BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH

TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ

📐📖✍️

ĐỒ ÁN MÔN HỌC

ĐỀ TÀI:

Dự đoán giá **Bitcoin** với các kỹ thuật học máy và học sâu

Học phần: Học máy

Nhóm Sinh Viên:

1. ĐỒNG ĐAN HOÀI
2. NGUYỄN PHÚC HẢI
3. VŨ NGUYỄN NĂNG KHÁNH
4. NGUYỄN MINH NHẬT

Chuyên Ngành: KHOA HỌC DỮ LIỆU

Khóa: K46

Giảng Viên: TS. Đặng Ngọc Hoàng Thành

TP. Hồ Chí Minh, Ngày 23 tháng 04 năm 2023

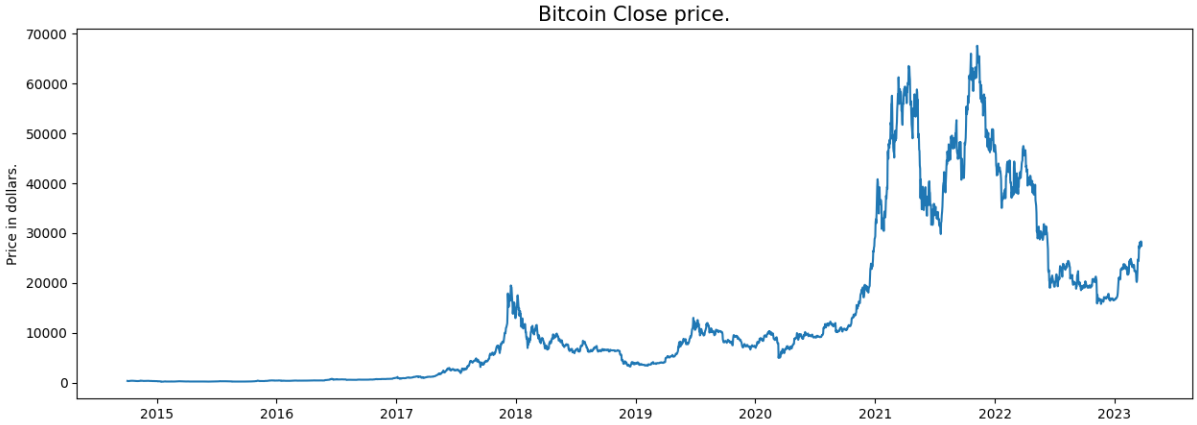
Mục lục

CHƯƠNG 1. TỔNG QUAN

1.1. Giới thiệu đề tài

Chuyển đổi kỹ thuật số [1] của các nền kinh tế là sự gián đoạn nghiêm trọng nhất hiện đang diễn ra ở tất cả các nền kinh tế và hệ thống tài chính. Các nền kinh tế và hệ thống tài chính trên thế giới đang trở thành kỹ thuật số với tốc độ nhanh chưa từng có. Theo một báo cáo, quy mô của nền kinh tế kỹ thuật số vào năm 2025 được ước tính là 25% (23 nghìn tỷ USD), bao gồm các tài sản kỹ thuật số hữu hình và vô hình. Công nghệ mới nhất để thiết lập và chi tiêu tài sản kỹ thuật số là công nghệ sổ cái phân tán (DLT) và ứng dụng nổi tiếng nhất của nó là tiền điện tử có tên Bitcoin. Sau những phát triển này, công nghệ chuỗi khối đã tìm thấy vị trí của mình trong các giao điểm của Fintech và các mạng thế hệ tiếp theo.

Một vấn đề quan trọng về các tài sản kỹ thuật số vô hình, và đặc biệt là tiền điện tử, là sự biến động giá cả. Giá của Bitcoin (BTC) trong khoảng thời gian từ ngày 3 tháng 10 năm 2014 đến ngày 24 tháng 3 năm 2023, có thể được xem trong Hình 1. Giá Bitcoin đã thể hiện sự biến động cực độ trong khoảng thời gian này. Từ năm 2017 đến trước năm 2021 giá chỉ giao động từ 0-20000 USD, sau đó đã có sự thay đổi đáng kể khi đã đạt đỉnh ở giá trị hơn 60000 USD ở 2 thời điểm đầu năm 2021 và cuối năm 2021. Tuy nhiên, đến đầu năm 2023 mức giá lại có xu hướng giảm đi một nửa đến khoảng 30000 USD. Trước năm 2013, mối quan tâm phổ biến đối với BTC, việc sử dụng nó trong các giao dịch ảo và giá của nó ở mức thấp. Khoảng thời gian đó không được xem xét trong các mô hình của nhóm. Mặc dù giá Bitcoin thể hiện sự biến động bất thường, nhưng Bitcoin với tư cách là một tài sản kỹ thuật số khá linh hoạt vì nó có thể lấy lại giá trị sau khi giảm đáng kể và ngay cả khi sự không chắc chắn cao trên thị trường, chẳng hạn như trong đại dịch COVID-19 [5].



Hình 1: Giá đóng cửa của Bitcoin từ 2014 đến 2023

1.2. Mục tiêu nghiên cứu

Từ những dữ liệu lịch sử của giá Bitcoin, nhóm hướng tới việc phân tích xu hướng và các yếu tố liên quan theo thời gian nhằm tạo ra công cụ dự báo giá của bitcoin trong một tương lai gần giúp cho người tham gia thị trường có cơ sở đưa ra quyết định thực hiện những giao dịch.

1.3. Phương pháp thực hiện

a. Nguồn dữ liệu

Dữ liệu các thuộc tính và giá của Bitcoin là có sẵn và được truy cập miễn phí trên website Yahoo Finance (URL: Bitcoin USD (BTC-USD) Price History & Historical Data - Yahoo Finance). Nó được lấy bằng cách truy cập vào trang lịch sử giá của đồng Bitcoin ở trên và sau đó tải xuống khoảng thời gian mong muốn của người thực hiện.

b. Phương pháp được sử dụng

Trong nghiên cứu này, nhóm thực hiện dự báo chuỗi thời gian bằng việc huấn luyện những mô hình máy học hồi quy truyền thống Decision Tree và ARIMA và học sâu Long-Short Term Memory(LSTM) để dự đoán được giá trị tương lai của tiền bitcoin trong một giai đoạn thời gian và đánh giá các chỉ số xác định tính chính xác của từng mô hình.

c. Công cụ thực hiện chương trình

Chương trình được tiến hành toàn bộ bằng ngôn ngữ lập trình Python trên môi trường Colab được Google cung cấp, và các thư viện pandas, numpy, matplotlib, seaborn, scikit-learn, …
Trong đó pandas, numpy, matplotlib, seaborn được dùng để thao tác các thuộc tính trên bộ dữ liệu, scikit-learn được dùng để tiền xử lý dữ liệu và huấn luyện mô hình.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Bài toán hồi quy giá tiền mã hóa

2.1.1. Giới thiệu về tiền mã hóa Bitcoin

a. Bitcoin là gì?

Bitcoin (BTC) [2] là một loại tiền điện tử, một loại tiền ảo được thiết kế để hoạt động như tiền và một hình thức thanh toán ngoài tầm kiểm soát của bất kỳ cá nhân, nhóm hoặc tổ chức nào, do đó loại bỏ nhu cầu tham gia của bên thứ ba vào các giao dịch tài chính. Nó được thưởng cho những người khai thác chuỗi khối vì đã hoàn thành công việc xác minh các giao dịch và có thể được mua trên một số sàn giao dịch.

Bitcoin được giới thiệu ra công chúng vào năm 2009 bởi một nhà phát triển ẩn danh hoặc một nhóm các nhà phát triển sử dụng tên Satoshi Nakamoto.

Kể từ đó, nó đã trở thành loại tiền điện tử nổi tiếng nhất trên thế giới. Sự phổ biến của nó đã truyền cảm hứng cho sự phát triển của nhiều loại tiền điện tử khác. Những đối thủ cạnh tranh này cố gắng thay thế nó như một hệ thống thanh toán hoặc được sử dụng làm mã thông báo tiện ích hoặc bảo mật trong các chuỗi khối khác và các công nghệ tài chính mới nổi.

b. Thông tin về bitcoin

Phần thưởng bitcoin giảm một nửa sau mỗi 210.000 khối. Ví dụ: phần thưởng khối là 50 bitcoin mới vào năm 2009. Vào ngày 11 tháng 5 năm 2020, đợt halving thứ ba đã xảy ra, đưa phần thưởng cho mỗi lần khám phá khối xuống còn 6,25 bitcoin.

c. Công nghệ blockchain của bitcoin

Tiền điện tử là một phần của blockchain và mạng cần thiết để cung cấp năng lượng cho nó. Blockchain là một sổ cái phân tán, một cơ sở dữ liệu dùng chung lưu trữ dữ liệu. Dữ liệu trong chuỗi khối được bảo mật bằng các phương thức mã hóa.

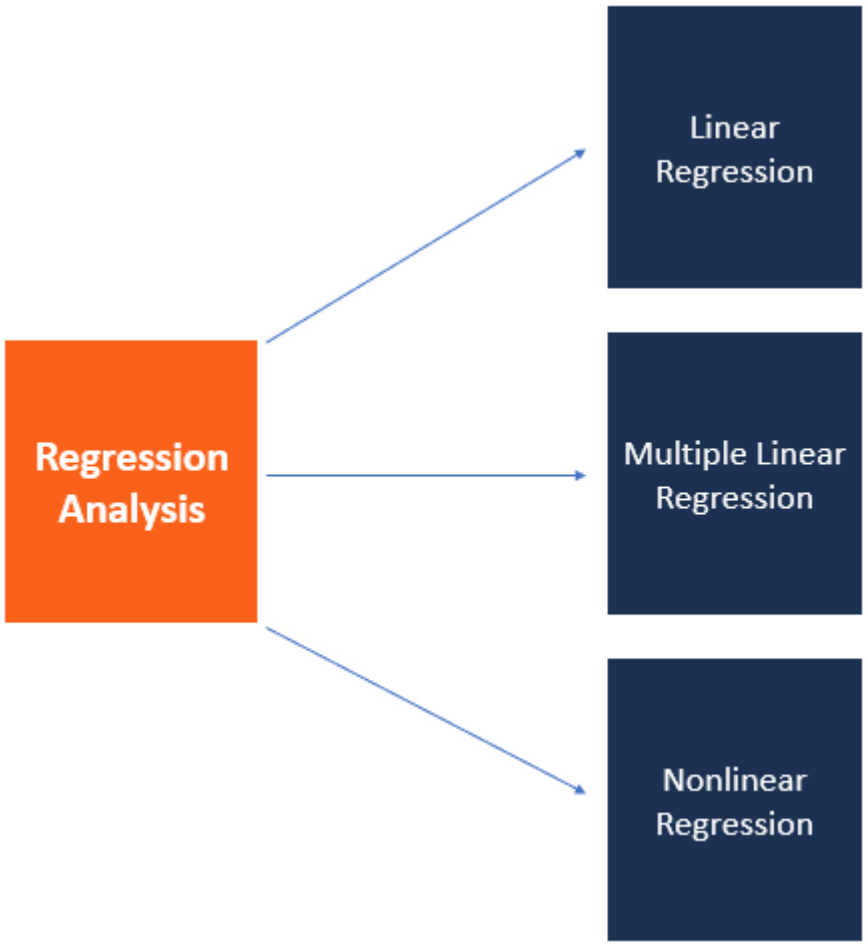
Khi một giao dịch diễn ra trên chuỗi khối, thông tin từ khối trước đó sẽ được sao chép sang một khối mới với dữ liệu mới, được mã hóa và giao dịch được xác minh bởi các trình xác thực—được gọi là thợ đào—trong mạng. Khi một giao dịch được xác minh, một block mới sẽ được mở và Bitcoin được tạo và trao làm phần thưởng cho (những) người khai thác đã xác minh dữ liệu trong block — sau đó họ có thể tự do sử dụng, giữ hoặc bán nó.

Bitcoin sử dụng thuật toán băm SHA-256 để mã hóa dữ liệu được lưu trữ trong blocks trên blockchain. Nói một cách đơn giản, dữ liệu giao dịch được lưu trữ trong một block được mã hóa thành số thập lục phân 256 bit. Con số đó chứa tất cả dữ liệu giao dịch và thông tin được liên kết với các blocks trước block đó.

2.2.2. Hồi quy giá tiền mã hóa

a. Phân tích hồi quy là gì?

Phân tích hồi quy [3] là một tập hợp các phương pháp thống kê được sử dụng để ước tính mối quan hệ giữa một biến phụ thuộc và một hoặc nhiều biến độc lập. Nó có thể được sử dụng để đánh giá sức mạnh của mối quan hệ giữa các biến và để mô hình hóa mối quan hệ trong tương lai giữa chúng.



Hình 2: Các kỹ thuật phân tích hồi quy

Phân tích hồi quy bao gồm một số biến thể, chẳng hạn như tuyến tính, đa tuyến tính và phi tuyến tính. Các mô hình phổ biến nhất là tuyến tính đơn giản và đa tuyến tính. Phân tích hồi quy phi tuyến tính thường được sử dụng cho các tập dữ liệu phức tạp hơn, trong đó các biến phụ thuộc và biến độc lập thể hiện mối quan hệ phi tuyến tính.

b. Hồi quy phi tuyến tính (Nonlinear Regression)

Hồi quy phi tuyến tính [4] là một dạng phân tích hồi quy trong đó dữ liệu phù hợp với mô hình và sau đó được biểu thị dưới dạng hàm toán học. Hồi quy tuyến tính đơn giản liên hệ hai biến (X và Y) với một đường thẳng ($y = mx + b$), trong khi hồi quy phi tuyến liên hệ hai biến trong mối quan hệ phi tuyến tính (cong).

Mục tiêu của mô hình là làm cho tổng bình phương càng nhỏ càng tốt. Tổng bình phương là thước đo theo dõi khoảng cách mà các quan sát Y thay đổi so với hàm phi tuyến tính (cong) được sử dụng để dự đoán Y.

Nó được tính toán bằng cách đầu tiên tìm sự khác biệt giữa hàm phi tuyến được gán vào và mọi điểm Y của dữ liệu trong tập hợp. Sau đó, mỗi sự khác biệt được bình phương. Cuối cùng, tất cả các con số bình phương được cộng lại với nhau. Tổng của các số bình phương này càng nhỏ thì hàm càng phù hợp với các điểm dữ liệu trong tập hợp. Hồi quy phi tuyến tính sử dụng các hàm logarit, hàm lượng giác, hàm mũ, hàm lũy thừa, đường cong Lorenz, hàm Gaussian và các phương pháp khớp khác.

Mô hình hồi quy phi tuyến tính tương tự như mô hình hồi quy tuyến tính ở chỗ cả hai đều tìm cách theo dõi một phản hồi cụ thể từ một tập hợp các biến bằng đồ thị. Các mô hình phi tuyến tính phức tạp hơn các mô hình tuyến tính để phát triển vì chức năng được tạo thông qua một loạt các phép tính gần đúng (lặp đi lặp lại) có thể xuất phát từ thử và sai. Các nhà toán học sử dụng một số phương pháp đã được thiết lập, chẳng hạn như phương pháp Gauss-Newton và phương pháp Levenberg-Marquardt.

2.2. Phương pháp tiền xử lý dữ liệu

2.2.1. Chuẩn hóa dữ liệu

a. Xử lý missing values

- Quy nạp(imputation) [5]

Trong thống kê, quy nạp là quá trình thay thế dữ liệu bị thiếu bằng các giá trị được thay thế. Khi thay thế cho một điểm dữ liệu, nó được gọi là "quy nạp đơn vị"; khi thay thế cho một thành phần của một điểm dữ liệu, nó được gọi là "xác định vật phẩm".

Có ba vấn đề chính mà việc thiếu dữ liệu gây ra: dữ liệu bị thiếu có thể gây ra một lượng sai lệch đáng kể, khiến việc xử lý và phân tích dữ liệu trở nên khó khăn hơn và làm giảm hiệu quả. Bởi vì dữ liệu bị thiếu có thể tạo ra các vấn đề khi phân tích dữ liệu, nên việc quy nạp được coi là một cách để tránh những cạm bẫy liên quan đến việc xóa theo danh sách các trường hợp có giá trị bị thiếu. Điều đó có nghĩa là, khi một hoặc nhiều giá trị bị thiếu cho một trường hợp, hầu hết các gói thống kê mặc định loại bỏ bất kỳ trường hợp nào có giá trị bị thiếu, điều này có thể gây ra sai lệch hoặc ảnh hưởng đến tính đại diện của kết quả. Quy nạp bảo tồn tất cả các trường hợp bằng cách thay thế dữ liệu bị thiếu bằng giá trị ước tính dựa trên thông tin có sẵn khác. Sau khi tất cả các giá trị bị thiếu đã được xác định, tập dữ liệu sau đó có thể được phân tích bằng các kỹ thuật tiêu chuẩn để có dữ liệu hoàn chỉnh.

Đã có nhiều lý thuyết được các nhà khoa học chấp nhận để giải thích cho dữ liệu bị thiếu nhưng phần lớn trong số đó đưa ra sự sai lệch. Một số nỗ lực nổi tiếng để xử lý dữ liệu bị thiếu bao gồm: quy nạp boong nóng và boong lạnh; xóa theo danh sách và theo cặp; quy nạp trung bình; hệ số ma trận không âm; quy nạp hồi quy; lần quan sát cuối cùng được chuyển tiếp; quy nạp ngẫu nhiên; và nhiều quy nạp khác.

