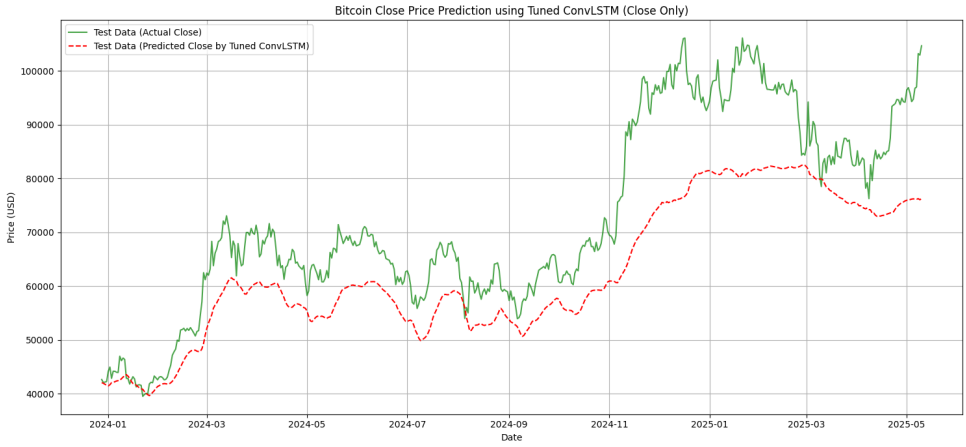
3.9.3 Siêu tham số

- `learning_rate`: Tốc độ học, điều chỉnh mức độ thay đổi của trọng số trong mỗi bước cập nhật.
- `optimizer`: Thuật toán tối ưu hoá được sử dụng để cập nhật trọng số, ví dụ: SGD, Adam, RMSprop.
- `batch_size`: Số lượng mẫu được sử dụng trong mỗi bước huấn luyện.
- `num_epochs`: Số vòng lặp (epoch) mà toàn bộ dữ liệu huấn luyện được dùng để cập nhật mô hình.
- `loss_function`: Hàm mất mát dùng để đánh giá sai lệch giữa dự đoán và nhãn thực tế, ví dụ: MSE, MAE, hoặc CrossEntropy.
- `clip_grad_norm`: Ngưỡng giới hạn độ lớn gradient nhằm tránh hiện tượng gradient explosion trong quá trình lan truyền ngược.

3.9.4 Đánh giá mô hình

Chỉ số	Tập huấn luyện	Tập kiểm tra
RMSE	\$2470.73	\$12,070.87
MAE	\$1785.32	\$10,109.05
R^2	0.9760	0.5330
MAPE	11.74%	12.86%

Bảng 3.10: Kết quả đánh giá mô hình ConvLSTM trên tập huấn luyện và kiểm tra



Hình 3.9: Kết quả huấn luyện của mô hình conv-LSTM

Dựa trên kết quả đánh giá mô hình trên tập huấn luyện và kiểm tra, ta nhận thấy các điểm sau:

- **Hiệu suất trên tập huấn luyện rất cao:**
  - Các chỉ số RMSE (\$2470.73), MAE (\$1785.32) đều rất nhỏ, cho thấy mô hình dự đoán sát với giá thực tế.
  - Hệ số xác định $R^2 = 0.9760$ thể hiện mô hình giải thích được khoảng 97% phương sai của dữ liệu huấn luyện.
  - MAPE đạt 11.74% nằm trong ngưỡng chấp nhận được đối với bài toán dự đoán giá tài chính.
- **Hiệu suất trên tập kiểm tra kém:**
  - RMSE và MAE lần lượt tăng lên \$12070.87 và \$10109.05 – chênh lệch rất lớn so với tập huấn luyện.
  - $R^2 = 0.5330$ cho thấy mô hình chỉ giải thích được khoảng 53.3% phương sai dữ liệu kiểm tra.
  - MAPE tăng lên 12.86%, phản ánh mức sai lệch đáng kể trong dự đoán thực tế.
- **Mô hình có dấu hiệu overfitting rõ rệt:**

