



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
KHOA HỆ THỐNG THÔNG TIN

# BÁO CÁO ĐỒ ÁN

## IS403 - PHÂN TÍCH DỮ LIỆU KINH DOANH

Giảng viên hướng dẫn: ThS Dương Phi Long

# NHÓM 17



**22520124**

Trần Vũ Bảo



**22520170**

Phan Thành Công



**22520423**

Phan Thị Thuỷ Hiền



**22520872**

Nguyễn Đỗ Đức Minh

TÊN ĐỀ TÀI

.....

# A Comparative Analysis of Machine Learning, Recurrent Neural Networks and Deep Learning Models for Bitcoin Price Forecasting

.....

x x x x



# NỘI DUNG

01

**GIỚI THIỆU  
ĐỀ TÀI**

02

**BỘ DỮ LIỆU**

03

**CÁC MÔ HÌNH &  
THỰC NGHIỆM**

04

**KẾT LUẬN &  
HƯỚNG PHÁT  
TRIỂN**

# GIỚI THIỆU ĐỀ TÀI

- **Bối cảnh:** Bitcoin là tài sản tài chính toàn cầu nhưng thị trường biến động mạnh, chịu ảnh hưởng từ tâm lý, sự kiện kinh tế và công nghệ, khiến việc dự báo trở thành một thách thức lớn.
- **Vấn đề:** Các mô hình truyền thống (ARIMA) dự báo kém hiệu quả với dữ liệu phi tuyến tính của thị trường tiền số.
- **Bài toán máy học:**
  - Input: Dữ liệu giá Bitcoin.
  - Output: Dự đoán giá đóng cửa (Close) trong 30 ngày tiếp theo.
- **Mục tiêu:** Phân tích và so sánh sự hiệu quả giữa các mô hình học máy, học sâu, giúp đánh giá khả năng dự báo và hỗ trợ ra quyết định đầu tư.

# BỘ DỮ LIỆU

- Nguồn dữ liệu:

- Giá Bitcoin từ nguồn Yahoo Finance (1/1/2018 – 29/4/2025)
- 2676 records, gồm 7 cột: Date, Adjusted Close, Close, High, Low, Open, Volume.

- Tiền xử lý:

- Dữ liệu đầy đủ, không có giá trị thiếu.
- Chia dữ liệu: 80% train, 20% test.
- Mục tiêu dự báo: Giá đóng cửa (Close)

```
print(df.isnull().sum())
Adj Close      0
Close          0
High           0
Low            0
Open           0
Volume         0
dtype: int64
```

	Adj Close	Close	High	Low	Open	Volume
count	2676.000000	2676.000000	2676.000000	2676.000000	2676.000000	2.676000e+03
mean	30911.611075	30911.611075	31555.799526	30174.365559	30883.254133	2.763285e+10
std	25349.645976	25349.645976	25855.937021	24768.714835	25323.647394	2.004455e+10
min	3236.761719	3236.761719	3275.377930	3191.303467	3236.274658	2.923670e+09
25%	9192.341309	9192.341309	9339.087646	9013.514404	9186.810791	1.436387e+10
50%	23651.378906	23651.378906	24124.528320	23162.128906	23634.151367	2.445092e+10
75%	46404.744141	46404.744141	47402.925781	45128.060547	46365.260742	3.615642e+10
max	106146.265625	106146.265625	109114.882812	105291.734375	106147.296875	3.509679e+11