- Xóa (Listwise deletion) [6]

Trong thống kê, xóa theo danh sách là một phương pháp xử lý dữ liệu bị thiếu. Trong phương pháp này, toàn bộ bản ghi sẽ bị loại khỏi phân tích nếu thiếu bất kỳ giá trị đơn lẻ nào.

b. Chuẩn hóa dữ liệu(Feature Scaling)

Chuẩn hóa dữ liệu [7] là một phương pháp được sử dụng để chuẩn hóa phạm vi của các biến độc lập hoặc tính năng của dữ liệu. Trong xử lý dữ liệu, nó còn được gọi là chuẩn hóa dữ liệu(data normalization) và thường được thực hiện trong bước tiền xử lý dữ liệu. Một trong những phương pháp thực hiện đó là MinMaxScaler.

- MinMaxScaler [8]

Công cụ ước tính này chia tỷ lệ và dịch từng tính năng riêng lẻ sao cho nó nằm trong phạm vi nhất định trên tập huấn luyện, ví dụ: giữa không và một.

Phép biến đổi được cho bởi:

$$x_{std} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

$$x_{scaled} = x_{std} * (x_{max} - x_{min}) + x_{min}$$

- log scaling [9]

Thang logarit là một cách hiển thị dữ liệu số trên một phạm vi giá trị rất rộng theo cách nhỏ gọn. Trái ngược với một trục số tuyến tính, trong đó mỗi đơn vị khoảng cách tương ứng với việc thêm một lượng như nhau, trên thang logarit, mỗi đơn vị độ dài tương ứng với việc nhân giá trị trước đó với cùng một lượng. Do đó, thang đo như vậy là phi tuyến tính: các số 1, 2, 3, 4, 5, v.v., không cách đều nhau. Thay vào đó, các số 10, 100, 1000, 10000 và 100000 sẽ cách đều nhau. Tương tự như vậy, các số 2, 4, 8, 16, 32, v.v. sẽ cách đều nhau. Thông thường, các đường cong tăng trưởng theo cấp số nhân được hiển thị trên thang logarit, nếu không, chúng sẽ tăng quá nhanh để vừa với một biểu đồ nhỏ.

c. Khử nhiễu [10]

Sử dụng đường trung bình chuyển động (Moving Averages), nó là một trong số chỉ số kỹ thuật được sử dụng rộng rãi và linh hoạt nhất trong phân tích kỹ thuật (technical analysis) trong phân tích thị trường tài chính (hàng hóa và cổ phiếu). Đường trung bình chuyển động là cơ sở trong

nhiều hệ thống theo sau xu hướng (xu thế following) thị trường hiện nay. Đường trung bình chuyển động là trung bình giá của cổ phiếu hoặc chỉ số trong một khoảng thời gian nào đó.Ví dụ, để tính giá đóng cửa trung bình trong 10 ngày, giá của mười ngày được cộng lại và chia cho 10. Tính từ 'chuyển động' được sử dụng là bởi vì chỉ có giá của 10 ngày gần nhất được sử dụng. Bởi vậy giá trị trung bình của giá trong 10 ngày gần nhất sẽ thay đổi sau mỗi ngày giao dịch. Dạng tính toán thông dụng nhất của đường trung bình chuyển động là tính trung bình giá đóng cửa của 10 ngày. Mỗi ngày tiếp theo, giá đóng cửa mới được cộng vào tổng giá và giá cách đó 11 ngày được trừ ra khỏi tổng giá, sau đó tổng giá được chia cho 10

2.2.2. Loại bỏ yếu tố xu hướng/mùa vụ

a. Tính dừng của chuỗi thời gian

Một chuỗi thời gian có thể được coi là có tính dừng khi các thuộc tính thống kê như giá trị trung bình, phương sai, hiệp phương sai (tại các độ trễ khác nhau) có giá trị không đổi theo thời gian. Chuỗi dừng có xu hướng trở về giá trị trung bình và những dao động quanh giá trị trung bình sẽ là như nhau. Nói cách khác, một chuỗi thời gian không dừng sẽ có giá trị trung bình thay đổi theo thời gian, hoặc giá trị phương sai thay đổi theo thời gian hoặc cả hai.

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

b. Kiểm định ADF

Kiểm định Dickey và Fuller mở rộng (ADF) là một dạng kiểm định nghiệm đơn vị được sử dụng khá phổ biến để kiểm định một chuỗi thời gian là dừng hay không dừng. Cụ thể, theo Dickey và Fuller (1981) mô hình kiểm định nghiệm đơn vị mở rộng ADF có dạng:

$$\Delta y_t = \alpha_0 + \beta y_{t-1} + \sum_{j=1}^{k} \phi_j \Delta y_{t-j} + \varepsilon_t \ (1)$$

$$\Delta y_t = \alpha_0 + \delta_t + \beta y_{t-1} + \sum_{j=1}^{k} \phi_j \Delta y_{t-j} + \varepsilon_t \ (2)$$

Trong đó:

- $\Delta y_t = y_t - y_{t-1}$
- k: Chiều dài độ trễ
- ε_t: Nhiễu trắng

Mô hình (2) có sự khác biệt với mô hình (1) ở đặc điểm là bổ sung thêm biến xu hướng về thời gian t. Biến xu hướng có giá trị từ 1 đến n, với 1 đại diện cho quan sát đầu tiên và n đại diện cho quan sát cuối cùng trong chuỗi dữ liệu. Nhiễu trắng là số hạng chỉ sai số

ngẫu nhiên xuất phát từ các giả định cổ điển rằng nó có giá trị trung bình bằng 0, phương sai là hằng số và không tự tương quan.

Nghiên cứu sẽ tiến hành kiểm định trong cả hai trường hợp không có và có xu hướng về thời gian bằng cách sử dụng lần lượt các mô hình (1) và (2). Giả thuyết kiểm định được đặt ra như sau:

$$H_0: \beta = 0 \ (y_t \text{ là chuỗi dữ liệu không dừng})$$

$$H_1: \beta < 0 \ (y_t \text{ là chuỗi dữ liệu dừng})$$

Trong kiểm định ADF, giá trị kiểm định ADF không theo phân phối chuẩn. Theo Dickey và Fuller (1981) giá trị t ước lượng của các hệ số trong các mô hình (1) và (2) sẽ theo phân phối xác suất τ (τ = giá trị hệ số ước lượng/sai số của hệ số ước lượng). Giá trị tới hạn τ được xác định dựa trên bảng giá trị tính sẵn của Mackinnon (1996).

Để kiểm định giả thuyết H0 nghiên cứu so sánh giá trị kiểm định τ tính toán với giá trị τ tới hạn của Mackinnon và kết luận về tính dừng của các chuỗi quan sát.

Cụ thể, nếu trị tuyệt đối của giá trị tính toán lớn hơn trị tuyệt đối giá trị tới hạn thì giả thuyết H0 sẽ bị bác bỏ, tức chuỗi dữ liệu có tính dừng và ngược lại chấp nhận giả thuyết H0, tức dữ liệu không có tính dừng.

2.2.3. Trích chọn đặc trưng (Feature engineering)

a. Trung bình động giản đơn (SMA)

Đường trung bình động giản đơn có thể được tính như sau:

$$SMA = \frac{P_1 + P_2 + \ldots + P_n}{n}$$

Với P là mức giá đóng cửa trong mỗi chu kỳ, n là số ngày/phiên giao dịch, thể hiện chu kỳ biến động.

b. Chỉ số sức mạnh tương đối (RSI)

Chỉ số RSI là một chỉ báo kỹ thuật được sử dụng trong phân tích kỹ thuật đối với thị trường tài chính. RSI là chỉ báo kỹ thuật được sử dụng đo lường mức độ biến động giá của tiền mã hóa hay các loại tài sản khác để đánh giá tình trạng mua quá mức (quá mua) hoặc bán quá mức (quá bán). RSI được phân loại là một bộ dao động động lượng.

RSI có thể được tính với 2 bước như sau:

$$RSI_{Bước\ 1} = 100 - \left(\frac{100}{1 + \frac{Mức\ lãi\ trung\ bình}{Mức\ lỗ\ trung\ bình}} \right)$$

$$RSI_{Bước\ 2} = 100 - \left(\frac{100}{1 + \frac{Mức\ lãi\ trung\ bình\ trước\ đó \times 13 + Mức\ lãi\ hiện\ tại}{Mức\ lỗ\ trung\ bình\ trước\ đó \times 13 + Mức\ lỗ\ hiện\ tại} } \right)$$

Mức lãi hoặc lỗ trung bình được sử dụng trong công thức là phần trăm lãi hoặc lỗ trung bình trong một khoảng thời gian nhất định. Mức lỗ trung bình được sử dụng trong công thức này với giá trị dương. Các giai đoạn lỗ thì mức lãi trung bình sẽ được tính là 0, các giai đoạn lãi thì mức lỗ trung bình sẽ được tính là 0. Cách tính mặc định là sử dụng giá trị trong 14 phiên giao dịch để tính toán giá trị RSI ban đầu. Chỉ số RSI sẽ tăng khi số lượng và quy mô của các lần đóng cửa ở mức giá tăng tăng lên, và sẽ giảm khi số lượng và quy mô của các khoản lỗ tăng lên.

c. Chỉ báo dao động Stochastic

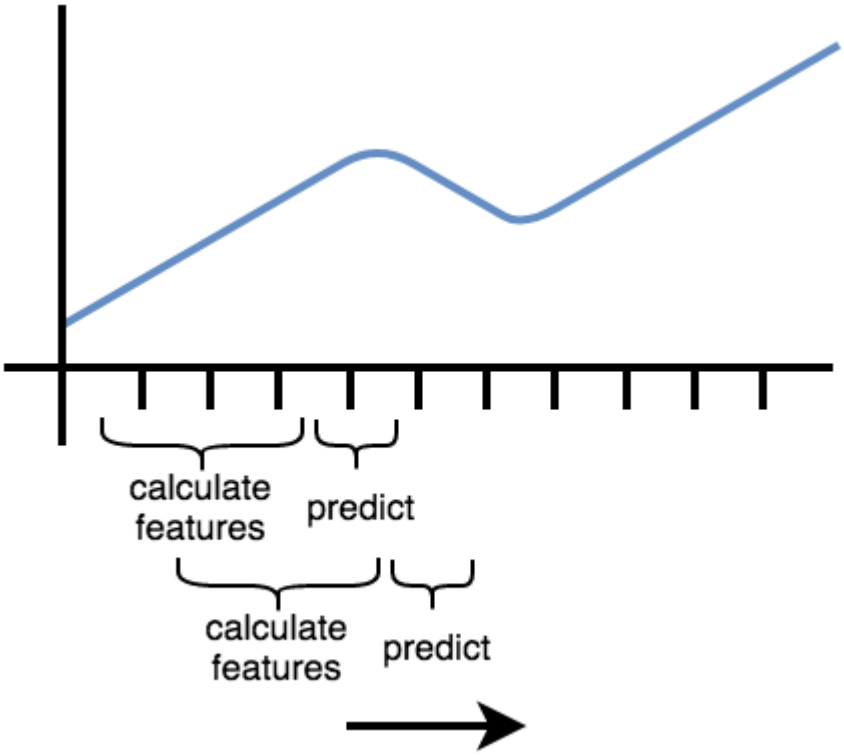
Chỉ báo dao động Stochastic thể hiện động lượng của giá bằng cách so sánh mức giá đóng cửa với một phạm vi giá trong một khoảng thời gian nhất định. Chỉ báo này được cấu tạo bởi 2 thành phần là đường %K và %D. Trong đó, %K là đường chính và %D là đường trung bình động 3 phiên của đường %K:

$$\%K = \left(\frac{C - L14}{H14 - L14} \right) \times 100$$

Trong đó C là mức giá đóng cửa gần nhất, L14 và H14 lần lượt là mức giá thấp nhất và cao nhất trong vòng 14 phiên.

2.2.4. Phân tích cửa sổ trượt (Sliding Window Analysis)

Phân tích luân phiên của mô hình chuỗi thời gian thường được sử dụng để đánh giá tính ổn định của mô hình theo thời gian. Khi phân tích dữ liệu tài chính dạng chuỗi thời gian bằng các mô hình thống kê, giả định chính là các tham số của mô hình không đổi theo thời gian. Tuy nhiên, môi trường kinh tế thường thay đổi đáng kể và có thể không hợp lý khi cho rằng các tham số của mô hình là không đổi. Một kỹ thuật phổ biến để đánh giá tính không đổi của các tham số của mô hình là sử dụng cửa sổ trượt (Sliding Window), qua đó các tham số ước tính sẽ được tính toán hoặc sử dụng trong phạm vi một cửa sổ với kích thước cố định thông qua việc lấy mẫu.



Hình 3: Minh họa phương pháp cửa sổ trượt (nguồn: https://tsfresh.readthedocs.io/en/latest/text/forecasting.html)

2.3. Mô hình hồi quy đề xuất

2.3.1. ARIMA

a. Giới thiệu mô hình ARIMA

ARIMA (Autoregressive Integrated Moving Average) [11] là một mô hình phân tích chuỗi thời gian kết hợp các kỹ thuật tự hồi quy, chênh lệch và trung bình trượt để mô hình hóa và dự báo dữ liệu chuỗi thời gian. Mô hình này được rộng rãi sử dụng trong lĩnh vực kinh tế lượng, tài chính và kỹ thuật để phân tích và dự đoán các hiện tượng chuỗi thời gian khác nhau.

Tự hồi quy (AR)(p): Trong thành phần này, giá trị hiện tại của chuỗi thời gian được mô hình hóa như một hàm của các giá trị trước đó của chuỗi, còn được gọi là độ trễ. Thứ tự của thành phần tự hồi quy, được ký hiệu bằng tham số p, xác định số lượng độ trễ được sử dụng trong mô hình. AR được định nghĩa là tổng trọng số của p thuật ngữ giá trị trước đó, mỗi thuật ngữ được nhân với một hệ số trọng số tương ứng:

$$AR = c + \phi_1 * Y_{t-1} + \phi_2 * Y_{t-2} + \ldots + \phi_p * Y_{t-p}$$

Trong đó:

- AR_t là giá trị của thành phần tự hồi quy tại thời điểm t.
- c là hằng số.
- Y_{t-1}, Y_{t-2}, ..., Y_{t-p} là các giá trị của chuỗi thời gian tại các thời điểm trước đó.

- ϕ_1, ϕ_2, ..., ϕ_p là các trọng số tương ứng với các giá trị trước đó.

Các hệ số trọng số φ được ước tính từ dữ liệu bằng cách sử dụng phương pháp tối ưu hóa mất mát như phương pháp cực đại hợp lý (maximum likelihood) hoặc phương pháp bình phương tối thiểu (least squares). Khi giá trị của p tăng, mô hình sẽ có độ phức tạp cao hơn và yêu cầu một lượng lớn dữ liệu để ước tính các hệ số trọng số chính xác.

Đồng tích hợp (I)(d): Thành phần này liên quan đến việc lấy sai phân dữ liệu chuỗi thời gian nhằm biến đổi nó thành chuỗi dừng. Sai phân là quá trình lấy sự khác biệt giữa các quan sát liên tiếp của chuỗi thời gian. Bậc sai phân, được ký hiệu bằng tham số d, xác định số lần dữ liệu phải được chênh lệch để đạt được tính ổn định. Giá trị I là sự khác biệt giữa giá trị hiện tại và giá trị trước đó sau khi lấy sai phân bậc d:

$$I_t = (1 - B)^D * Y_t$$

Trong đó:

- $I_t$là giá trị chênh lệch của chuỗi thời gian tại thời điểm t
- Y_t là giá trị của chuỗi thời gian tại thời điểm t
- B là toán tử lùi thời gian (backshift operator) và được định nghĩa như sau: $B^D * Y_t = Y_t$-d

Để tính I, ta sẽ lấy giá trị của chuỗi thời gian tại thời điểm t trừ đi giá trị của chuỗi thời gian tại thời điểm t - d, sau đó nhân với $(1 - B)^D$ để chuyển đổi về dạng hiện tại.

Trung bình di động (MA)(q): Thành phần này mô hình hóa sai số của dự đoán, đó là sự khác biệt giữa giá trị quan sát và giá trị dự đoán của chuỗi thời gian. Thứ tự của thành phần trung bình di động, được ký hiệu bằng tham số q, xác định số thuật ngữ lỗi trước đó được sử dụng trong mô hình.

→ Công thức tính (MA): là tổng trọng số của q thuật ngữ lỗi trước đó, mỗi thuật ngữ lỗi được nhân với một hệ số trọng số tương ứng:

$$MA_t = c + e_t + \theta_1 * e_{t-1} + \theta_2 * e_{t-2} + \ldots + \theta_q * e_{t-q}$$

Trong đó:

- MA_t là giá trị của thành phần trung bình di động tại thời điểm t
- c là hằng số
- e_t là thuật ngữ lỗi tại thời điểm t
- θ_1, ..., θq là các hệ số trọng số tương ứng với các thuật ngữ lỗi trước đó.