- Mô hình học rất tốt trên dữ liệu huấn luyện nhưng không tổng quát tốt cho dữ liệu mới.
- Nguyên nhân có thể bao gồm:
  * Cấu trúc mô hình quá phức tạp so với lượng dữ liệu huấn luyện.
  * Thiếu các kỹ thuật regularization (dropout, L2, early stopping).
  * Quy trình tiền xử lý dữ liệu giữa tập huấn luyện và kiểm tra chưa đồng nhất.

Chương 4

Đóng gói và thử nghiệm

4.1 Bài toán ứng dụng

4.1.1 Mô tả bài toán

Mục tiêu là dự báo giá của một loại tiền điện tử (ví dụ: Bitcoin) trong khoảng 1 ngày tới bằng cách sử dụng dữ liệu thị trường lịch sử và các chỉ báo kỹ thuật. Giá hoặc xu hướng được dự báo sẽ được dùng để tạo ra các tín hiệu giao dịch cụ thể (mua, bán, giữ).

Hỗ trợ ra quyết định giao dịch ngắn hạn

Mục tiêu chính của mô hình là cung cấp một dự báo về giá đóng cửa điều chỉnh của Bitcoin cho ngày giao dịch tiếp theo (T+1). Thông tin này có thể được các nhà đầu tư cá nhân hoặc các nhà phân tích sử dụng để:

- **Xác định xu hướng tiềm năng:** So sánh giá dự đoán (T+1) với giá đóng cửa hiện tại (T) hoặc các mức giá quan trọng khác (ví dụ: trung bình trượt) để nhận định về khả năng tăng, giảm hoặc đi ngang của giá Bitcoin trong phiên tới.
- **Cân nhắc điểm vào/ra lệnh:**
  - Nếu mô hình dự đoán giá T+1 cao hơn đáng kể so với giá T, đây có thể là một tín hiệu để cân nhắc mở vị thế mua (long) hoặc giữ vị thế mua hiện có.
  - Ngược lại, nếu giá T+1 dự đoán thấp hơn đáng kể, nhà đầu tư có thể cân nhắc việc bán (short), chốt lời vị thế mua, hoặc tránh mở vị thế mua mới.
  - Nếu chênh lệch không lớn, có thể là tín hiệu để giữ vị thế hiện tại hoặc chờ đợi thêm các xác nhận khác.

4.1.2 Người dùng và ứng dụng

- **Nhà đầu tư cá nhân:** Sử dụng dự báo giá để tối ưu hoá quyết định mua/bán.
- **Nhóm giao dịch thuật toán:** Tự động hóa chiến lược giao dịch dựa trên đầu ra mô hình.
- **Chuyên viên phân tích tài chính:** Tích hợp dự báo vào chiến lược đầu tư tổng thể.
- **Nhóm quản trị rủi ro:** Theo dõi dự báo để giảm thiểu tổn thất.

4.1.3 Các mô hình được áp dụng

- **Mô hình truyền thống**
  - **ARIMA:** Mô hình chuỗi thời gian nắm bắt xu hướng và tính mùa vụ.
  - **Hồi quy tuyến tính, Lasso, Hồi quy đa thức:** Mô hình hoá mối quan hệ tuyến tính và phi tuyến.
- **Mô hình học máy**
  - **SVR (Hồi quy vectơ hỗ trợ):** Xử lý dữ liệu phi tuyến hiệu quả.
  - **Random Forest:** Kháng nhiễu tốt và hỗ trợ chọn lọc đặc trưng.
- **Mô hình học sâu**
  - **LSTM:** Nắm bắt được mối quan hệ dài hạn trong chuỗi thời gian.
  - **GRU:** Phiên bản hiệu quả và gọn nhẹ hơn của LSTM.