# CÁC MÔ HÌNH & THỰC NGHIỆM

01

**ARIMA, ARIMAX**

02

**XGBOOST**

03

**LSTM**

04

**GRU**

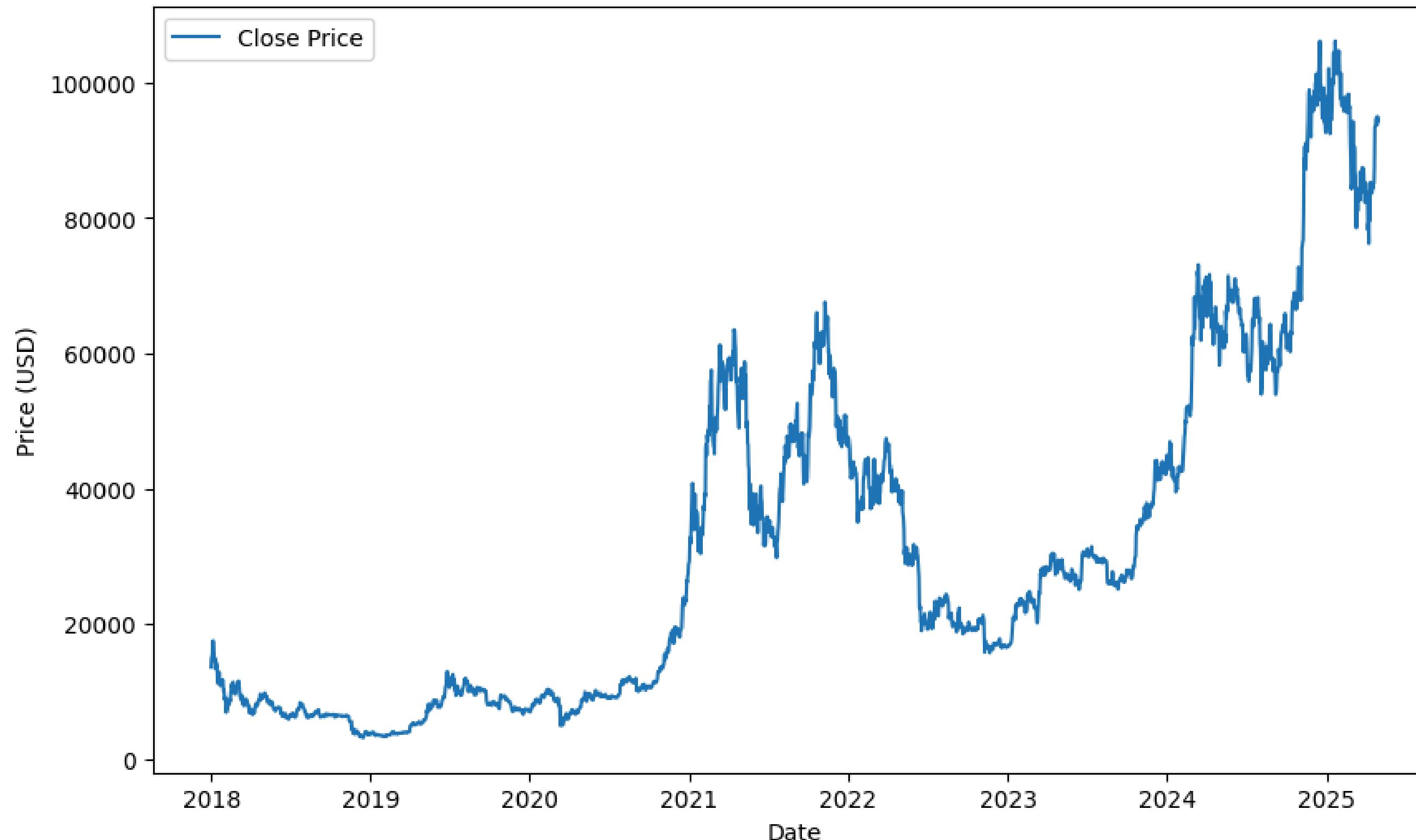
05

**TRANSFORMER**

# ARIMA

# ARIMA

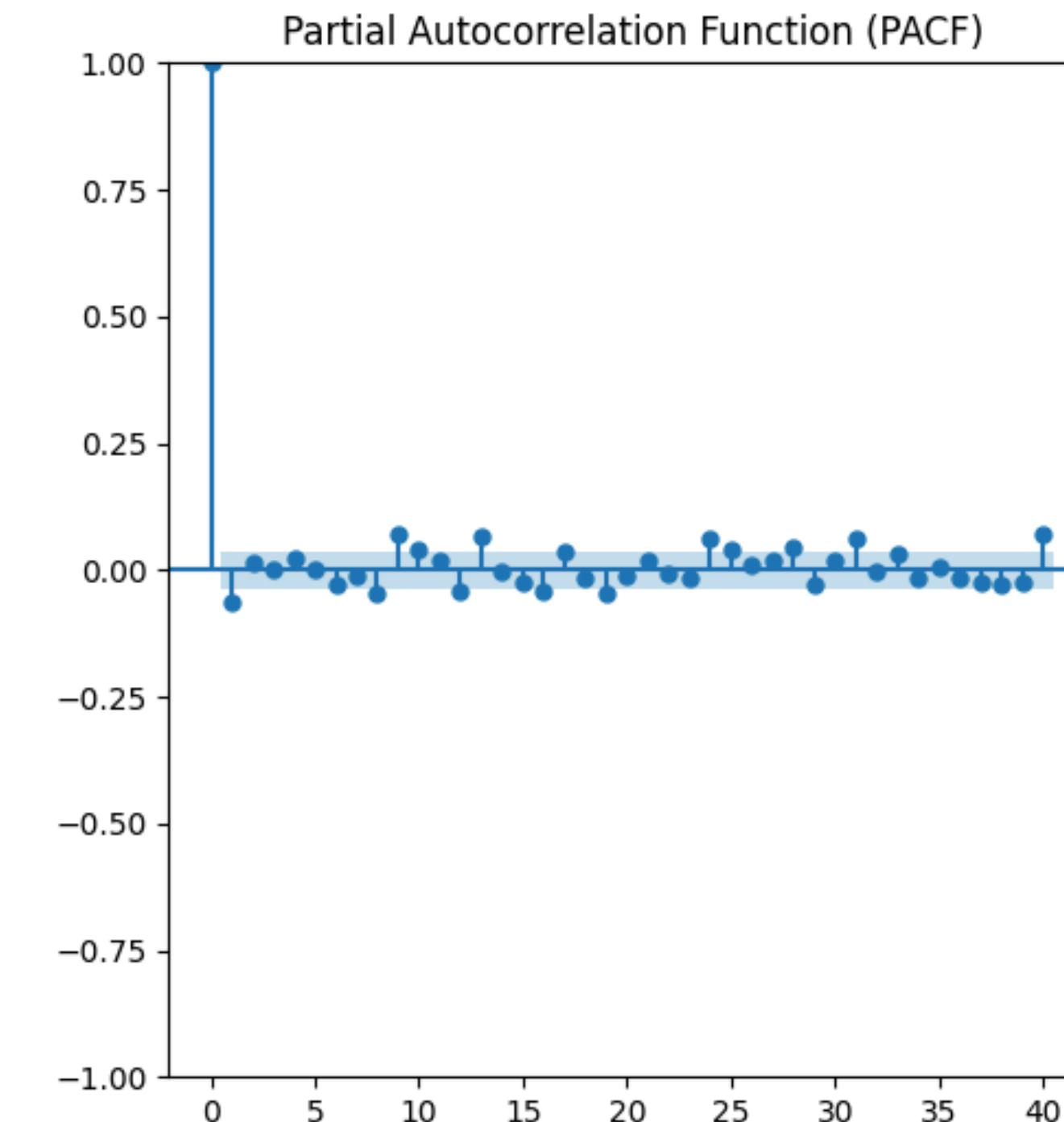
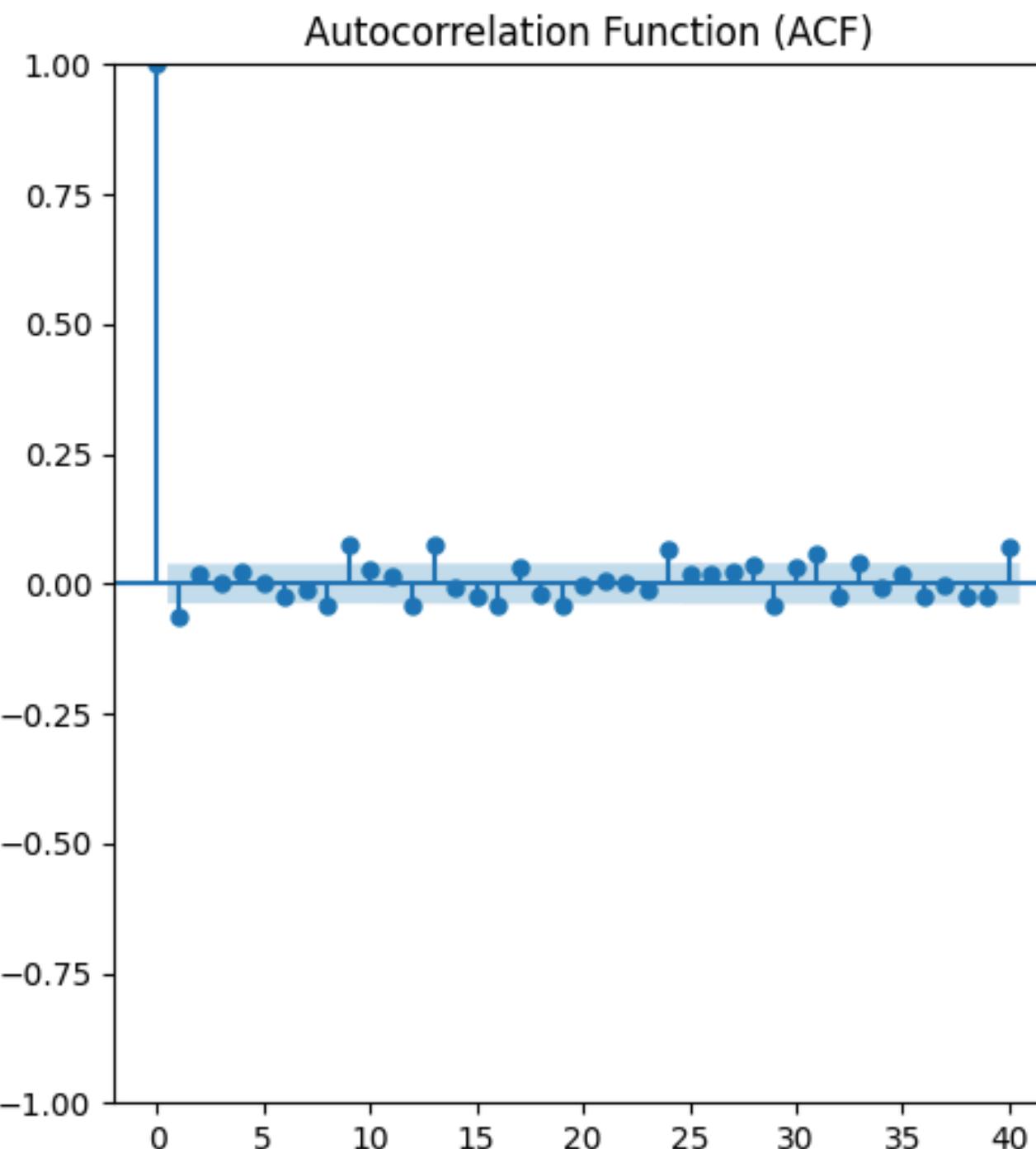
Bitcoin Price Over Time



Biểu đồ giá “Close” theo thời gian

# ARIMA

- Kiểm định ADF để kiểm tra chuỗi dừng
- Thực hiện sai phân bậc 1



# ARIMA

- Từ biểu đồ ACF và PACF, đề xuất 3 mô hình:
  - Mô hình 1: ARIMA(1, 1, 0)
  - Mô hình 2: ARIMA(0, 1, 1)
  - Mô hình 3: ARIMA(1, 1, 1)
- Chia tập train và tập test theo tỷ lệ 8:2

```
train_size = int(len(df) * 0.8)
train, test = df['Close'][:train_size], df['Close'][train_size:]
```

# ARIMA

## Kết quả dự báo của 3 mô hình trên tập test



# ARIMA

## Đánh giá hiệu suất của 3 mô hình bằng các chỉ số AIC, BIC, RMSE

Model	AIC	BIC	RMSE
ARIMA(1,1,0)	35442.096336	35453.432523	37119.364383
ARIMA(0,1,1)	35442.123915	35453.460102	37119.723344
ARIMA(1,1,1)	35443.963081	35460.967362	37113.260644