Các hệ số trọng số θ được ước tính từ dữ liệu bằng cách sử dụng phương pháp tối ưu hóa mất mát như phương pháp cực đại hợp lý (maximum likelihood) hoặc phương pháp bình phương

tối thiểu (least squares). Khi giá trị của q tăng, mô hình sẽ có độ phức tạp cao hơn và yêu cầu một lượng lớn dữ liệu để ước tính các hệ số trọng số chính xác.

Kết luận: Phương trình hồi quy ARIMA(p,d,q): $\Delta x_t = \phi_1 \Delta x_{t-1} + \phi_2 \Delta x_{t-2} + \phi_p \Delta x_{t-p} + \cdots + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$

p : đề cập đến độ trễ giữa các quan sát trong mô hình. Độ trễ bậc p chính là giá trị lùi về quá khứ p bước thời gian của chuỗi.

d*:* đề cập đến số lượng các phép biến đổi khác biệt cần thiết cho chuỗi thời gian để đạt được trạng thái dừng.

q: đề cập đến kích thước của cửa sổ trung bình động.

b. Phương pháp tìm tham số mô hình ARIMA (p,d,q)

Phương pháp thủ công:

- Kiểm tra tính dừng của mô hình dựa vào p-value, quyết định tham số d sẽ là 0 hoặc 1.
- Dựa vào **tự tương quan (ACF - AutoCorrelation Function)**: Tự tương quan là một khái niệm quan trọng trong chuỗi thời gian. Hầu hết các chuỗi thời gian sẽ có sự tương quan với giá trị trễ của nó và các giá trị càng gần nhau thì tương quan càng mạnh hoặc các giá trị cùng thuộc 1 chu kì của chuỗi thì sẽ có tương quan cao (chẳng hạn như cùng tháng trong chu kỳ năm hay cùng quý trong chu kỳ năm) để **quyết định tham số q** và tự tương quan riêng phần (PACF - Partial AutoCorrelation Function) để **quyết định tham số p**.
- **Tiêu chí lựa chọn bậc q** không quá lớn, nếu tại một độ trễ nhỏ nhất mà đoạn thẳng (vuông góc với trục hoành) mà độ dài đại diện cho giá trị của hệ số tự tương quan nằm ngoài khoảng tin cậy thì đó chính là độ trễ phù hợp lớn nhất mà ta nên lựa chọn cho quá trình trung bình trượt.

Phương pháp tự động:

- Dựa vào chỉ số AIC, sử dụng thư viện pmdarima để tự động tìm mô hình phù hợp nhất (lựa chọn mô hình có chỉ số AIC càng nhỏ càng tốt).
- Chỉ số AIC: AIC được hình thành dựa trên lý thuyết thông tin (information theory). Khi một mô hình thống kê được sử dụng để dự báo, kết quả sẽ gần như không bao giờ chính xác hoàn toàn. Vì vậy một số thông tin sẽ bị mất do không thể dự báo từ mô hình. AIC ước tính lượng thông tin tương đối bị mất bởi một mô hình nhất định: mô hình mất càng ít thông tin thì chất lượng của mô hình đó càng cao. Giả sử rằng chúng ta có một mô hình thống kê tương ứng với một bộ dữ liệu. Gọi k là số lượng tham số

ước tính trong mô hình. Đặt $\hat{L}$ là giá trị tối đa của hàm hợp lý (maximum likelihood function) của mô hình. Khi đó, giá trị AIC của mô hình được tính như sau:

$$AIC = 2k - 2ln(\hat{L})$$

2.3.2. Mô hình cây hồi quy

a. Giới thiệu về mô hình cây hồi quy (Regression Tree)

Trong học máy, cây quyết định [12] là thuật toán sử dụng một loạt các quyết định nếu-không thì để phân loại dữ liệu đầu vào theo các câu trả lời được tạo ra. Cây hồi quy là một dạng cụ thể của cây quyết định và là một kỹ thuật học máy có giám sát, với đầu ra được dự đoán là một biến liên tục thay vì biến phân loại như cây quyết định. Cũng giống như cây quyết định, cây hồi quy chọn các phép tách để tách các quan sát nhằm làm giảm sự phân tán của các giá trị thuộc tính đích. Do đó, các giá trị thuộc tính đích có thể được dự đoán từ các giá trị trung bình của chúng trong các lá.

Cây quyết định bao gồm các thành phần sau:

- Nút gốc (Root node): đây là nút bắt đầu quá trình phân tách bằng cách tìm biến phân tách tốt nhất cho biến mục tiêu.
- Các nút quyết định (Decision nodes): đây là các nút tiếp theo hoặc trung gian, trong đó biến mục tiêu lại được phân chia thêm bởi các biến khác.
- Độ tinh khiết của nút (Node purity): Các nút quyết định thường không tinh khiết hoặc chứa hỗn hợp các giá trị biến mục tiêu. Một nút thuần túy là nút mà các giá trị biến mục tiêu chỉ thuộc vào một phân lớp.
- Các nút lá hoặc nút cuối (Leaf/Terminal nodes) là các nút thuần túy, do đó được sử dụng để đưa ra dự đoán về số hoặc lớp được tạo.

Một cây quyết định/cây hồi quy được xem là cây nhị phân (binary tree) thì các nút của nó chỉ có thể có 2 nút con (nhánh).

b. Khối xây dựng cây hồi quy

- ***Các phần tử trong cây hồi quy***

Cây hồi quy được vẽ từ trái sang phải, một cây hồi quy chỉ có các nút phân nhánh (đường dẫn phân tách) nhưng không có nút chìm (đường dẫn hội tụ). Vì vậy, khi được xây dựng thủ công, các mô hình cây có thể phát triển rất lớn và càng về sau thì thường khó vẽ hoàn toàn bằng tay. Ngày nay có nhiều phần mềm chuyên dụng được sử dụng thay thế để xây dựng các mô hình ứng dụng Cây quyết định.

- ***Luật/Quy tắc quyết định***

Cây hồi có thể được tuyến tính hóa thành các luật quyết định, trong đó kết quả là nội dung của nút lá và các điều kiện dọc theo đường dẫn tạo thành một liên kết trong mệnh đề nếu. Các luật

quyết định có thể được tạo ra bằng cách xây dựng các luật kết hợp với biến mục tiêu. Chúng cũng có thể biểu thị mối quan hệ về mặt thời gian hoặc nhân quả.

c. Chi tiết về việc xây dựng mô hình Decision Tree

❖ Cách hoạt động, phương pháp phân nhánh của mô hình Decision Tree

Cây quyết định dựa trên lý thuyết chia để trị bằng cách tìm kiếm tham lam để có thể xác định các điểm phân chia tối ưu trên cây. Quá trình phân tách này sau đó được lặp lại theo cách đệ quy từ trên xuống cho đến khi tất cả hoặc phần lớn các bản ghi đã được phân loại theo các nhãn lớp cụ thể. Toàn bộ quá trình xây dựng nên Cây quyết định đều được thực hiện dựa trên phương pháp đệ quy và chia để trị.

Tại mỗi lần phân chia, không gian dự đoán được chia làm 2 và được thể hiện bằng hai nhánh mới hướng xuống dưới (ở ví dụ đầu tiên, cây được đặt nằm ngang cho thuận tiện việc nhìn, theo đúng lý thuyết thì ví dụ sẽ nằm dọc từ trên xuống). Thuật toán này được gọi là tham lam (greedy) vì ở mỗi bước của quy trình phân chia, lựa chọn tốt nhất được sẽ được chọn ngay trong bước đó. Thuật toán không "nhìn" các bước tiếp theo và cố gắng tìm ra nhánh có thể tối ưu hơn cho toàn bộ cây. Do vậy, thuật toán này đánh giá tất cả các biến theo một số tiêu chí thống kê và sau đó chọn biến thể hiện tốt nhất theo các tiêu chí được đề ra.

❖ Tiêu chí lựa chọn biến/nút trên từng bước phân chia của Cây Quyết Định

➢ MSE (Mean Squared Error)

Vì bộ dữ liệu được sử dụng có tính liên tục (giá trị Bitcoin theo thời gian) nên MSE sẽ được sử dụng làm tiêu chí chọn các nút trên từng bước phân chia của Cây Quyết Định.

Bằng việc đo lường chênh lệch bình phương trung bình giữa giá trị dự báo và giá trị thực tế (MSE, sai lệch bình phương trung bình), Cây Hồi Quy (Regression Decision Tree) sẽ dùng giá trị chênh lệch đó để xác định chất lượng của các giá trị dự báo ở từng bước phân nhánh của cây. Thuật toán hoạt động dựa trên "greedy" (thuật toán tham lam) bằng việc tìm ra MSE nhỏ nhất ở từng giai đoạn phân nhánh (tối ưu cục bộ) để giảm thiểu MSE của kết quả dự báo so với kết quả thực tế.

❖ Pruning (Cắt tỉa)

➢ Pre Pruning (Cắt tỉa trước)

Kỹ thuật cắt tỉa trước (pre pruning) đề cập đến việc dừng sớm sự phát triển của cây quyết định. Kỹ thuật pre pruning liên quan đến việc điều chỉnh các siêu tham số (hyperparameter của mô hình Decision tree trước quy trình huấn luyện. Các siêu tham số của DecisionTreeClassifier trong SkLearn bao gồm max_depth, min_samples_leaf, min_samples_split có thể được điều chỉnh để sớm ngăn chặn sự phát triển của cây và ngăn mô hình gặp phải tình trạng overfitting.

Cách tốt nhất là sử dụng kỹ thuật GridSearchCV để tìm bộ siêu tham số tốt nhất cho mô hình Cây quyết định.

Một thách thức với phương pháp cắt tỉa sớm là nó phải đối mặt với vấn đề 'horizon' (đường chân trời)', trong đó việc cắt tỉa sớm có thể ngăn cản một số sự phân chia hiệu quả hơn trong các bước phân tách.

➢ **Post Pruning (Cắt tỉa sau)**

Trong kỹ thuật này, chúng ta cho phép cây phát triển đến độ sâu tối đa của nó. Sau đó, loại bỏ các nhánh của cây để tránh vấn đề về overfitting. Xem xét hiệu quả các cây con của cây đầy đủ được đánh giá theo một tiêu chí và sau đó loại bỏ các cây con không đạt tiêu chí. Đây là phương pháp mà việc chúng ta làm là "xây dựng" cây một cách hiệu quả và chuyển đổi các lá thành các nút và cây con. Các tiêu chí đánh giá các cây con thường là MSE cho cây hồi quy (regression tree) và lỗi phân loại (classification error) cho cây phân loại (classification tree).

Một trong những vấn đề của việc post pruning là một cây quyết định có thể phát triển rất sâu và lớn, do đó việc đánh giá tất cả các nhánh của cây có thể tốn kém về mặt tính toán. Một kỹ thuật pruning quan trọng là Cắt tỉa độ phức tạp chi phí (**Cost complexity pruning**, CCP) là một giải pháp để giải quyết vấn đề này. CCP là một kỹ thuật phức tạp và nâng cao, được tham số hóa bởi tham số α trong module DecisionTreesclassifier của Scikit Learn.

d. Ứng dụng của Decision Tree trong việc dự báo

Về mặt thực tế, Cây quyết định sau khi được huấn luyện và đem đi thực nghiệm, việc dự báo trở nên rất dễ dàng. Về cơ bản, Cây cung cấp một flowchart dựa trên các biến dự đoán khác nhau. Giả sử chúng ta có một phiên bản mới tham gia vào luồng cùng với các giá trị của các biến dự báo khác nhau. Không giống như dữ liệu huấn luyện và kiểm tra, nó sẽ không có lớp cho biến mục tiêu được dự báo. Việc dự đoán lớp này sẽ được thực hiện bằng cách di chuyển xuống các nhánh cây, kiểm tra giá trị của các biến dự báo khác nhau ở các nhánh khác nhau. Cuối cùng, lớp được dự báo đó sẽ di chuyển vào một nút lá và sẽ được phân loại theo lớp chiếm ưu thế trong nút lá.

e. Vấn đề về Overfit đối với mô hình Decision Tree

Overfit là một trong những vấn đề lớn đối với với thuật toán Decision Tree. Điều này dẫn đến một cấu trúc cây cực kỳ phức tạp và không thể xử lý tốt các trường hợp tổng quát mà chỉ có thể áp dụng cho các bộ dữ liệu giống với bộ dữ liệu huấn luyện hoặc các trường hợp đặc thù. Điều này là do mỗi lá sẽ đại diện cho một tập hợp các kết hợp thuộc tính rất cụ thể được nhìn thấy trong dữ liệu huấn luyện và cây sẽ không thể phân loại các kết hợp thuộc tính không được nhìn thấy trong dữ liệu huấn luyện. Có 3 cách tiếp cận chung để tránh vấn đề overfit:

- Pre pruning: Ngăn cây phát triển quá lớn hoặc quá sâu (quá phức tạp, chi tiết)
- Post pruning: Cho phép cây phát triển tối đa rồi sau đó loại bỏ các nhánh dựa trên các tiêu chí khác nhau
- Sử dụng các mô hình tập hợp nhiều cây với nhau chẳng hạn như Random Forest

f. Ưu điểm - khuyết điểm của mô hình Decision Tree

Ưu điểm	**Nhược điểm**
- Xử lý được các biến liên tục và cả biến phân loại - Thể hiện rõ ràng các thuộc tính quan trọng nhất để dự đoán hoặc phân lớp dữ liệu - Dễ sử dụng và có khả năng phát triển - Đối phó hiệu quả với các giá trị bị thiếu - Xử lý các mối quan hệ phi tuyến tính và các dữ liệu mất cân bằng	- Không thích hợp đối với các tác vụ ước tính có bao thuộc tính liên tục - Các mô hình Cây quyết định có thể dễ mắc lỗi về các vấn đề phân loại với nhiều lớp nếu bộ dữ liệu huấn luyện không đủ lớn - Có thể tốn rất nhiều tài nguyên để huấn luyện mô hình - Dễ gặp vấn đề Overfit

g. Một số bản cải thiện của mô hình Decision Tree

Một số thuật toán như Random Forest hoặc Gradient-Boosted Tree kết hợp nhiều cây quyết định để cải thiện độ chính xác và giảm tình trạng overfitting. Các thuật toán này đặc biệt chuyên dụng đối với các bộ dữ liệu phức tạp và trong nhiều trường hợp mang lại kết quả tốt hơn so với Decision Tree.

2.3.3. Long-Short Term Memory (LSTM)

a. Giới thiệu về mô hình LSTM

Mạng bộ nhớ dài-ngắn hạn (Long Short Term Memory), hay còn được viết tắt là LSTM, là một dạng mạng Nơron hồi quy (Recurrent Neural Network - RNN). Mạng LSTM được thiết kế đặc biệt để có thể xử lý dữ liệu tuần tự, chẳng hạn như chuỗi thời gian, văn bản hay giọng nói. Mạng LSTM có khả năng học các phụ thuộc dài hạn trong dữ liệu tuần tự, giúp chúng phù hợp với các tác vụ như phiên dịch ngôn ngữ, nhận dạng giọng nói và dự báo giá trị chuỗi thời gian. Đối với dữ liệu tuần tự, việc huấn luyện các mô hình truyền thống như mạng Nơron sâu (Deep Neural) hoặc mạng Nơron hồi quy để học các phụ thuộc dài hạn (long-term dependencies) bằng phương pháp lan truyền ngược (backpropagation) thường gặp khó khăn và tỏ ra kém hiệu quả.