4.1.4 Triển khai và đầu ra

- Dự báo giá tại bước thời gian tiếp theo (1 ngày).
- Tạo tín hiệu giao dịch dựa trên xu hướng dự báo.
- Hiển thị độ tin cậy của dự báo và cảnh báo biến động.

4.2 Các chỉ số đánh giá

- **MAE:** Nếu MAE thấp hơn một ngưỡng nhất định (ví dụ $<$ \$1500 với BTC), mô hình có thể ứng dụng được.
- **RMSE:** Nếu RMSE thấp hơn một ngưỡng nhất định (ví dụ $<$ \$2000 với BTC), mô hình có thể ứng dụng được.
- **MAPE**
  - **MAPE $<$ 5%**: Mô hình rất tốt
  - **MAPE từ 5–10%**: Mô hình chấp nhận được
  - **MAPE $>$ 10%**: Cần cải thiện trước khi triển khai
- **R-squared (R^2):**
  - $R^2 > 0.9$: Tuyệt vời
  - $0.8 - 0.9$: Tốt
  - < 0.5: Không đủ điều kiện triển khai

	Linear	Lasso	Polynomial	ARIMA	SVR	Random Forest	LSTM	GRU
RMSE	1974.59	**1970.59**	1982.89	1971.3	4050.4	3107.3	4015.62	4704.74
MAE	1393.98	1388.66	1407.12	**1382.27**	2755.55	2356.34	3045.34	3667.24
R2	**0.9891**	**0.9891**	0.9892	0.989	0.9549	0.9567	0.9537	0.9365
MAPE	**1.98%**	**1.98%**	2.01%	**1.98%**	3.40%	3.03%	4.38%	4.88%