- ARIMA(1,1,0): Ưu thế về AIC và BIC, nghĩa là mô hình đơn giản và tổng thể tốt hơn theo tiêu chí thống kê
- **ARIMA(1,1,1): Ưu thế RMSE thấp nhất, tức là dự báo chính xác nhất trên tập test**

# ARIMA

## Mô hình ARIMA (1,1,1) dự báo trên tập test

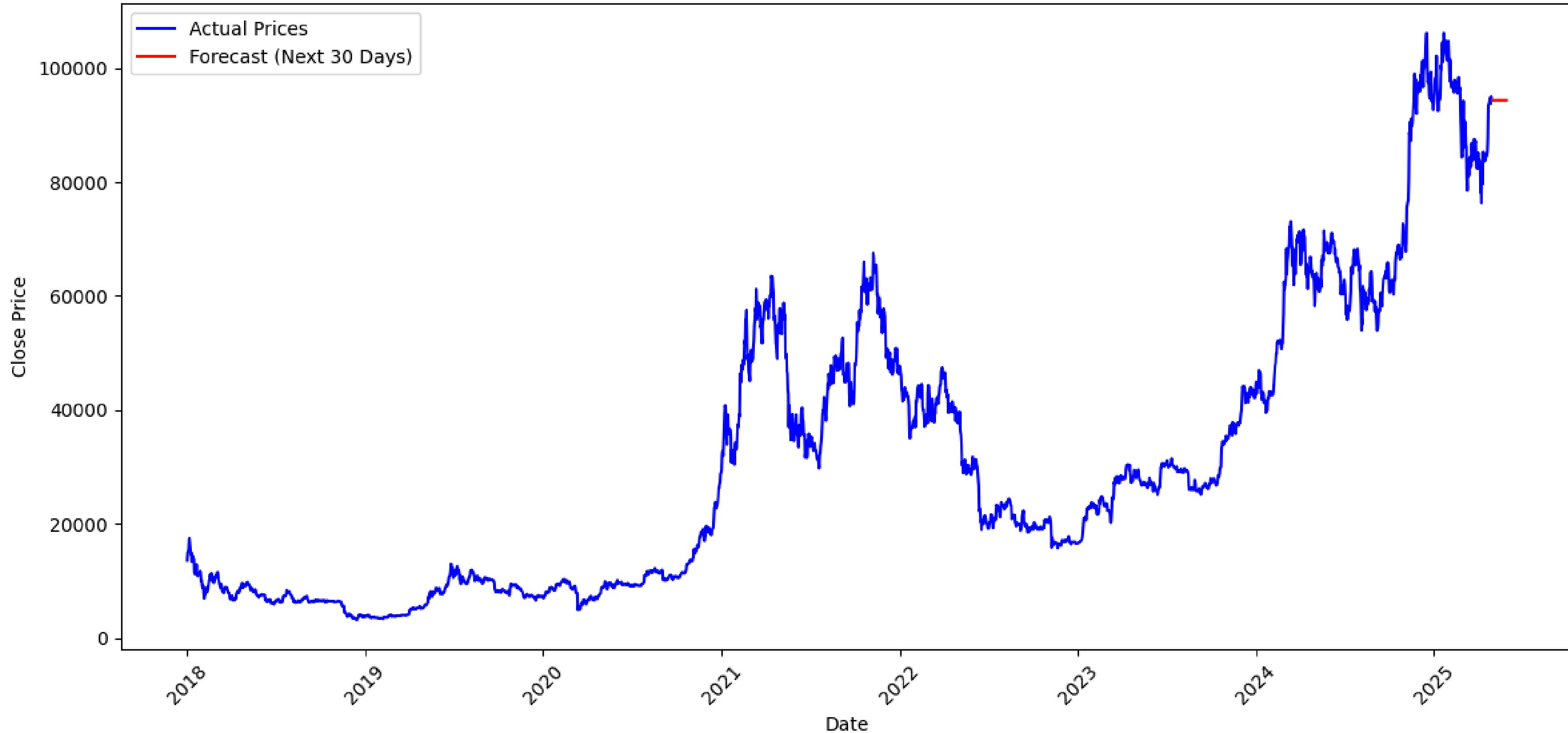
Bitcoin forecasting on test set with ARIMA (1,1,1) model



# ARIMA

## Dự báo 30 ngày tiếp theo trong tương lai

Bitcoin Price Forecast Using ARIMA(1,1,1)





# ARIMAX



# ARIMAX

**Biến ngoại sinh (Exogenous Variable) là Khối lượng giao dịch (Volume)**

```
df[['Close', 'Volume']].corr()
```

	Close	Volume
Close	1.000000	0.494287
Volume	0.494287	1.000000

- Hệ số tương quan giữa Close (giá đóng cửa) và Volume (khối lượng giao dịch) là **0.4943**. Đây là mức **tương quan trung bình**, không quá yếu, nhưng cũng không quá mạnh.
- Volume có ảnh hưởng nhất định đến giá → có thể cân nhắc sử dụng mô hình ARIMAX để đưa “Volume” làm biến ngoại sinh (exogenous variable).