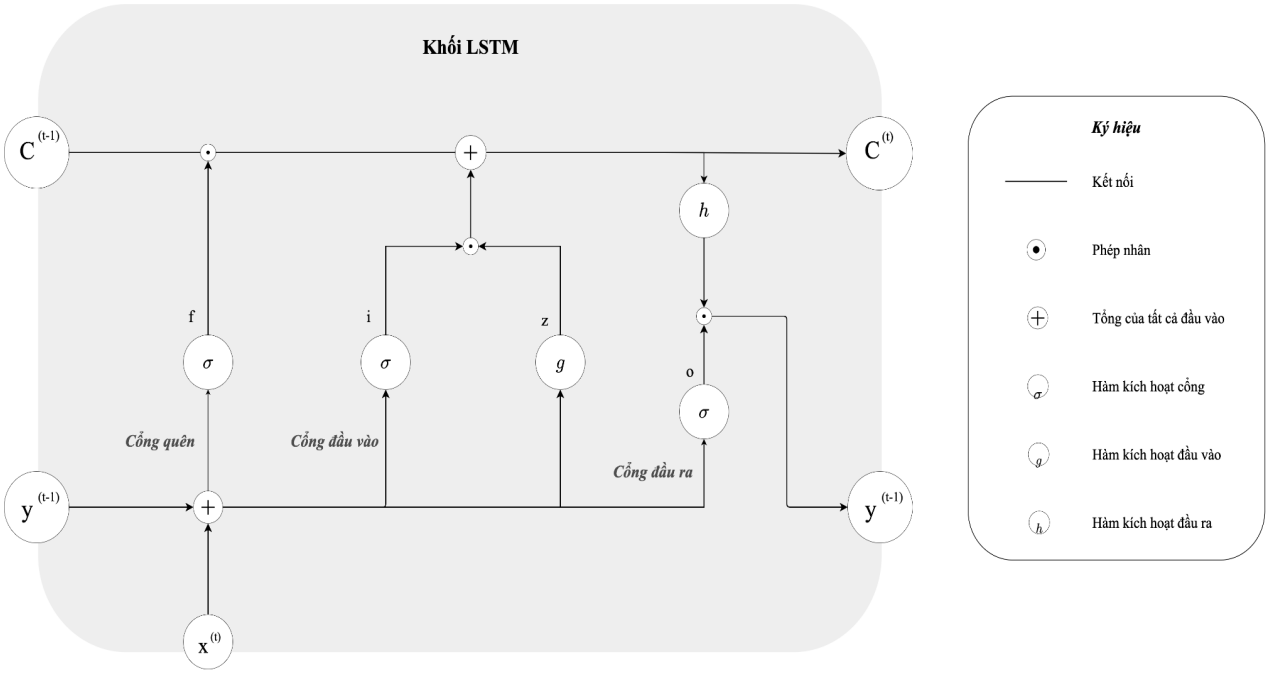
Vấn đề này xuất phát từ hiện tượng triệt tiêu hoặc bùng nổ đạo hàm có hướng (Gradient) (Bengio et al., 1994 [13], Hochreiter et al., 2001 [14]). Nhằm vượt qua hạn chế này khi học các phụ thuộc dài hạn, mô hình LSTM (Hochreiter and Schmidhuber 1997a [15]) đã được giới thiệu. Hiệu năng học của mô hình LSTM đã tạo ra ảnh hưởng lên nhiều lĩnh vực khác nhau, cả về mặt lý thuyết và thực tiễn. Chính vì vậy, có thể nói mô hình LSTM là một mô hình tân tiến và đạt được trạng thái State-of-the-art. Một số ứng dụng thành công của mô hình LSTM có thể kể đến như: hệ thống nhận diện giọng nói của Google, cải thiện hiệu năng dịch máy của Google dịch, Amazon sử dụng LSTM để cải thiện hiệu năng của trợ lý ảo Alexa,...

b. Kiến trúc mô hình LSTM

❖ **Kiến trúc tổng quan**

Nhìn chung, một khối LSTM hay một đơn vị LSTM (unit) sẽ bao gồm các thành phần chính là: một ô (Cell), một cổng đầu vào (Input Gate), một cổng đầu ra (Output Gate) và cổng quên (Forget Gate). Cổng quên này ban đầu không phải là một phần của mạng LSTM, nhưng được đề xuất bởi Gers et al. (2000) để cho phép mạng LSTM tái thiết lập lại trạng thái của nó. Các ô trong mạng LSTM có nhiệm vụ chính là ghi nhớ các giá trị trong khoảng thời gian tùy ý. Những giá trị này sẽ được điều chỉnh bởi các cổng dựa trên luồng thông tin liên quan đến ô.

Một cách ngắn gọn, kiến trúc mạng LSTM bao gồm một tập hợp các mạng hồi quy (Recurrent) con, các mạng con này còn được gọi là khối bộ nhớ. Ý tưởng đằng sau khối bộ nhớ là duy trì trạng thái của khối theo thời gian và điều chỉnh luồng thông tin bằng cách sử dụng các đơn vị cổng phi tuyến. Hình dưới đây minh họa kiến trúc tổng quan của mạng LSTM, bao gồm đầu vào $x^{(t)}$, đầu ra $y^{(t)}$, các hàm kích hoạt. Đầu ra của khối được kết nối hồi quy trở lại với đầu vào của khối và tất cả các cổng.



Hình 4: Kiến trúc mạng LSTM

Giả sử một mạng bao gồm N khối xử lý và M đầu vào. Quá trình chuyển tiếp trong hệ thống Nơron đệ quy này được thực hiện như mô tả dưới đây.

- **Đầu vào khối**

Bước này dành cho việc cập nhật thành phần đầu vào khối bằng cách kết hợp đầu vào hiện tại $x^{(t)}$ và đầu ra $y^{(t-1)}$ của khối LSTM đó trong lần lặp cuối cùng trước đó. Điều này có thể được thực hiện như mô tả dưới đây:

$$\boldsymbol{z^{(t)} = g(W_z x^{(t)} + R_z y^{(t-1)} + b_z)}$$

Trong đó $\boldsymbol{W_z}$ và $\boldsymbol{R_z}$ lần lượt là trọng số gắn liền với $\boldsymbol{x^{(t)}}$ và $y^{(t-1)}$, $\boldsymbol{b_z}$ là vector chệch (bias vector). Giá trị $\boldsymbol{z^{(t)}}$ có thể được hiểu là các giá trị tiềm năng có thể được thêm vào các trạng thái ô $\boldsymbol{c^{(t)}}$ trong dài hạn.

- **Cổng đầu vào**

Tại bước này, cổng đầu vào được cập nhật bằng cách kết hợp giá trị đầu vào hiện tại là $\boldsymbol{x^{(t)}}$, đầu ra $y^{(t-1)}$ và trạng thái ô $c^{(t-1)}$ của khối LSTM đó trong lần lặp cuối cùng trước đó:

$$\boldsymbol{i^{(t)} = \sigma(W_i x^{(t)} + R_i y^{(t-1)} + p_i \odot c^{(t-1)} + b_i)}$$

Với $\odot$ đại diện cho tích Hadamard giữa hai vector, W_i , R_i và p_i lần lượt là trọng số gắn liền với $\boldsymbol{x^{(t)}}$, $y^{(t-1)}$ và $c^{(t-1)}$. $\boldsymbol{b_i}$ là vector chệch (bias vector) gắn liền với thành phần này. Giá trị kích hoạt đầu vào $i^{(t)}$ sẽ quyết định lưu trữ bao nhiêu giá trị tiềm năng $\boldsymbol{z^{(t)}}$ trong dài hạn. Bằng cách sử dụng các giá trị tiềm năng z^t và giá trị kích hoạt đầu vào $i^{(t)}$, lớp LSTM có thể xác định lượng thông tin sẽ được giữ lại trong các trạng thái ô $\boldsymbol{c^{(t)}}$ của mạng.

❖ **Cổng quên**

Tại bước này, khối LSTM xác định thông tin nào sẽ bị xóa khỏi trạng thái ô $c^{(t-1)}$ trước đó của nó. Do đó, các giá trị kích hoạt $f^{(t)}$ của các cổng quên ở bước thời gian t hiện tại sẽ được tính dựa trên giá trị đầu vào hiện tại $\boldsymbol{x^{(t)}}$, đầu ra $y^{(t-1)}$ và giá trị ô $c^{(t-1)}$ của các ô nhớ ở bước thời gian (t - 1) trước đó:

$$f^{(t)} = \sigma(\boldsymbol{W_f x^{(t)}} + \boldsymbol{R_f y^{(t-1)}} + \boldsymbol{p_f} \odot \boldsymbol{c^{(t-1)}} + \boldsymbol{b_f})$$

Với W_f , R_f và p_f lần lượt là trọng số gắn liền với $\boldsymbol{x^{(t)}}$, $y^{(t-1)}$ và $c^{(t-1)}$. $\boldsymbol{b_f}$ là vector chệch (bias vector) gắn liền với thành phần này.

❖ **Giá trị ô**

Sau khi đã xác định được lượng thông tin sẽ bị xóa (hoặc "quên") và lượng thông tin sẽ được giữ lại, khối LSTM sẽ tiến hành tính toán giá trị ô hiện tại dựa trên đầu vào khối $\boldsymbol{z^{(t)}}$, giá trị kích hoạt đầu vào $i^{(t)}$, giá trị kích hoạt cổng quên $f^{(t)}$ và trạng thái ô trước đó $c^{(t-1)}$. Điều này có thể được thực hiện như sau:

$$c^{(t)} = \boldsymbol{z^{(t)}} \odot \boldsymbol{i^{(t)}} + \boldsymbol{c^{(t-1)}} \odot \boldsymbol{f^{(t)}}$$

Kết quả của phép tính $\boldsymbol{c^{(t-1)}} \odot \boldsymbol{f^{(t)}}$ chính là lượng thông tin sẽ bị xóa khỏi trạng thái ô, trong khi kết quả của phép tính $\boldsymbol{z^{(t)}} \odot \boldsymbol{i^{(t)}}$ chính là lượng thông tin sẽ được giữ lại trong dài hạn.

❖ **Cổng đầu ra**

Tại bước này, giá trị cổng đầu ra sẽ được tính dựa trên giá trị đầu vào hiện tại $\boldsymbol{x^{(t)}}$, đầu ra $y^{(t-1)}$ và giá trị ô $c^{(t-1)}$ của khối LSTM đó trong lần lặp cuối cùng trước đó:

$$o^{(t)} = \sigma(\boldsymbol{W_o x^{(t)}} + \boldsymbol{R_o y^{(t-1)}} + \boldsymbol{p_o} \odot \boldsymbol{c^{(t-1)}} + \boldsymbol{b_o})$$

Với W_o , R_o và p_0 lần lượt là trọng số gắn liền với $\boldsymbol{x^{(t)}}$, $y^{(t-1)}$ và $c^{(t-1)}$. $\boldsymbol{b_o}$ là vector chệch.

❖ **Đầu ra khối**

Cuối cùng, khối LSTM sẽ tính giá trị đầu ra khối dựa trên trạng thái ô hiện tại $c^{(t)}$ và giá trị cổng đầu ra $o^{(t)}$:

$$y^{(t)} = h(\boldsymbol{c^{(t)}}) \odot \boldsymbol{o^{(t)}}$$

$y^{(t)}$ chính là giá trị đầu ra cuối cùng của toàn bộ khối LSTM, đồng thời cũng là giá trị ngắn hạn mới sẽ được dùng trong việc tính toán tại các bước thời gian tiếp theo.

c. Huấn luyện mô hình LSTM

Mô hình LSTM sử dụng hình thức huấn luyện đầy đủ dựa trên đạo hàm có hướng/độ dốc (Gradient) để điều chỉnh các tham số có thể học được (trọng số) trong mạng. Cụ thể hơn, phương thức lan truyền ngược theo thời gian (Werbos 1990) được sử dụng để tính toán các trọng số kết nối các thành phần khác nhau trong mạng. Do đó, trong quá trình truyền ngược,

trạng thái ô $\boldsymbol{c^{(t)}}$ nhận độ dốc từ $y^{(t)}$ cũng như trạng thái ô $\boldsymbol{c^{(t+1)}}$ tiếp theo. Những độ dốc đó được tích lũy trước khi được sao lưu vào lớp hiện tại.

Trong lần lặp cuối cùng thứ T, mức độ thay đổi $\delta_y^{(t)}$ tương ứng với sai số của mạng được xác định bởi hàm mất mát. Mặt khác, $\delta_y^{(t)}$ cũng là vector chứa các giá trị delta $\Delta^{(t)}$ được truyền lại từ lớp trước đó bao gồm cả các phụ thuộc đệ quy:

$$\delta_y^{(t)} = \Delta^{(t)} + R_z^T \delta_z^{(t+1)} + R_i^T \delta_i^{(t+1)} + R_f^T \delta_f^{(t+1)} + R_f^T \delta_f^{(t+1)}$$

Tiếp theo, mức độ thay đổi trong các tham số gắn liền với các cổng và bộ nhớ ô được tính như sau:

$$\delta_{o^{(t)}} = \delta_y^{(t)} \odot h(\boldsymbol{c^{(t)}}) \odot \boldsymbol{\sigma'}(\hat{\boldsymbol{o}}^{(t)})$$

$$\delta_c^{(t)} = \delta_y^{(t)} \odot \boldsymbol{o^{(t)}} \odot \boldsymbol{h'}(\boldsymbol{c^{(t)}}) + \boldsymbol{p_0} \odot \boldsymbol{\delta_0^{(t)}} + \boldsymbol{p_i} \odot \boldsymbol{\delta_i^{(t+1)}} + \boldsymbol{p_f} \odot \boldsymbol{\delta_f^{(t+1)}} + \boldsymbol{\delta_c^{(t+1)}} \odot \boldsymbol{f^{(t+1)}}$$

$$\delta_{f^{(t)}} = \delta_c^{(t)} \odot \boldsymbol{c^{(t-1)}} \odot \boldsymbol{\sigma'}(\hat{\boldsymbol{f}}^{(t)})$$

$$\delta_{i^{(t)}} = \delta_c^{(t)} \odot \boldsymbol{z^{(t)}} \odot \boldsymbol{\sigma'}(\hat{\boldsymbol{i}}^{(t)})$$

$$\delta_{z^{(t)}} = \delta_c^{(t)} \odot \boldsymbol{i^{(t)}} \odot \boldsymbol{g'}(\hat{\boldsymbol{z}}^{(t)})$$

Với $\hat{\boldsymbol{o}}^{(t)}$, $\hat{\boldsymbol{i}}^{(t)}$, $\hat{\boldsymbol{z}}^{(t)}$, $\hat{\boldsymbol{f}}^{(t)}$ lần lượt là giá trị thô gắn liền với các cổng đầu ra, cổng đầu vào, đầu vào khối và cổng quên, trước khi được biến đổi bởi các hàm kích hoạt tương ứng.

Cuối cùng, độ dốc cho các trọng số được tính như sau:

$$\delta w_* = \sum_{t=0}^{T} \delta_*^t \otimes x^{(t)} \qquad \delta p_i = \sum_{t=0}^{T-1} c^{(t)} \odot \delta_i^{(t+1)}$$

$$\delta R_* = \sum_{t=0}^{T-1} \delta_*^{(t+1)} \otimes y^{(t)} \qquad \delta p_f = \sum_{t=0}^{T-1} c^{(t)} \odot \delta_f^{(t+1)}$$

$$\delta b_* = \sum_{t=0}^{T} \delta_*^{(t)} \qquad \delta p_o = \sum_{t=0}^{T-1} c^{(t)} \odot \delta_o^{(t+1)}$$

2.4. Đánh giá mô hình

- ***MSE***

Lỗi bình phương trung bình (MSE) *[16]* đo lượng sai số trong các mô hình thống kê. Nó đánh giá sự khác biệt bình phương trung bình giữa các giá trị được quan sát và dự đoán. Khi một mô hình không có lỗi, MSE bằng không. Khi sai số mô hình tăng lên, giá trị của nó tăng lên. Sai số bình phương trung bình còn được gọi là độ lệch bình phương trung bình (MSD).

MSE có thể được tính như sau:

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

Với y_i là giá trị quan sát thứ i, $\hat{y}_i$ là giá trị dự đoán tương ứng, n là số lượng quan sát.

- ***RMSE***

Sai số bình phương trung bình gốc [17] hoặc độ lệch bình phương trung bình gốc là một trong những biện pháp được sử dụng phổ biến nhất để đánh giá chất lượng dự đoán. Nó cho thấy các dự đoán giảm bao xa so với các giá trị thực được đo bằng cách sử dụng khoảng cách Euclide. Để tính RMSE, tính phần dư (chênh lệch giữa dự đoán và giá trị đúng) cho từng điểm dữ liệu, tính định mức của phần dư cho từng điểm dữ liệu, tính giá trị trung bình của phần dư và lấy căn bậc hai của giá trị trung bình đó. RMSE thường được sử dụng trong các ứng dụng học có giám sát, vì RMSE sử dụng và cần các phép đo thực tại mỗi điểm dữ liệu được dự đoán. RMSE có thể được tính như sau:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y_i - \hat{y}_i\|^2}{N}}$$

- ***MAPE***

Sai số phần trăm tuyệt đối trung bình (MAPE) [18] có thể được sử dụng trong học máy để đo lường độ chính xác của một mô hình. Cụ thể hơn thì MAPE là một hàm mất mát giải thích sai số một mô hình nhất định..

MAPE được tính bằng cách tìm chênh lệch tuyệt đối giữa giá trị thực và giá trị dự đoán, chia cho giá trị thực. Các tỷ lệ này được thêm vào cho tất cả các giá trị và giá trị trung bình được lấy. MAPE có thể được tính như sau:

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

Giá trị MAPE càng thấp thì mô hình học máy dự đoán giá trị càng tốt.

- ***R Squared***

R-squared (R2) [19] là thước đo thống kê biểu thị tỷ lệ phương sai của một biến phụ thuộc được giải thích bởi một biến độc lập trong mô hình hồi quy.