Hình 4.1: Kết quả huấn luyện của 8 mô hình được xây dựng

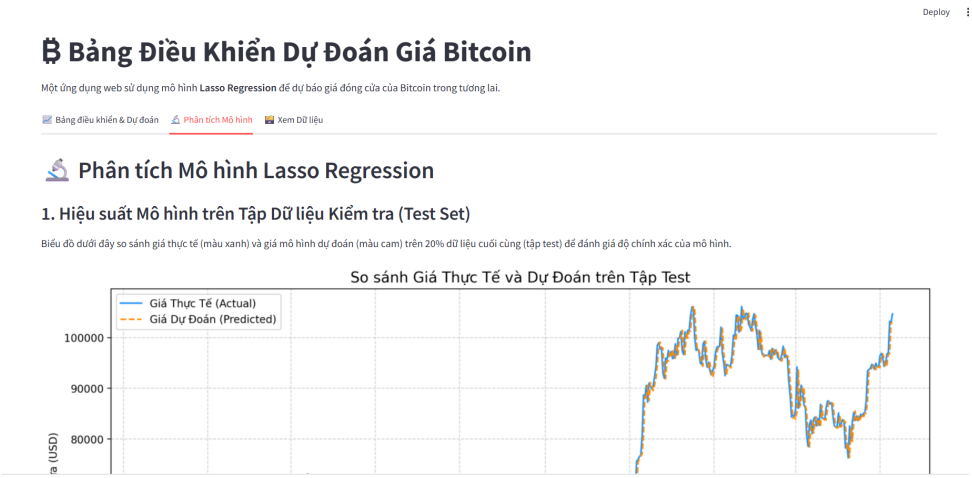
4.3 Xây dựng chương trình

4.3.1 Mô tả giao diện và chức năng

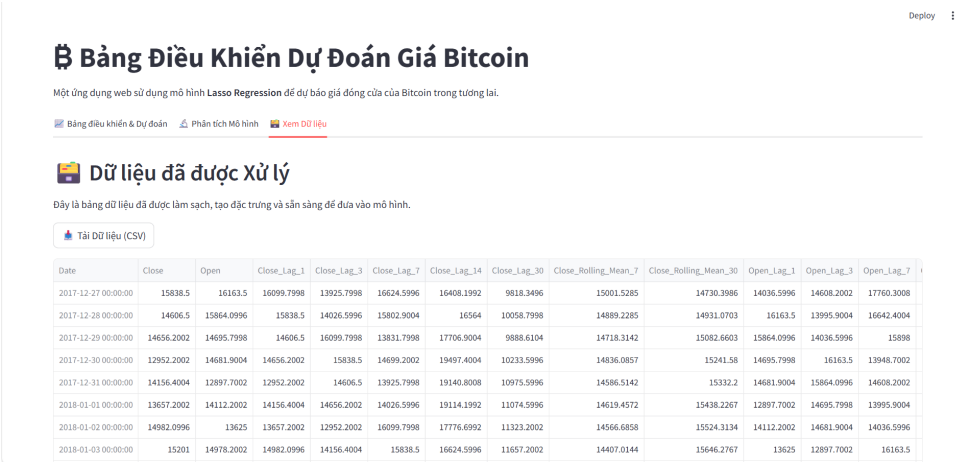
- **Tổng quan:** Giao diện được thiết kế theo dạng thẻ (tabs) để phân chia rõ ràng các khu vực chức năng, giúp người dùng dễ dàng điều hướng và tương tác. Ứng dụng bao gồm 3 thẻ chính.
- **Thẻ 1: Bảng điều khiển & Dự đoán**
  - Đây là màn hình tương tác chính của ứng dụng, được chia thành hai cột:
  - **Cột trái - Tùy chọn Dự đoán:**
    - `Chọn ngày muốn dự đoán:` Một ô nhập liệu ngày tháng cho phép người dùng chọn một ngày cụ thể trong tương lai để mô hình đưa ra dự báo.
    - `Số ngày lịch sử hiển thị trên biểu đồ:` Một thanh trượt (slider) cho phép người dùng tùy chỉnh khoảng thời gian của dữ liệu lịch sử sẽ được hiển thị trên biểu đồ, giúp quan sát xu hướng trong ngắn hạn hoặc dài hạn.
    - `Nút "Chạy Dự Đoán":` Nút bấm chính để kích hoạt quá trình tính toán và hiển thị kết quả dự đoán.
    - `Thông tin cập nhật:` Một hộp thông báo cho biết ngày cuối cùng của bộ dữ liệu được sử dụng để huấn luyện mô hình.
  - **Cột phải - Kết quả & Biểu đồ:**
    - `Các chỉ số (Metrics):` Hiển thị hai chỉ số quan trọng:
      - Giá trị dự đoán cho ngày đã chọn, được định dạng theo đơn vị tiền tệ (USD).
      - So sánh giá trị dự đoán với giá trị thực tế của ngày cuối cùng trong dữ liệu, đi kèm với mức chênh lệch (delta) tăng hoặc giảm.
    - `Biểu đồ Lịch sử và Dự đoán:`
      - Trực quan hóa dữ liệu giá đóng cửa lịch sử (đường màu xanh).
      - Vẽ đường xu hướng dự đoán cho các ngày trong tương lai (đường đứt nét màu cam).
      - Đánh dấu điểm dự đoán cụ thể cho ngày người dùng đã chọn (chấm tròn màu đỏ) để làm nổi bật kết quả.