# ARIMAX

- Cũng giống ARIMA, dựa vào biểu đồ ACF và PACF, đề xuất được 3 mô hình để thử nghiệm trên mô hình ARIMAX
  - ARIMAX(1,1,0)
  - ARIMAX(0,1,1)
  - ARIMAX(1,1,1)
- Chia tập train và tập test

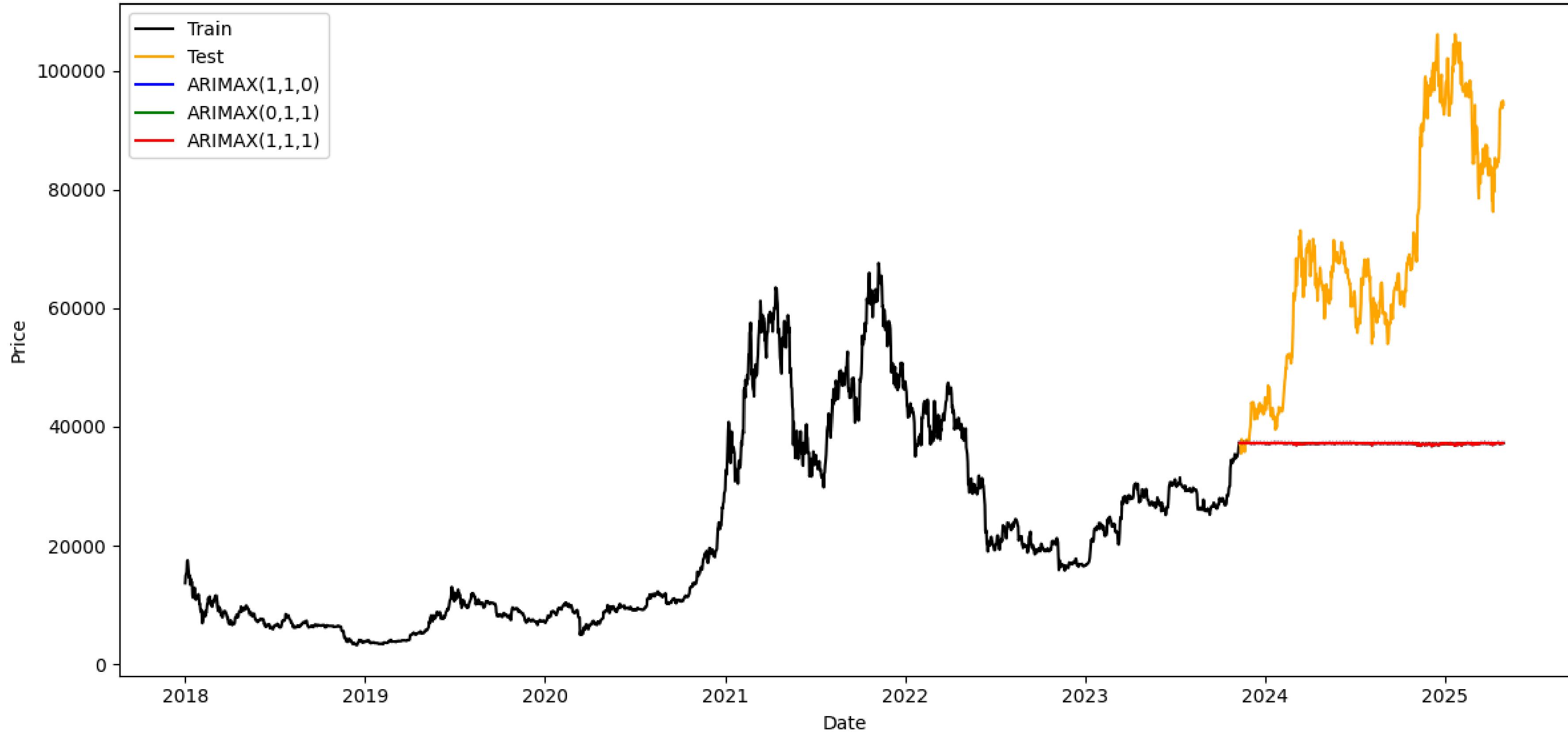
```
train_size = int(len(df) * 0.8)

y_train = df['Close'][:train_size]
y_test = df['Close'][train_size:]

exog_train = df['Volume'][:train_size]
exog_test = df['Volume'][train_size:]
```