Trong khi mối tương quan giải thích độ mạnh của mối quan hệ giữa biến độc lập và biến phụ thuộc, R-squared giải thích mức độ mà phương sai của một biến giải thích cho phương sai của biến thứ hai. Vì vậy, nếu R2 của một mô hình là 0.50, thì khoảng một nửa biến thể quan sát được có thể được giải thích bằng các đầu vào của mô hình. R-Squared được cho như sau:

$$R^2 = 1 - \frac{RSS}{TSS}$$

Trong đó:

- R^2: là hệ số xác định
- RSS (Residual sum of square): Tổng phần dư bình phương của đường hồi quy
- TSS (Total sum of square): Tổng phần dư bình phương của đường trung bình

CHƯƠNG 3. CÀI ĐẶT, THỬ NGHIỆM MÔ HÌNH

3.1. Môi trường và công cụ được sử dụng

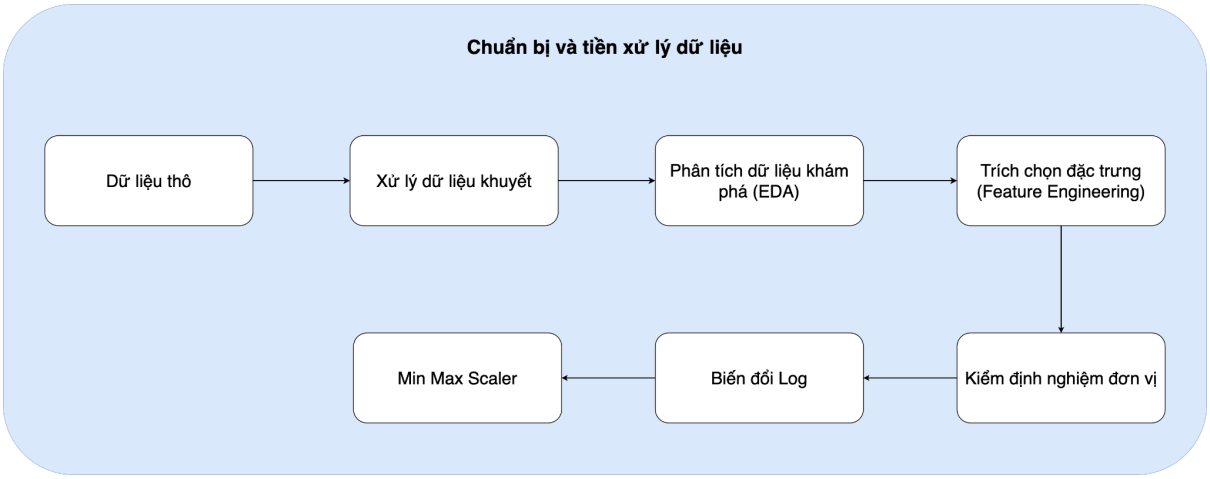
3.2. Mô tả bộ dữ liệu

3.2.1. Tổng quan Bộ dữ liệu gốc

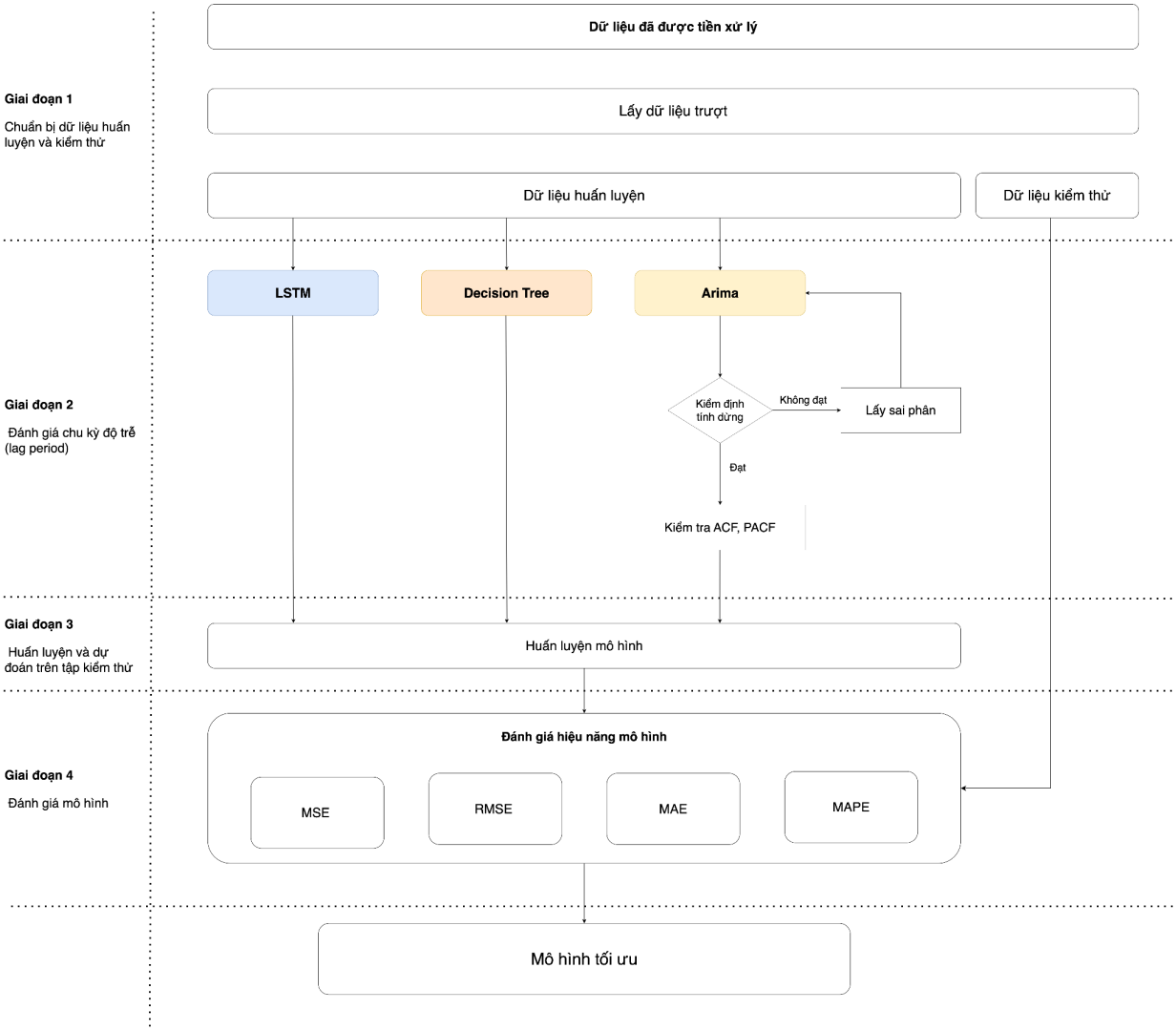
- Bộ dữ liệu bao gồm 12 cột và 3095 dòng, bao gồm:

Thuộc tính	***Giải thích thuật ngữ***
Date	Ngày giao dịch (yyyy–mm-dd)
Open	Giá mở cửa (USD)
High	Giá cao nhất trong 24h (USD)
Low	Giá thấp nhất trong 24h (USD)
Close	Giá đóng cửa (USD)
Volume	Khối lượng giao dịch trong 24h (USD)
RSI	Chỉ số sức mạnh tương đối
Stoch_k	Dao động ngẫu nhiên (giá trị thực tế)
Stoch_d	Dao động ngẫu nhiên (trung bình trượt 3 ngày)
SMA5	Đường trung bình động giản đơn
SP500	Vốn hóa của 500 công ty Mỹ lớn nhất
Gold	Giá vàng

3.3. Quy trình thực nghiệm



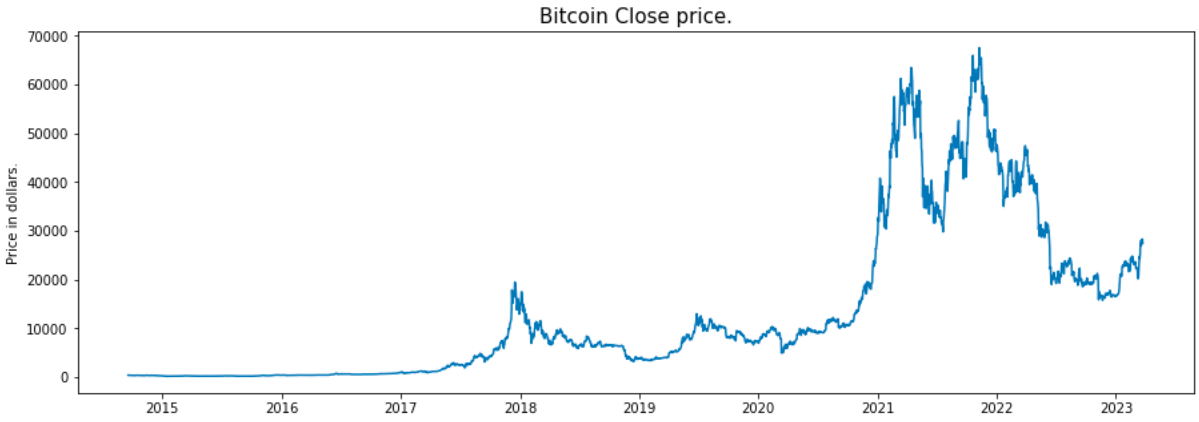
Hình 5: Quy trình chuẩn bị và tiền xử lý dữ liệu



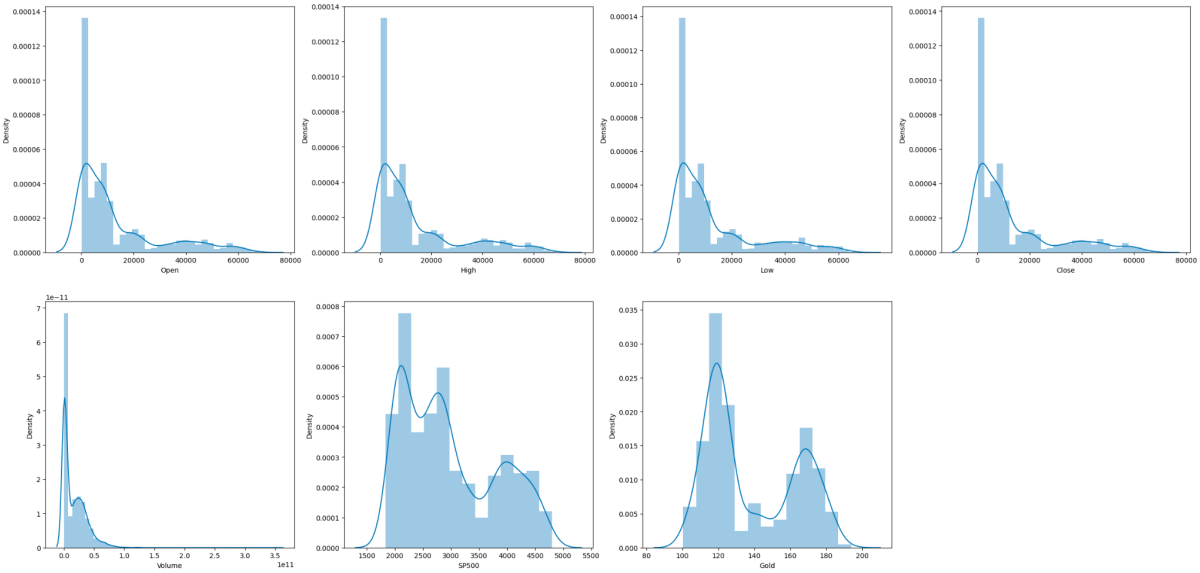
Hình 6: Quy trình huấn luyện mô hình

3.4. Kết quả thực nghiệm

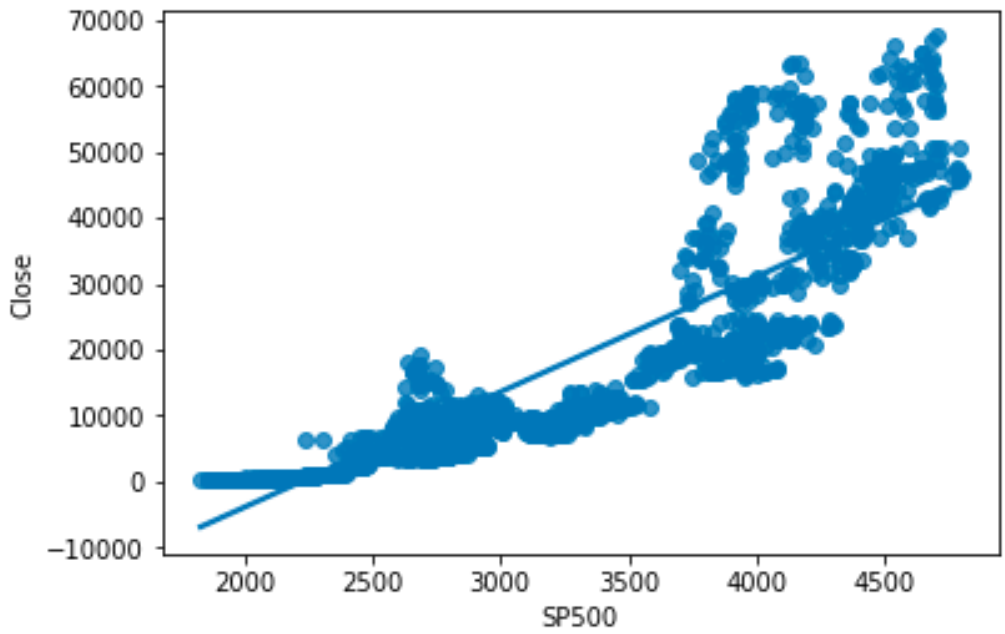
3.4.1. Phân tích dữ liệu khám phá (EDA)



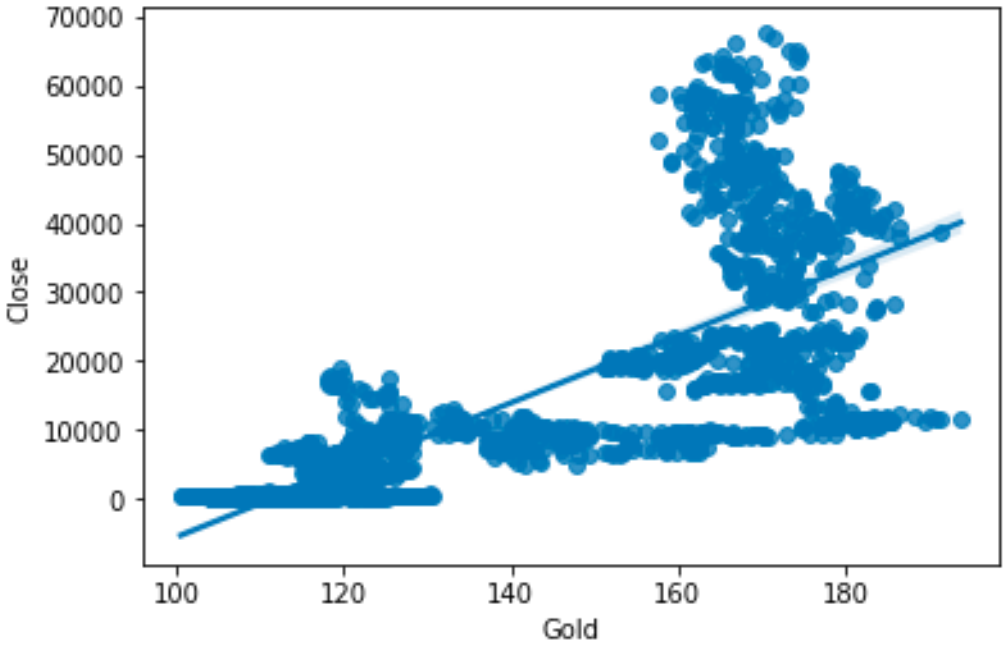
Hình 7: Giá đóng cửa của Bitcoin từ 2014 đến 2023



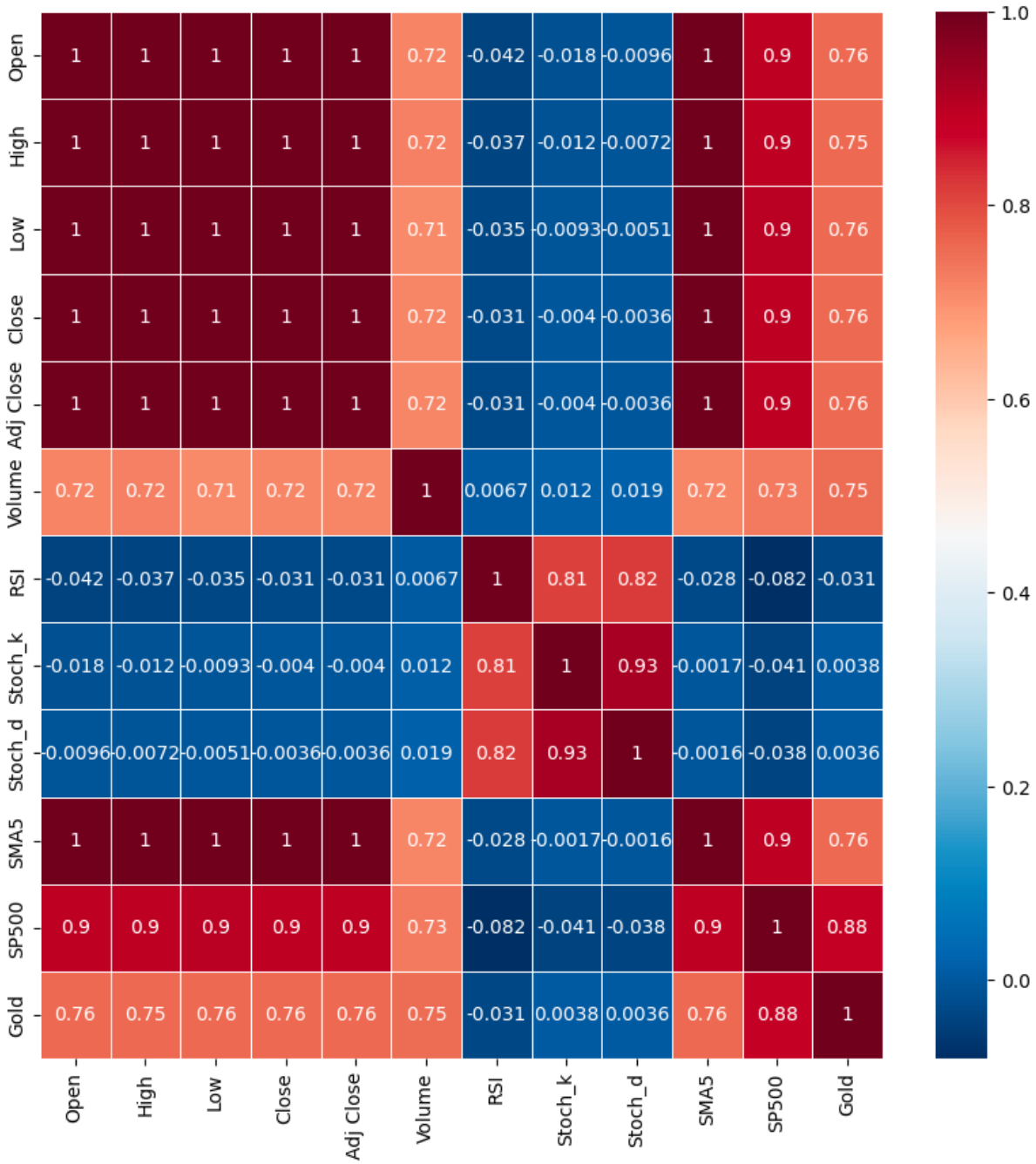
Hình 8: Biểu đồ phân phối của các thuộc tính