- **Thẻ 2: Phân tích Mô hình**
  - Cung cấp cái nhìn sâu hơn về hiệu suất và cách hoạt động của mô hình `Lasso Regression`.
  - `Hiệu suất trên Tập Dữ liệu Kiểm tra (Test Set):`
    - Hiển thị một biểu đồ đường so sánh giữa giá trị thực tế (Actual) và giá trị do mô hình dự đoán (Predicted) trên 20% dữ liệu cuối cùng (tập test).
    - Biểu đồ này giúp người dùng đánh giá mức độ chính xác và độ bám sát của mô hình so với thực tế.
  - `Mức độ Quan trọng của Đặc trưng (Feature Importances):` (Không hiển thị đầy đủ trong ảnh chụp nhưng là một phần của chức năng)
    - Trình bày một biểu đồ cột cho thấy hệ số (coefficient) của từng đặc trưng.
    - Giúp người dùng hiểu được những yếu tố nào (ví dụ: giá đóng cửa 7 ngày trước, giá mở cửa 30 ngày trước) có ảnh hưởng lớn nhất đến kết quả dự đoán của mô hình.
- **Thẻ 3: Xem Dữ liệu**
  - Cho phép người dùng xem và tương tác trực tiếp với bộ dữ liệu đã qua xử lý.
  - `Nút "Tải Dữ liệu (CSV)":` Cung cấp chức năng cho phép người dùng tải về máy bộ dữ liệu đã được làm sạch và tạo đặc trưng dưới định dạng file `.csv`.
  - `Bảng dữ liệu (DataFrame):` Hiển thị một bảng tương tác chứa toàn bộ dữ liệu được sử dụng cho việc huấn luyện và dự đoán, bao gồm các cột gốc và các cột đặc trưng được tạo ra (ví dụ: `Close_Lag_1`, `Close_Rolling_Mean_7`, v.v.).

Hình 4.2: Giao diện chương trình



Hình 4.3: Giao diện chương trình



Date	Close	Open	Close_Lag_1	Close_Lag_3	Close_Lag_7	Close_Lag_14	Close_Lag_30	Close_Rolling_Mean_7	Close_Rolling_Mean_30	Open_Lag_1	Open_Lag_3	Open_Lag_7
2017-12-27 00:00:00	15838.5	16163.5	16099.7998	13925.7998	16624.5996	16408.1992	9818.3496	15001.5285	14730.3986	14036.5996	14608.2002	17760.3008
2017-12-28 00:00:00	14606.5	15864.0996	15838.5	14026.5996	15802.9004	16564	10058.7998	14889.2285	14931.0703	16163.5	13995.9004	16642.4004
2017-12-29 00:00:00	14656.2002	14695.7998	14606.5	16099.7998	13831.7998	17706.9004	9888.6104	14718.3142	15082.6603	15864.0996	14036.5996	15898
2017-12-30 00:00:00	12952.2002	14681.9004	14656.2002	15838.5	14699.2002	19497.4004	10233.5996	14836.0857	15241.58	14695.7998	16163.5	13948.7002
2017-12-31 00:00:00	14156.4004	12897.7002	12952.2002	14606.5	13925.7998	19140.8008	10975.5996	14586.5142	15332.2	14681.9004	15864.0996	14608.2002
2018-01-01 00:00:00	13657.2002	14112.2002	14156.4004	14656.2002	14026.5996	19114.1992	11074.5996	14619.4572	15438.2267	12897.7002	14695.7998	13995.9004
2018-01-02 00:00:00	14982.0996	13625	13657.2002	12952.2002	16099.7998	17776.6992	11323.2002	14566.6858	15524.3134	14112.2002	14681.9004	14036.5996
2018-01-03 00:00:00	15201	14978.2002	14982.0996	14156.4004	15838.5	16624.5996	11657.2002	14407.0144	15646.2767	13625	12897.7002	16163.5

Hình 4.4: Giao diện chương trình

4.3.2 Quy trình hoạt động

1. **Xác định Ngày Mục Tiêu (Target Date):** Ứng dụng nhận ngày người dùng muốn dự đoán (gọi là `Target_Date`).
2. **Chuẩn bị Dữ liệu Lịch sử:**
   - Nếu `Target_Date` ở tương lai: Lấy toàn bộ dữ liệu lịch sử (`df_processed`) làm cơ sở.
   - Nếu `Target_Date` ở quá khứ: Kiểm tra xem có dữ liệu cho ngày đó không.
3. **Tính toán Đặc trưng Lặp (Iterative Feature Calculation - nếu `Target_Date` ở tương lai):**
   - Lặp từ ngày sau ngày dữ liệu cuối cùng (`Last_Known_Date`) đến `Target_Date`:
     (a) **Tạo Đặc trưng (Feature Engineering):** Dựa trên chuỗi giá trị `Close` và `Open` hiện có (bao gồm cả các dự đoán trước đó), tính toán các đặc trưng (lags, rolling means) cho ngày dự đoán hiện tại (`Current_Prediction_Date`).
     (b) **Chuẩn hóa Đặc trưng (Scaling):** Các đặc trưng vừa tạo được chuẩn hóa bằng đối tượng `scaler` đã được huấn luyện.