# ARIMAX

Forecast Comparison of 3 ARIMAX Models



# ARIMAX

**Đánh giá hiệu suất của 3 mô hình bằng các chỉ số AIC, BIC, RMSE**

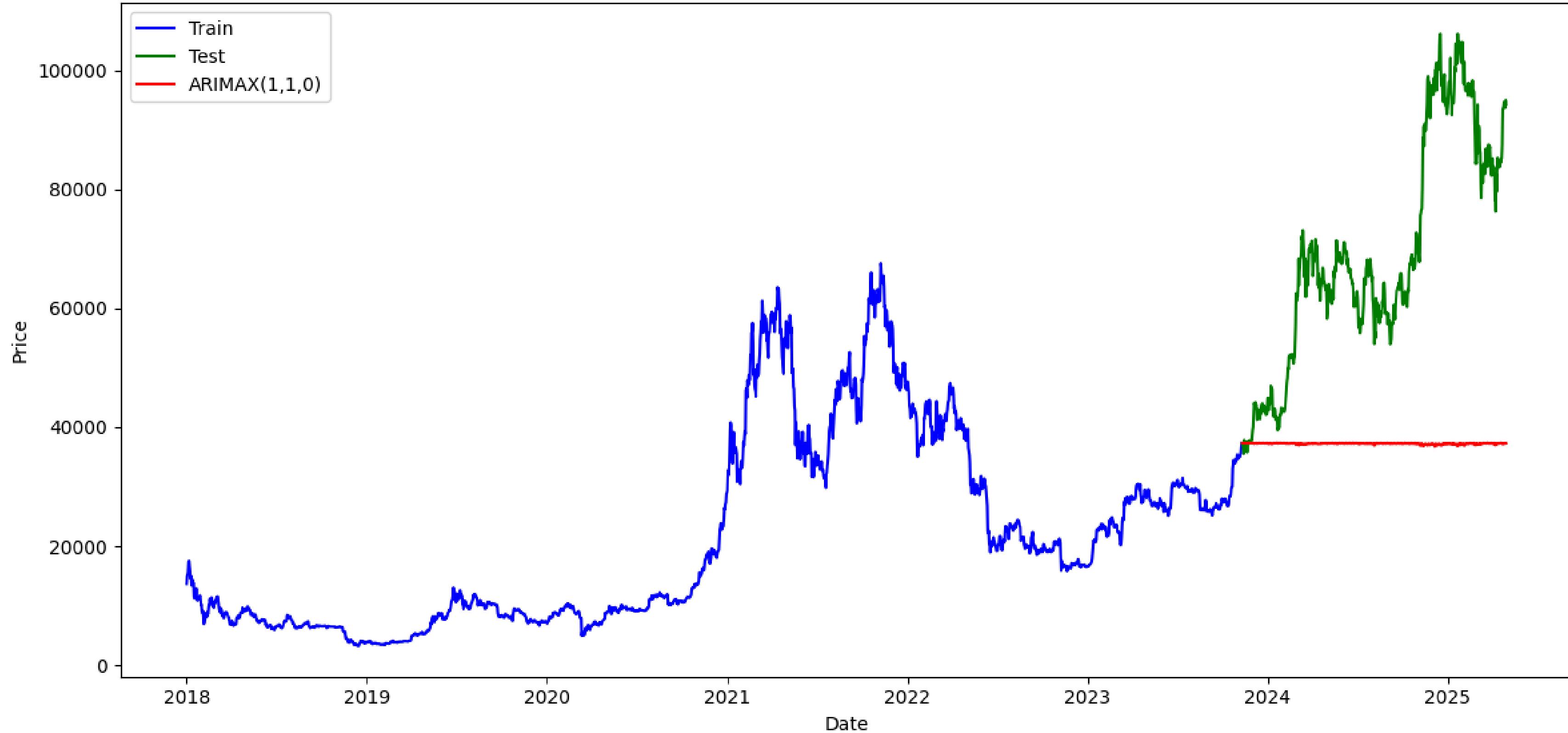
Model	AIC	BIC	RMSE
ARIMAX(1,1,0)	35438.266133	35455.270414	37194.015014
ARIMAX(0,1,1)	35438.311418	35455.315699	37195.108246
ARIMAX(1,1,1)	35440.271858	35462.944233	37194.115101

Chọn mô hình **ARIMAX(1,1,0)**. Vì mô hình này tốt nhất về tất cả các tiêu chí AIC, BIC, RMSE.

# ARIMAX

## Mô hình ARIMAX (1,1,0) dự báo trên tập test

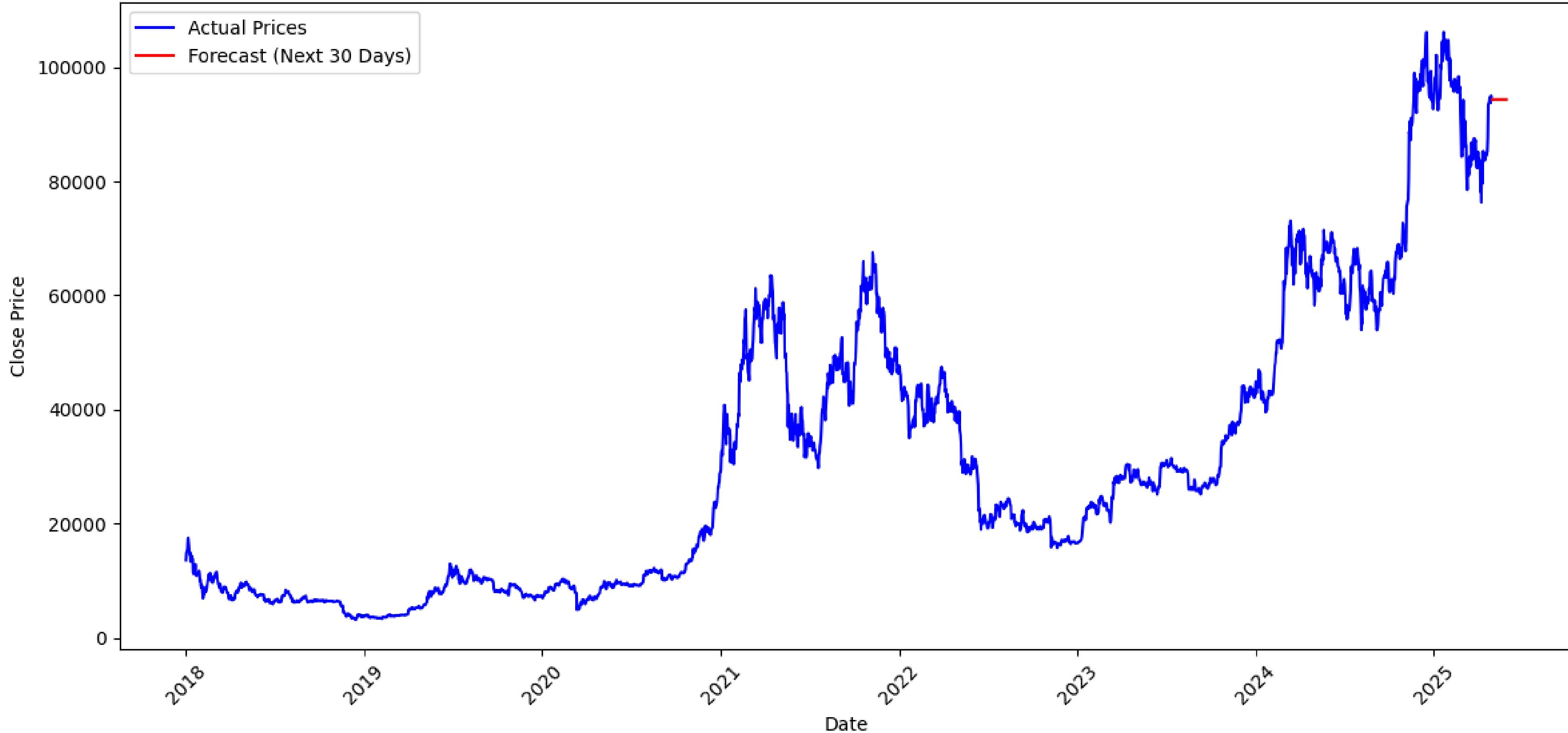
Bitcoin forecasting on test set with ARIMAX (1,1,0) model