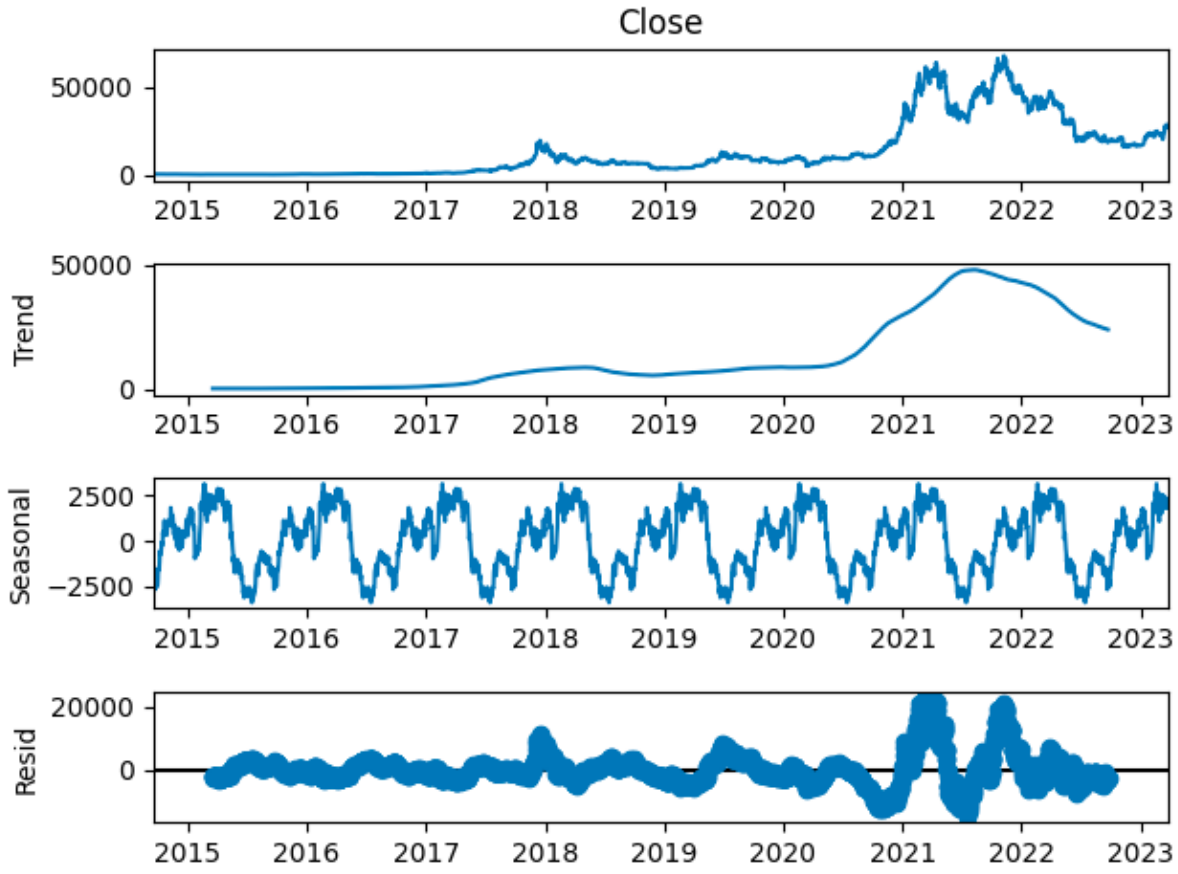
Hình 9: Biểu đồ phân tán giữa giá đóng cửa và chỉ số SP500



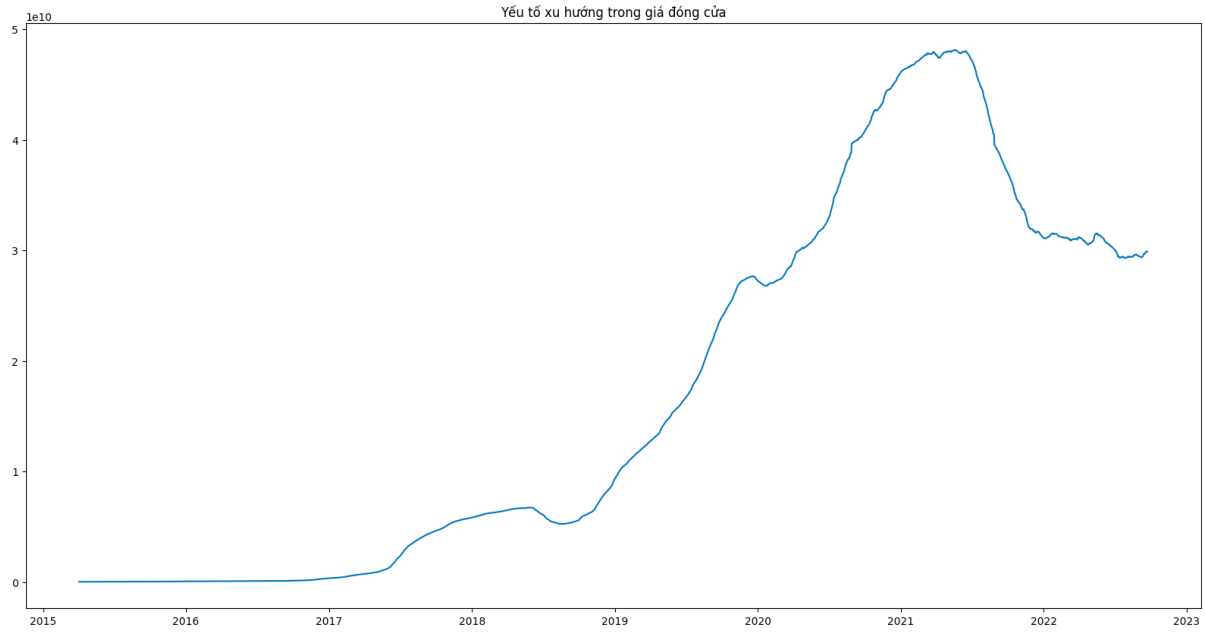
Hình 10: Biểu đồ phân tán giữa giá đóng cửa và giá vàng



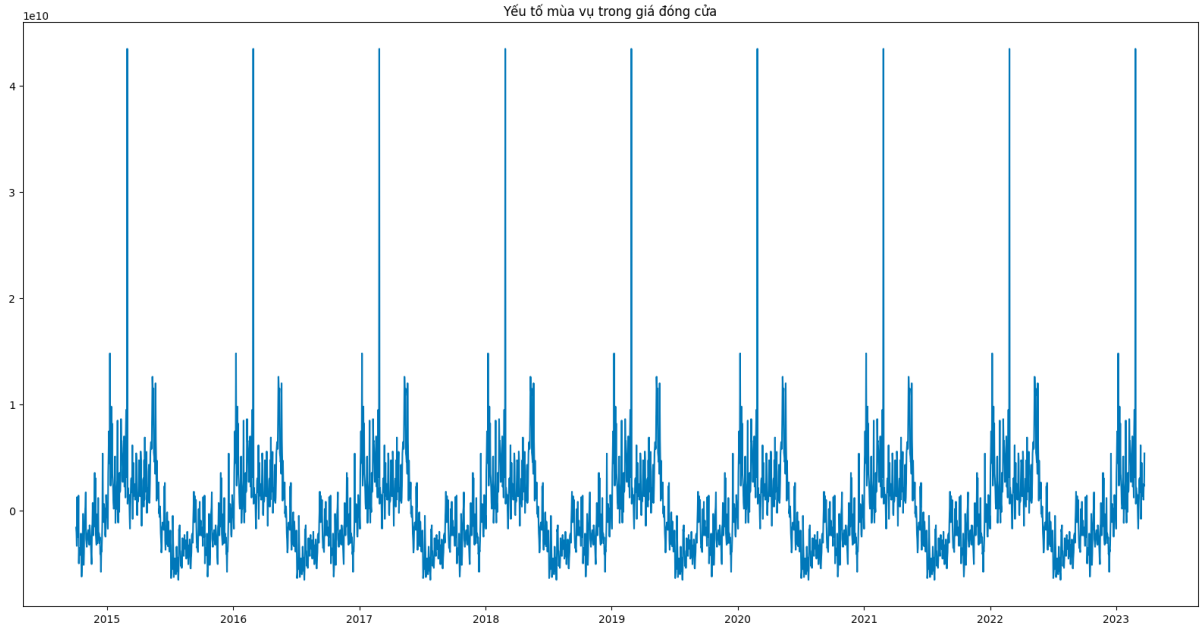
Hình 11: Ma trận tương quan Pearson giữa các biến



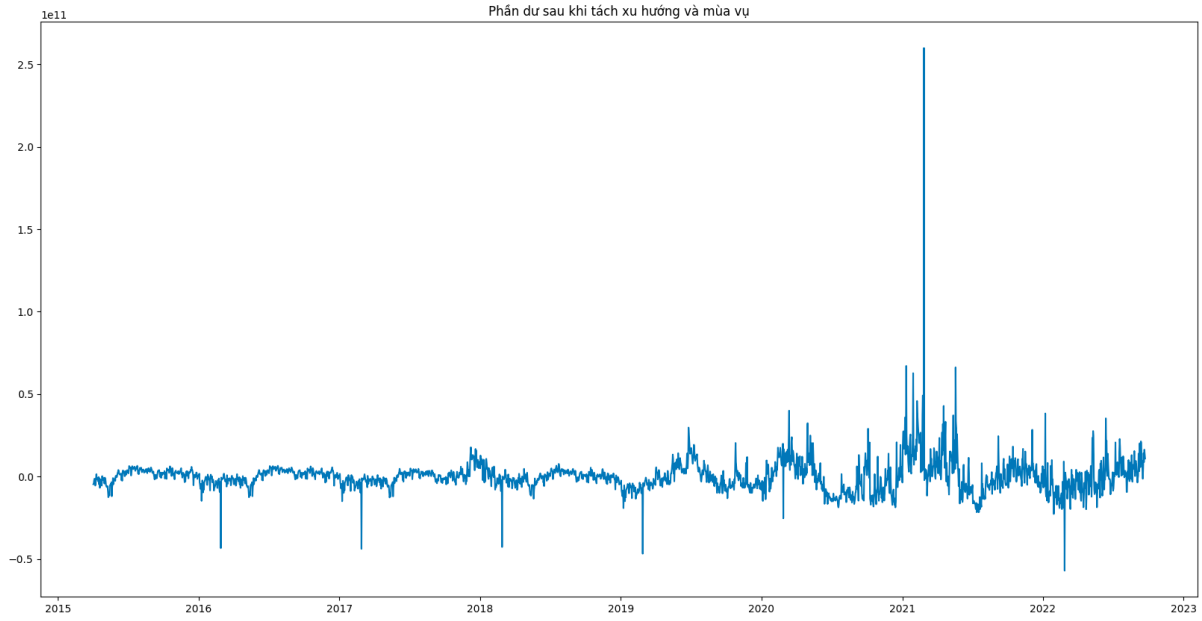
Hình 12: Phân rã mùa vụ của giá đóng cửa



Hình 13: Yếu tố xu hướng trong giá đóng cửa Bitcoin



Hình 14: Yếu tố mùa vụ trong giá đóng cửa Bitcoin



Hình 15: Phần dư sau khi tách yếu tố xu hướng và mùa vụ

3.4.2. Tiền xử lý dữ liệu

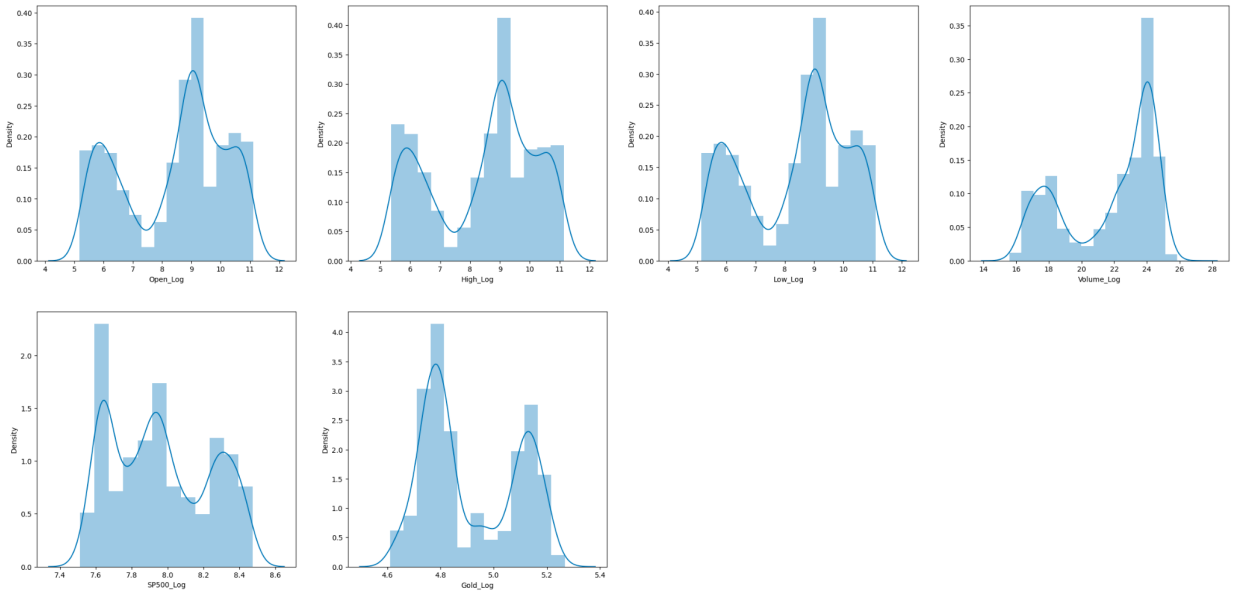
3.4.2.1. Xử lý dữ liệu khuyết

Thuộc tính	Số lượng dữ liệu khuyết
Open	0
High	0

Low	0
Close	0
Volume	0
RSI	13
Stoch_k	13
Stoch_d	15
SMA5	5
SP500	966
Gold	966

Đối với 2 thuộc tính là SP500 và Gold, vì sàn giao dịch chứng khoán và vàng đóng cửa vào 2 ngày cuối tuần là thứ bảy và chủ nhật, nên dữ liệu khuyết ở 2 thuộc tính này sẽ được điền bởi dữ liệu trước ngày đóng cửa, tức là ngày thứ 6.