(c) **Dự đoán (Prediction):** Vector đặc trưng đã chuẩn hóa được đưa vào mô hình `Lasso` đã huấn luyện.

(d) **Cập nhật Dữ liệu:** Mô hình đưa ra giá trị `Close` dự đoán cho `Current_Prediction_Date`. Giá `Open` được giả định bằng giá `Close` này. Cặp giá trị này được thêm vào chuỗi dữ liệu để sử dụng cho vòng lặp tiếp theo.

4. **Lấy Kết quả Cuối cùng:**

- Nếu `Target_Date` ở tương lai: Giá trị dự đoán cho `Target_Date` là kết quả từ bước 3.d của vòng lặp cuối cùng.
- Nếu `Target_Date` ở quá khứ và có dữ liệu: Lấy giá trị thực tế từ `df_processed`.

5. **Hiển thị Kết quả:**

- Giá trị dự đoán (hoặc thực tế) được hiển thị trên giao diện.
- Các đặc trưng đã sử dụng để đưa ra dự đoán cho `Target_Date` (hoặc ngày trước đó nếu là giá thực tế) được hiển thị.
- Biểu đồ được cập nhật để bao gồm đường lịch sử, đường dự đoán trên tập test, và (nếu có) đường dự đoán tương lai cùng với điểm đánh dấu cho `Target_Date`.

Chương 5

Kết luận

5.1 Tóm tắt các kết quả chính

Qua quá trình thực hiện dự án, từ bước tiền xử lý dữ liệu, tạo đặc trưng, lựa chọn đặc trưng, đến huấn luyện và đánh giá nhiều mô hình học máy, chúng tôi đã thu được các kết quả đáng chú ý như sau:

- **Xây dựng thành công quy trình dự đoán:** Một quy trình hoàn chỉnh từ xử lý dữ liệu thô đến xây dựng và đánh giá mô hình dự đoán giá đóng cửa tiền điện tử đã được thiết kế và triển khai hiệu quả.
- **Hiệu quả của kỹ thuật và lựa chọn đặc trưng:** Việc tạo ra 20 đặc trưng từ dữ liệu lịch sử giá Bitcoin (bao gồm đặc trưng trễ, trung bình trượt, độ biến động, tỷ lệ thay đổi, RSI và đặc trưng ngày tháng) đã cung cấp thêm thông tin có giá trị so với việc chỉ dùng một đặc trưng trễ duy nhất. Quy trình lựa chọn đặc trưng dựa trên tầm quan trọng của Random Forest đã chọn ra 15 đặc trưng tối ưu, góp phần nâng cao hiệu suất dự đoán và giảm độ phức tạp tính toán.
- **Xác định mô hình có hiệu quả cao nhất:** Trong 9 mô hình được xây dựng, mỗi chương trình cho ra kết quả khác nhau và mỗi mô hình đạt được các chỉ số khác nhau, mỗi chỉ số tìm được những mô hình tốt nhất theo từng chỉ số, nhưng để chọn ra mô hình tốt nhất thì mô hình Lasso Regression là tốt nhất.
  - **RMSE:** 1970.59
  - **MAE:** 1388.66
  - R^2**:** 0.9891
  - **MAPE:** 1.95%

5.2 Đóng góp của đề tài

Nghiên cứu này mang lại một số đóng góp đáng chú ý trong lĩnh vực dự đoán giá Bitcoin:

- Hệ thống hóa và đánh giá hiệu suất của mô hình hồi quy Lasso (Lasso Regression) trong việc dự đoán giá Bitcoin, xem xét các yếu tố như lựa chọn tham số alpha và kỹ thuật tạo đặc trưng dựa trên dữ liệu lịch sử giá mở cửa và đóng cửa.