# ARIMAX

## Dự báo 30 ngày tiếp theo trong tương lai

Bitcoin Price Forecast Using ARIMAX(1,1,0)



# ARIMA & ARIMAX

## So sánh giữa ARIMA (1,1,1) và ARIMAX (1,1,0) được chọn

Model	AIC	BIC	RMSE
ARIMA(1,1,1)	35443.963081	35460.967362	37113.260644
ARIMAX(1,1,0)	35438.266133	35455.270414	37194.015014

☒ **AIC:** ARIMAX(1,1,0) có AIC = 35438.27 < ARIMA(1,1,1) = 35443.96 → ARIMAX tốt hơn theo AIC

☒ **BIC:** ARIMAX(1,1,0) có BIC = 35455.27 < ARIMA(1,1,1) = 35460.97 → ARIMAX tốt hơn theo BIC

☒ **RMSE:** ARIMA(1,1,1) có RMSE = 37113.26 < ARIMAX(1,1,0) = 37194.02 → ARIMA tốt hơn theo RMSE

- Cả hai mô hình đều chưa theo kịp xu hướng tăng đột biến của giá Bitcoin trong tập test
- Điều này cho thấy cả hai mô hình đều mang tính bảo thủ, có thể vì **đặc điểm của ARIMA/ARIMAX vốn không dự đoán tốt các xu hướng mang tính phi tuyến và đột ngột**