3.4.2.2. Biến đổi Log



Hình 16: Phân phối của các thuộc tính sau khi thực hiện biến đổi Logarit

3.4.2.3. Xử lý mùa vụ

3.4.3. Kết quả mô hình

3.4.3.1. Mô hình Decision Tree

Cài đặt tìm kiếm siêu tham số tối ưu cho mô hình (hyperparameter tuning):

Chọn tập siêu tham số:

- "max-depth": 2, 4, 6, 8, 10
- "min_samples_split": 2, 4, 6, 8, 10, 20
- "min_samples_leaf": 1, 2, 4, 8, 10
- "max_features": auto, sqrt, log2

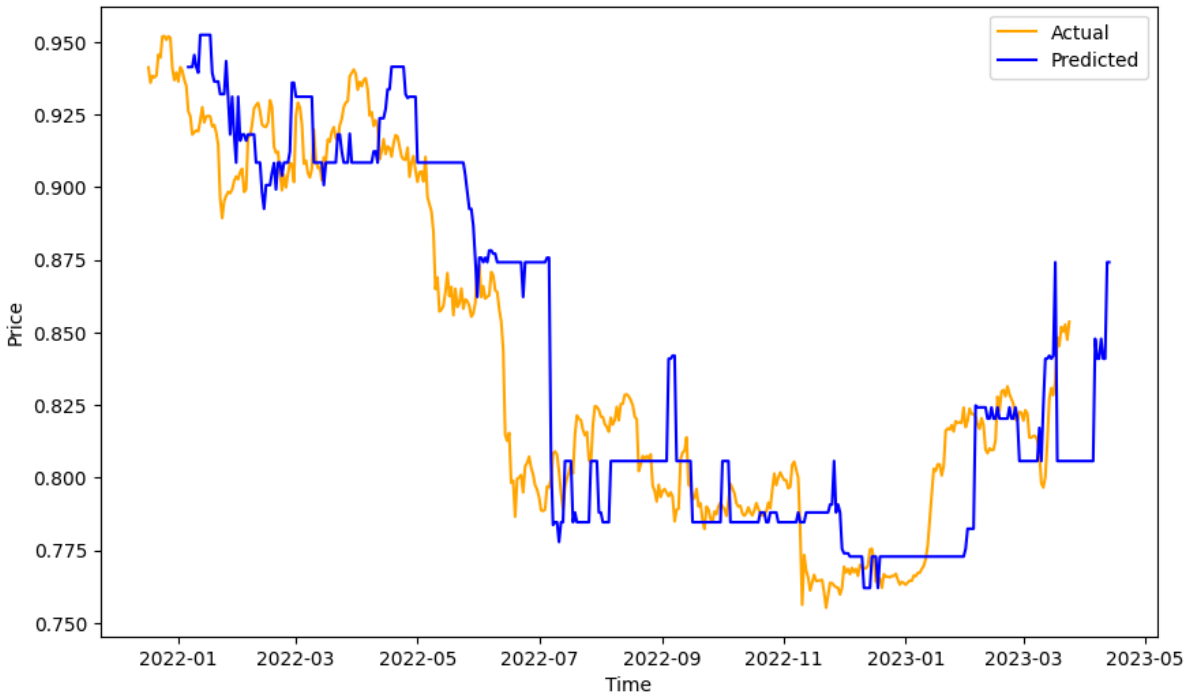
```
# Define regressor with default hyperparameter
dtree = DecisionTreeRegressor()

# Hyperparameter tuning
param_grid = {
    'max_depth': [2, 4, 6, 8, 10],
    'min_samples_split': [2, 4, 6, 8, 10, 20],
    'min_samples_leaf': [1, 2, 4, 8, 10],
    'max_features': ['auto', 'sqrt', 'log2']
}
```

```
# Create a GridSearchCV object to find the best hyperparameters
grid_search = GridSearchCV(dtree, param_grid, scoring='neg_mean_squared_error')
```

Siêu tham số	***Diễn giải***	***Kết quả***
Max_depth	Độ sâu tối đa của cây	10
Max_features	Số biến thuộc tính được sử dụng để đánh giá phân chia tốt nhất	auto
Min_sample_leaf	Số lượng mẫu tối thiểu cần có của một nút lá	2
Min_sample_split	Số lượng mẫu tối thiểu cần có để chia nút quyết định	2
Criterion	Thước đo chất lượng của một lần chia	Lỗi bình phương (Squared Error)

a. Kết quả dự đoán với tập dữ liệu kiểm thử



Hình 17: Kết quả dự đoán trên tập dữ liệu kiểm thử của mô hình Decision Tree

Nhận xét kết quả:

- Ta có thể thấy mô hình đã làm khá tốt việc dự đoán xu hướng giá đóng cửa của Bitcoin. Tuy vậy độ chính xác không thật sự cao, có khá ít điểm dữ liệu mà mô hình đoán đúng.

3.4.3.2. Mô hình LSTM

a. Huấn luyện mô hình

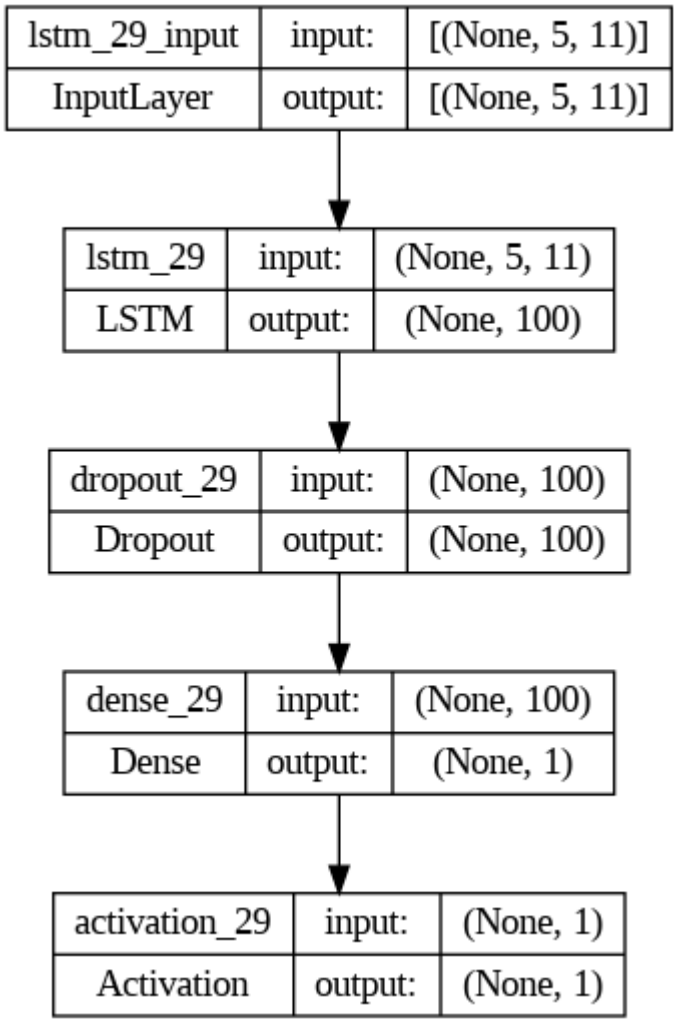
Siêu tham số	***Diễn giải***	***Kết quả***
Batch size	Số lượng mẫu dữ liệu trong một lần huấn luyện	128
Epochs	Số lần duyệt qua hết các dữ liệu trong tập huấn luyện	250
Dropout	Tỷ lệ bỏ qua các đơn vị (1 nút mạng) trong quá trình đào tạo 1 cách ngẫu nhiên	0.2
Learning rate	Tốc độ học	0.01
Neurons	Số lượng Neuron trong mạng	100

Layer	Kích thước đầu ra	Tham số
LSTM	(None, 100)	44800
Dropout	(None, 100)	0
Dense	(None, 1)	101
Activation	(None, 1)	0

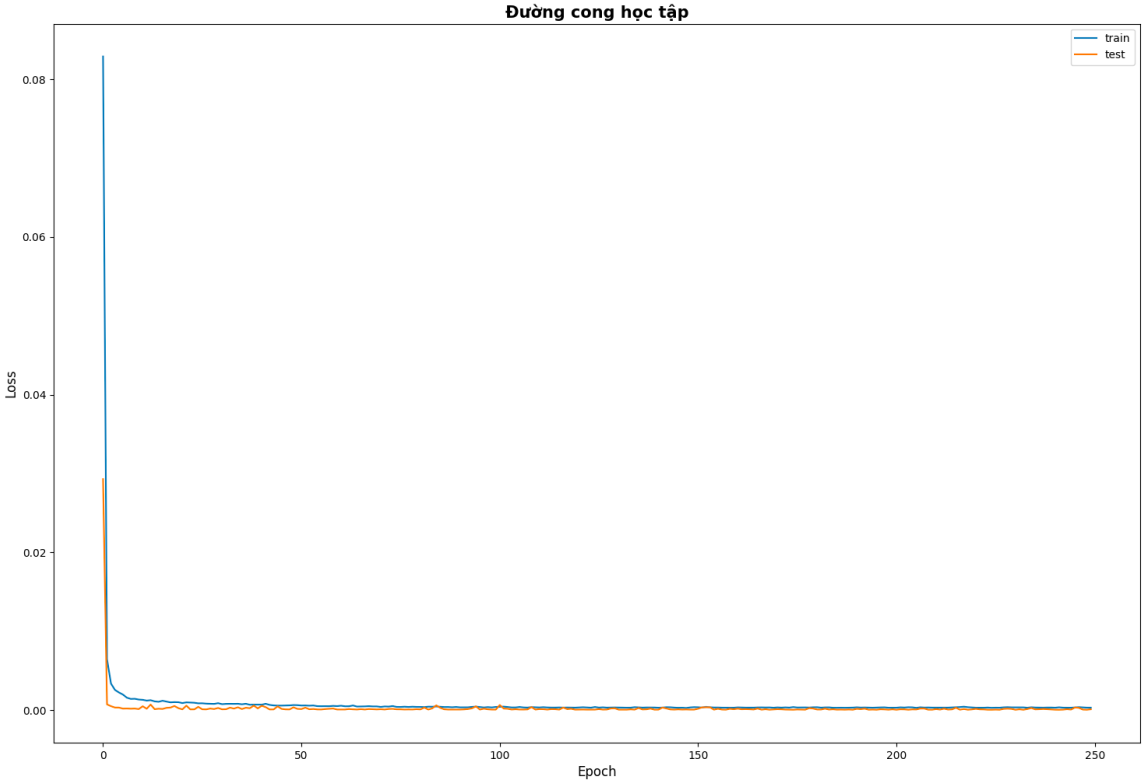
Tổng tham số: 44901

Tham số có thể huấn luyện được: 44901

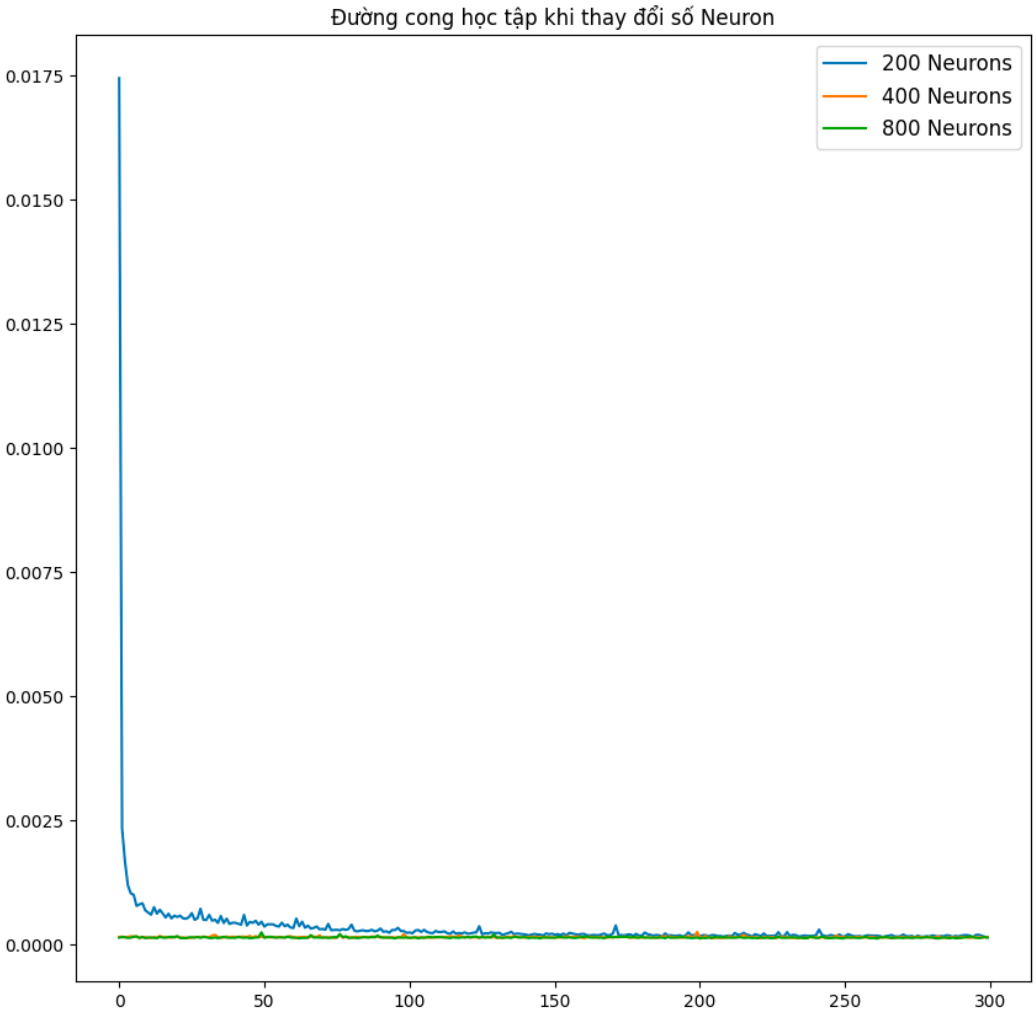
Tham số không thể huấn luyện được: 0



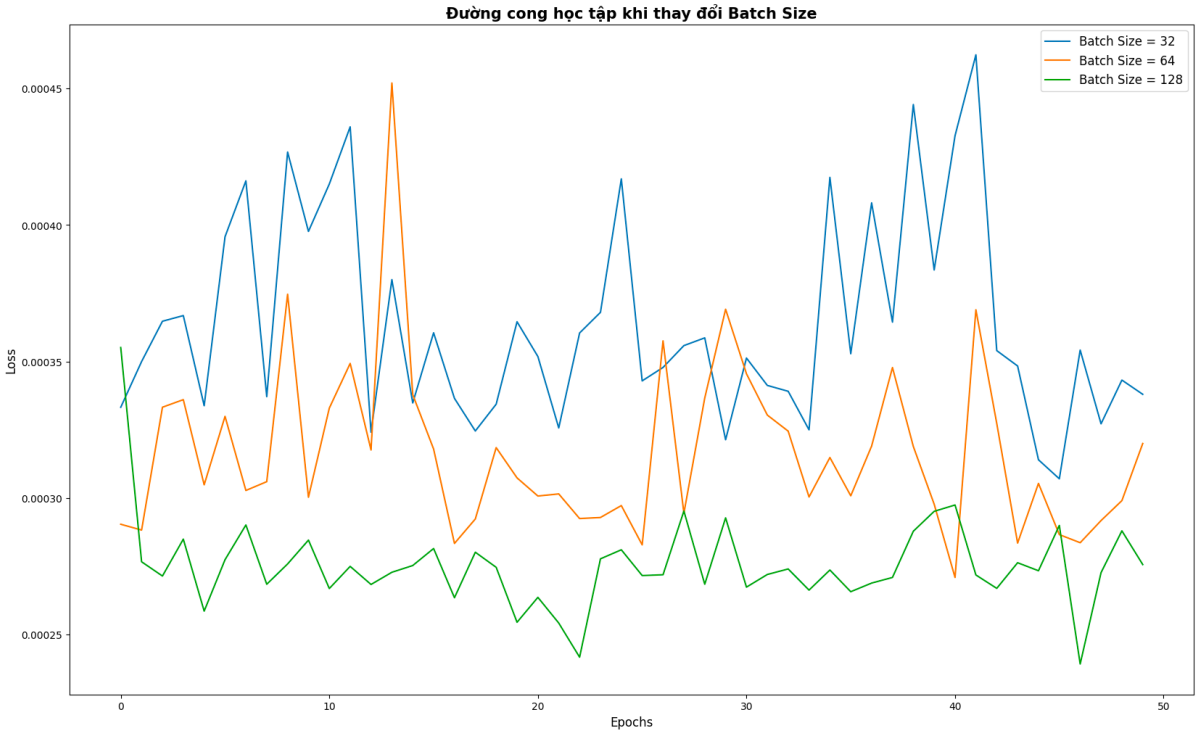
Hình 18: Kiến trúc mạng LSTM



Hình 19: Đường cong học tập của mạng LSTM

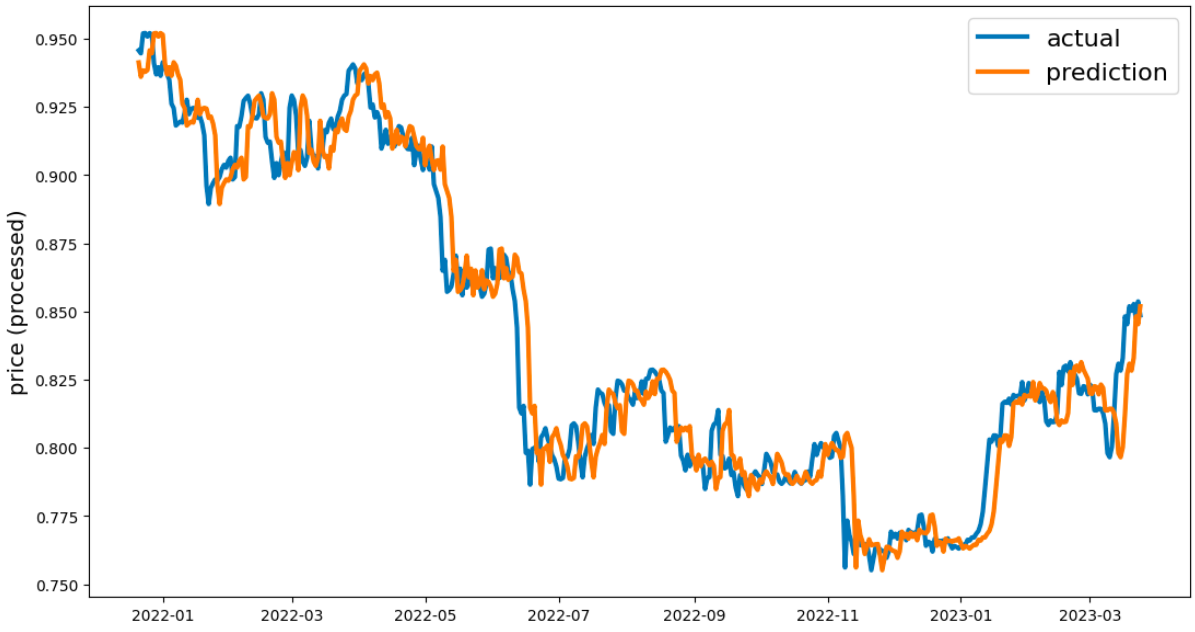


Hình 20: Đường cong học tập của mạng LSTM khi thay đổi số lượng Neuron



Hình 21: Đường cong học tập của mạng LSTM khi thay đổi Batch Size

b. Kết quả dự đoán với tập dữ liệu kiểm thử



Hình 22: Kết quả dự đoán của mô hình LSTM

3.4.3.3. Mô hình ARIMA

a. Xác định tham số (p,d,q)