- Làm nổi bật vai trò quan trọng của kỹ thuật tạo đặc trưng (feature engineering), bao gồm việc sử dụng các giá trị trễ (lags) và trung bình trượt (rolling means) của giá, trong việc cải thiện độ chính xác và khả năng diễn giải của mô hình dự đoán giá Bitcoin.
- Đề xuất và triển khai một mô hình dự đoán giá Bitcoin sử dụng Lasso Regression, đạt được hiệu suất tốt (ví dụ, có thể nêu một chỉ số cụ thể nếu có, như $R^2 >$ giá trị hoặc MAPE $<$ giá trị%), có tiềm năng ứng dụng làm công cụ tham khảo trong việc đưa ra nhận định về xu hướng giá.
- Xây dựng một giao diện ứng dụng web tương tác (sử dụng Streamlit) cho phép người dùng dễ dàng thực hiện dự đoán giá Bitcoin cho một ngày cụ thể, trực quan hóa kết quả và hiểu rõ hơn về các đặc trưng đầu vào của mô hình, từ đó tăng tính minh bạch và khả năng tiếp cận của mô hình.
- Phân tích các hệ số của mô hình Lasso, giúp xác định các đặc trưng có ảnh hưởng lớn nhất đến giá Bitcoin, qua đó cung cấp cái nhìn sâu sắc hơn về các yếu tố động lực giá trong ngắn hạn dựa trên dữ liệu lịch sử.

5.3 Hạn chế của nghiên cứu

Mặc dù nghiên cứu đã đạt được những kết quả nhất định, vẫn tồn tại một số hạn chế cần được xem xét:

- **Tính biến động cao của thị trường Bitcoin:** Giá Bitcoin chịu ảnh hưởng bởi nhiều yếu tố phức tạp và khó lường (tin tức, quy định, tâm lý thị trường), khiến việc dự đoán chính xác trong dài hạn trở nên thách thức. Mô hình hiện tại chủ yếu dựa trên dữ liệu lịch sử giá.
- **Độ trễ của đặc trưng:** Các đặc trưng dựa trên giá quá khứ (lags, rolling means) có thể không phản ánh kịp thời các thay đổi đột ngột trên thị trường.
- **Giới hạn của mô hình Lasso:** Mặc dù Lasso có khả năng lựa chọn đặc trưng, nó vẫn là một mô hình tuyến tính và có thể không nắm bắt được các mối quan hệ phi tuyến phức tạp trong dữ liệu giá Bitcoin.
- **Phạm vi dữ liệu và đặc trưng:** Nghiên cứu chỉ tập trung vào dữ liệu giá 'Open' và 'Close'. Việc tích hợp thêm các nguồn dữ liệu khác (ví dụ: khối lượng giao dịch, chỉ số kỹ thuật, dữ liệu on-chain, tin tức/sentiment) có thể cải thiện hiệu suất mô hình.
- **Dự đoán lặp và tích lũy lỗi:** Khi dự đoán cho các ngày ở tương lai xa, việc sử dụng các giá trị dự đoán trước đó làm đầu vào có thể dẫn đến tích lũy lỗi, làm giảm độ chính xác của các dự đoán xa hơn.