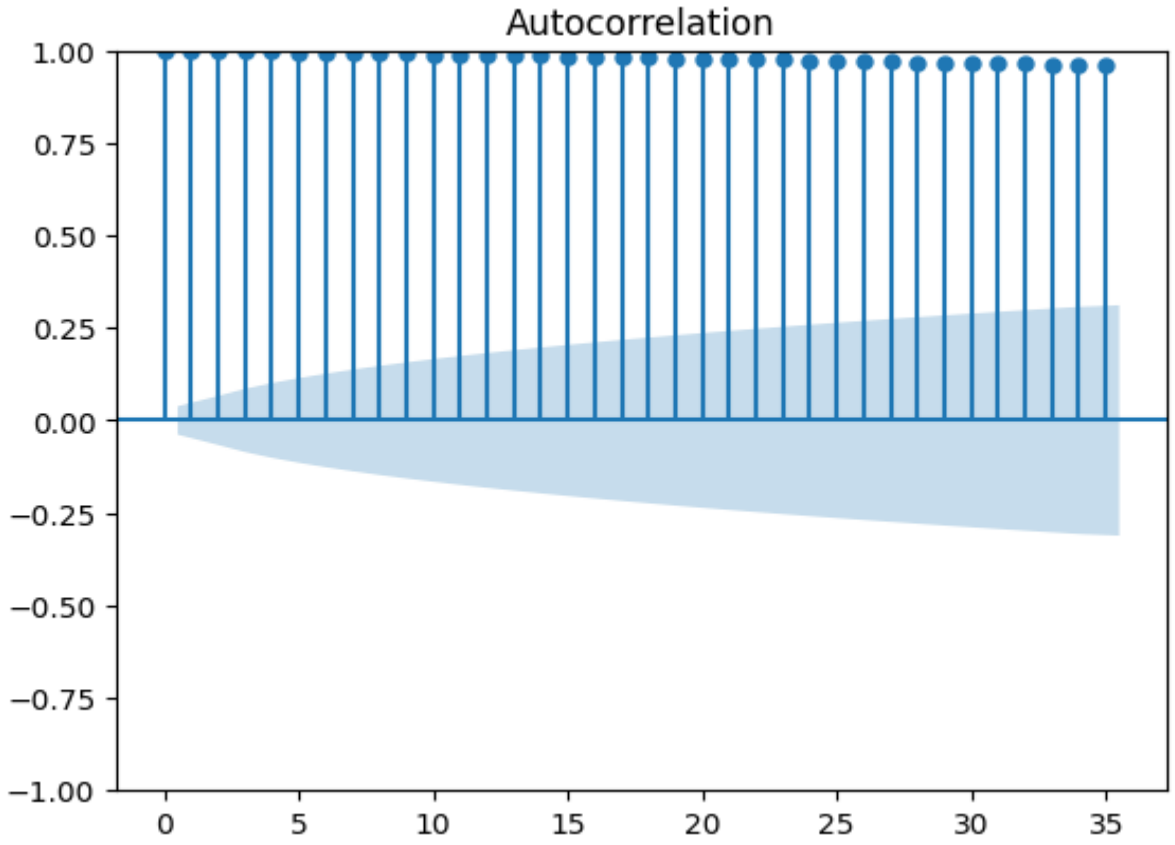
Trước tiên, xét về tính dừng [20] (d) của chuỗi dữ liệu thời gian ở cột phụ thuộc *Close* bằng phương pháp kiểm định nghiệm đơn vị:

```
[59] from statsmodels.tsa.stattools import adfuller
     result = adfuller(train)
     print('p-value: %f' % result[1])
     print('Critical Values:')
     for key, value in result[4].items():
       print('\t%s: %.3f' % (key, value))
```

```
p-value: 0.718313
Critical Values:
        1%: -3.432
        5%: -2.862
        10%: -2.567
```

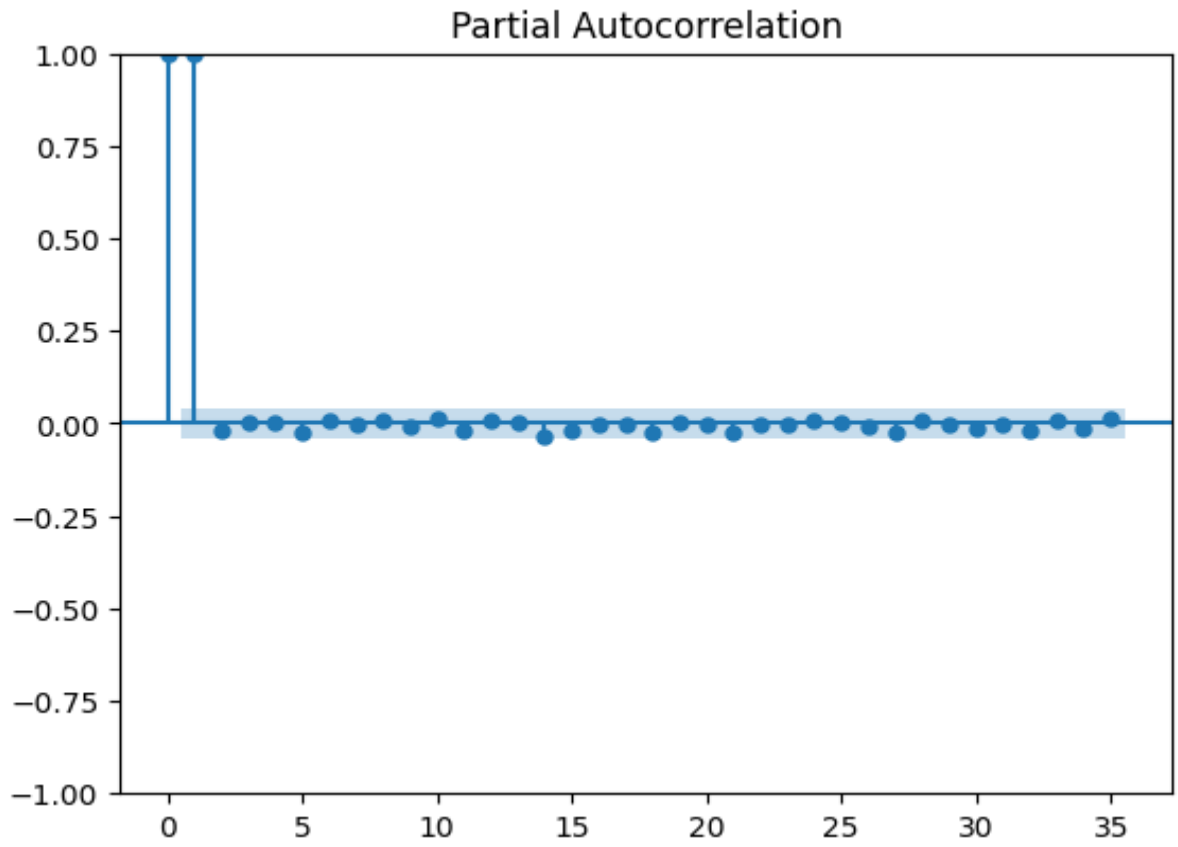
Dựa vào kết quả thu được, ta có p-value > 0.05 chấp nhận H_o , Phương trình đặc trưng có nghiệm đơn vị. Do đó chuỗi lợi suất không có tính chất dừng. → **d = 1.[1]**

Tiếp đến, sử dụng AutoCorrelation Function để tìm tham số (q):



Với trục hoành là độ trễ, trục tung là giá trị của hệ số tự tương quan tương ứng với độ trễ. Dải màu xanh nhạt là khoảng tin cậy 95% để giá trị hệ số tự tương quan bằng 0. Dựa vào cơ sở lý thuyết, lựa chọn **bậc của q tối đa là 2**.**[2]**

Cuối cùng là sử dụng PACF - Partial AutoCorrelation Function để quyết định tham số p. Dựa vào quan sát và cơ sở lý thuyết, **bậc của p tối là 2** tương tự như tham số q. **[3]**



Từ [1,2,3], tham số mô hình ARIMA cần xây dựng có d = 1, p và q thuộc khoảng [0;2]. Để lựa chọn một mô hình có tham số tối ưu nhất, sử dụng đến tiêu chí điểm AIC và tốc độ xử lý của mô hình nào nhỏ nhất sẽ là mô hình tối ưu nhất cho việc sử dụng.

```
[12] from pmdarima import auto_arima
     import warnings
     warnings.filterwarnings('ignore')

     step_fit = auto_arima(train, trace = True,\
                           suppress_warnings = True)
     print(step_fit)

     Performing stepwise search to minimize aic
      ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=-18924.311, Time=2.05 sec
      ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-18932.314, Time=0.35 sec
      ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-18931.868, Time=0.53 sec
      ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-18931.856, Time=0.76 sec
      ARIMA(0,1,0)(0,0,0)[0]             : AIC=-18928.446, Time=0.14 sec
      ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-18929.884, Time=1.32 sec

     Best model:  ARIMA(0,1,0)(0,0,0)[0] intercept
     Total fit time: 5.175 seconds
      ARIMA(0,1,0)(0,0,0)[0] intercept
```

➔ **Dựa vào kết quả lựa chọn được mô hình tối ưu nhất là ARIMA(0,1,0).**

c. Dự đoán chuỗi thời gian, trực quan hóa

Sau khi áp dụng mô hình ARIMA(0,1,0) với bộ dữ liệu train, dự đoán giá trị từ tháng 01/2022 đến tháng 03/2023 cho ra kết quả khả quan và vẫn chưa khớp 100%, tuy vậy vẫn thể hiện được đúng xu hướng giá trong khoảng thời gian thực tế cũng sự giao động giá qua mỗi ngày thể hiện gần như chính xác.

3.5. Phân tích và đánh giá

Tên mô hình	**RMSE**	**MSE**	**MAPE**	**R-Squared**
Decision Tree	0.01334	0.000178	0.01206	0.94795
LSTM	0.013	0.00017	0.0105	0.9508
ARIMA	0.0055	0.000031	0.0044	0.99

CHƯƠNG 4. KẾT LUẬN VÀ ĐỀ XUẤT

Dựa vào điểm số thẩm định các biến liên tục của 3 mô hình trên, nhận thấy ARIMA là mô hình đạt được số khả quan, cộng thêm với trực quan hóa 3 mô hình dự đoán giá trị test cũng chỉ ra rằng ARIMA có tỷ lệ khớp với giá thực tế cao nhất trong 3 mô hình. Song đó, ARIMA là mô hình kinh tế lượng chỉ dựa vào 1 biến để thực thi bài toán và dự đoán chuỗi thời gian - trong khi đó LSTM và Decision Tree dựa vào 1 bộ dữ liệu hoàn chỉnh nhiều biến để dự đoán biến mục tiêu (**close price)** nên sự nhận định ARIMA là mô hình tốt nhất cho bài toán này là chưa phù hợp, tuy nhiên trong khuôn khổ bài đồ án hiện tại ARIMA được chọn là mô hình thích hợp nhất.

Mặt khác, việc 3 mô hình cần được cải thiện về thông số và đầu vào dữ liệu cần được mở rộng hơn chứ **không phải chỉ dựa vào mỗi mức giá trong lịch sử (***những sự kiện hy hữu, tâm lý người đầu tư, các quyết định điều chỉnh giá thị trường luôn có một ảnh hưởng lớn đối với giá BTC nói riêng cũng nhưng các loại hình đầu tư khác nói chung***) là điều chắc chắn.**

TÀI LIỆU THAM KHẢO

[1] Mudassir, M., Bennbaia, S., Unal, D. et al.
Time-series forecasting of Bitcoin prices using high-dimensional features: a machine learning approach. Neural Comput & Applic (2020).
https://doi.org/10.1007/s00521-020-05129-6

[2] Frankenfield, J. (2023, April 5).
What Is Bitcoin? How to Mine, Buy, and Use It. Investopedia.
https://www.investopedia.com/terms/b/bitcoin.asp

[3] Taylor, S. (2023, March 4).
Regression Analysis. Corporate Finance Institute.
https://corporatefinanceinstitute.com/resources/data-science/regression-analysis/

[4] Kenton, W. (2022). What Is Nonlinear Regression? Comparison to Linear Regression. Investopedia.
https://www.investopedia.com/terms/n/nonlinear-regression.asp#:~:text=Nonlinear%20regression%20is%20a%20form,1

[5] Wikipedia contributors. (2023, February 19).
Imputation (statistics). Wikipedia.
https://en.wikipedia.org/wiki/Imputation_(statistics)#:~:text=In%20statistics%2C%20imputation%20is%20the,known%20as%20%22item%20imputation%22.

[6] Wikipedia contributors. (2021, August 11).
Listwise deletion. Wikipedia.
https://en.wikipedia.org/wiki/Listwise_deletion#cite_note-Allison2001-1

[7] Wikipedia contributors. (2023b).
Feature scaling. Wikipedia.
https://en.wikipedia.org/wiki/Feature_scaling

[8] sklearn.preprocessing.MinMaxScaler. (n.d.).
Scikit-learn.
http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[9] Wikipedia contributors. (2023b).
Logarithmic scale. Wikipedia.

https://en.wikipedia.org/wiki/Logarithmic_scale

[10] Những người đóng góp vào các dự án Wikimedia. (2023).
Trung bình trượt. vi.wikipedia.org.
https://vi.wikipedia.org/wiki/Trung_b%C3%ACnh_tr%C6%B0%E1%BB%A3t

[11] Khánh, P. Đ. (2019, December 12).
Khoa học dữ liệu. Khanh's Blog.
https://phamdinhkhanh.github.io/2019/12/12/ARIMAmodel.html

[12] Almog, U. (2022, June 2).
Decision Trees, Explained - Towards Data Science. Medium.
https://towardsdatascience.com/decision-trees-explained-d7678c43a59e

[13] Bengio et al., 1994
Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2), 157–166.

[14] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., & others (2001). *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*.

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

[16] Frost, J. (2021).
Mean Squared Error (MSE). Statistics by Jim.
https://statisticsbyjim.com/regression/mean-squared-error-mse/

[17] C3.ai. (2021, September 28).
Root Mean Square Error (RMSE). C3 AI.
https://c3.ai/glossary/data-science/root-mean-square-error-rmse/

[18] Nik. (2022).
How to Calculate MAPE in Python. Datagy.
https://datagy.io/mape-python/

[19] Fernando, J. (2023).

R-Squared: Definition, Calculation Formula, Uses, and Limitations. Investopedia. https://www.investopedia.com/terms/r/r-squared.asp

[20] Tính dừng của dữ liệu chuỗi thời gian và các kiểm định tính dừng. (n.d.).

Trung Tâm Nghiên Cứu Định Lượng | Tư Vấn SPSS, STATA, R, EViews. https://nghiencuudinhluong.com/tinh-dung-cua-du-lieu-chuoi-thoi-gian-va-cac-kiem-dinh-tinh-dung/

[21] Dash, S. (2022, November 3).

Decision Trees Explained — Entropy, Information Gain, Gini Index, CCP Pruning. Medium.

https://towardsdatascience.com/decision-trees-explained-entropy-information-gain-gini-index-ccp-pruning-4d78070db36c

[22] Wikiwand - Mean squared error. (n.d.). Wikiwand.

https://www.wikiwand.com/en/Mean_squared_error

[23] What is a Decision Tree | IBM. (n.d.).

https://www.ibm.com/topics/decision-trees

[24] Arghyadeep. (n.d.).

GitHub - Arghyadeep/Gold-and-Silver-Price-Prediction-with-Rolling-Regression-and-LSTM. GitHub.

https://github.com/Arghyadeep/Gold-and-Silver-Price-Prediction-with-Rolling-Regression-and-LSTM

[25] GeeksforGeeks. (2023).

Decision Tree. GeeksforGeeks.

https://www.geeksforgeeks.org/decision-tree/

PHỤ LỤC

Mã nguồn báo cáo:
https://github.com/minhnhat2001vt/Bitcoin-close-price-regression