5.4 Hướng phát triển tương lai

Để tiếp tục cải thiện và mở rộng nghiên cứu này, một số hướng phát triển tiềm năng có thể được xem xét:

- **Tích hợp đa dạng nguồn dữ liệu:** Mở rộng tập đặc trưng bằng cách kết hợp thêm các loại dữ liệu khác như:

  - **Dữ liệu On-chain:** Số lượng địa chỉ hoạt động, khối lượng giao dịch trên blockchain, dòng tiền ra/vào các sàn giao dịch.
  - **Dữ liệu thị trường phái sinh:** Khối lượng hợp đồng tương lai, tỷ lệ long/short.
  - **Phân tích Sentiment:** Trích xuất tâm lý thị trường từ tin tức, mạng xã hội (ví dụ: Twitter, Reddit) bằng các kỹ thuật xử lý ngôn ngữ tự nhiên (NLP).
  - **Các chỉ số kinh tế vĩ mô:** Lãi suất, lạm phát, các chỉ số thị trường chứng khoán toàn cầu có thể có tương quan với giá Bitcoin.

- **Ứng dụng các mô hình học máy và học sâu tiên tiến hơn:**
  - **Mô hình phi tuyến:** Thử nghiệm các mô hình có khả năng nắm bắt mối quan hệ phi tuyến tốt hơn như Support Vector Regression (SVR) với các kernel khác nhau, Random Forest, Gradient Boosting Machines (XGBoost, LightGBM).
  - **Mô hình học sâu cho chuỗi thời gian:** Khám phá các kiến trúc mạng nơ-ron như Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), hoặc Transformers, vốn được thiết kế để xử lý dữ liệu tuần tự.
  - **Mô hình kết hợp (Hybrid Models):** Kết hợp điểm mạnh của các mô hình khác nhau, ví dụ như kết hợp mô hình thống kê truyền thống với mô hình học sâu.

- **Tối ưu hóa siêu tham số và lựa chọn đặc trưng nâng cao:**
  - Sử dụng các kỹ thuật tối ưu hóa siêu tham số tự động (ví dụ: Bayesian Optimization, Optuna) để tìm ra bộ tham số tốt nhất cho mô hình.
  - Áp dụng các phương pháp lựa chọn đặc trưng phức tạp hơn để xác định tập con đặc trưng tối ưu nhất.

- **Đánh giá mô hình trong các điều kiện thị trường khác nhau:** Phân tích hiệu suất của mô hình trong các giai đoạn thị trường cụ thể (ví dụ: thị trường tăng giá mạnh, giảm giá mạnh, đi ngang) để hiểu rõ hơn về độ tin cậy và khả năng thích ứng của mô hình.

- **Phát triển chiến lược giao dịch dựa trên dự đoán:** Xây dựng và kiểm thử (backtesting) các chiến lược giao dịch đơn giản dựa trên tín hiệu từ mô hình dự đoán, đồng thời xem xét các yếu tố quản lý rủi ro.

- **Cải thiện giao diện người dùng và tính năng ứng dụng:**
  - Cho phép người dùng tùy chỉnh các tham số của mô hình hoặc lựa chọn các đặc trưng đầu vào.
  - Cung cấp các cảnh báo hoặc thông báo dựa trên dự đoán.
  - Tích hợp khả năng so sánh hiệu suất của nhiều mô hình khác nhau.

Tài liệu tham khảo

[1] Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

[2] Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press.

[3] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[4] Montgomery, D. C., Jennings, C. L., Kulahci, M. (2015). *Introduction to Time Series Analysis and Forecasting*. Wiley.

[5] Tsay, R. S. (2005). *Analysis of Financial Time Series*. Wiley.

[6] Khashei, M., Bijari, M. (2011). A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. *Applied Soft Computing*, **11**(2), 2664–2675.

[7] Fischer, B. E., Goldberg, A. P. (1993). Trading an asset with a gold-price-contingent call option. *The Journal of Derivatives*, **1**(1), 75–80.

[8] Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, **50**, 159–175.

[9] Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

[10] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, **30**.

[11] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*, **31**.

[12] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, **9**(8), 1735–1780.

[13] Streamlit Inc. (2023). *Streamlit: The fastest way to build and share data apps*. https://streamlit.io.