# XGBOOST



× × × ×



× × × ×



# XGBOOST

## Evolution of Tree Algorithms



Decision  
Trees

Bagging

Random  
Forest

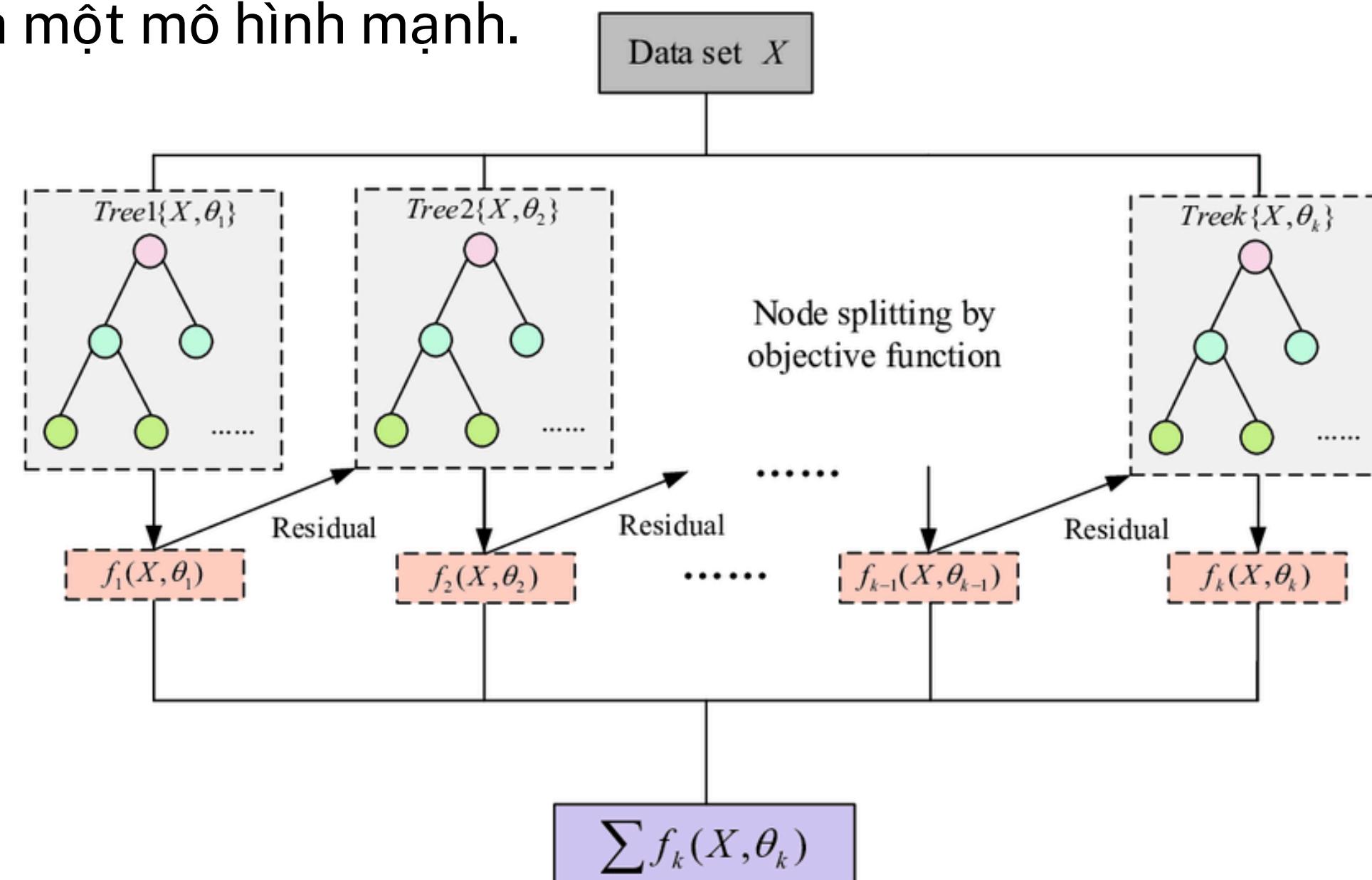
Boosting

Gradient  
Boosting

XG-Boost

# XGBOOST

- XGBoost (Extreme Gradient Boosting) là một thuật toán boosting – huấn luyện nhiều cây quyết định (decision trees) liên tiếp, **mỗi cây mới học từ sai số (residual) của các cây trước.**
- XGBoost là thuật toán theo kiểu mô hình tổng hợp (ensemble) – kết hợp nhiều mô hình yếu để tạo ra một mô hình mạnh.



# XGBOOST

- **Quy trình thực hiện**

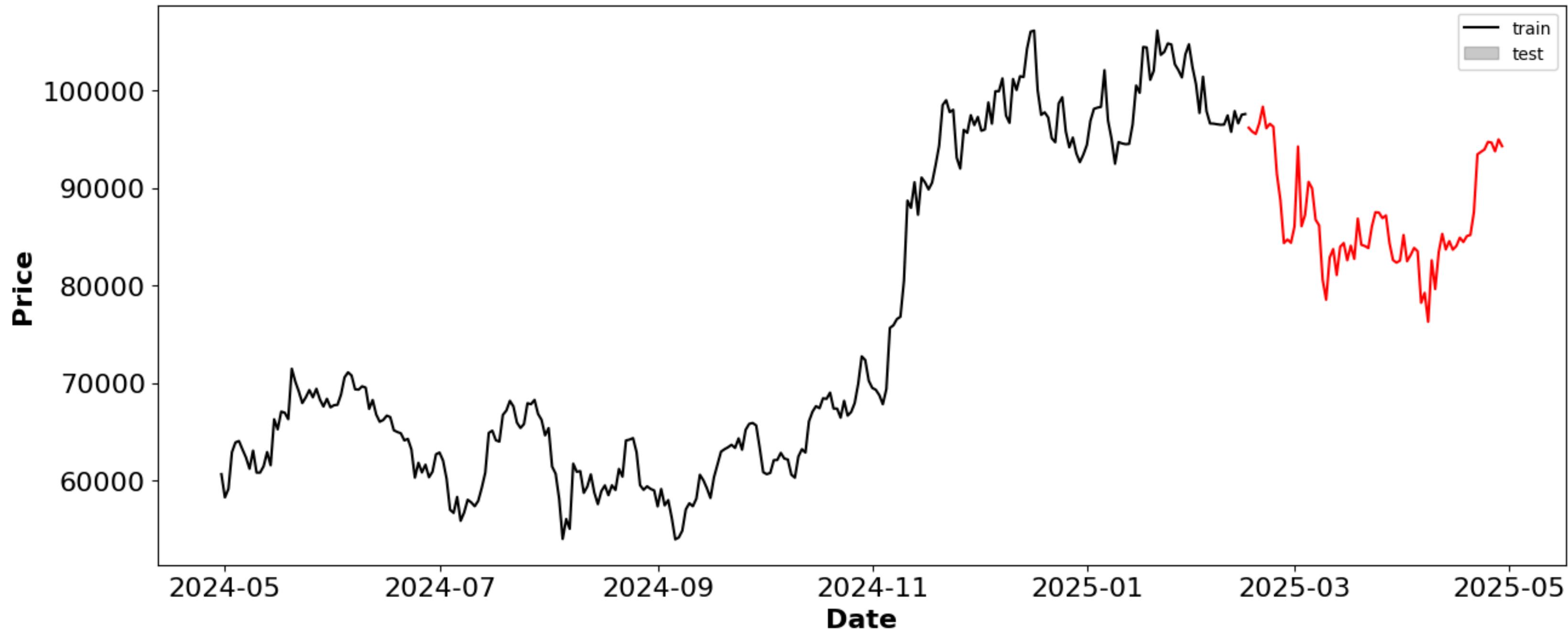
- MinMaxScaler về [0, 1]
- Chia train/test (8:2)
- Tạo tập dữ liệu theo time\_step
- Huấn luyện
- Dự đoán
- Inverse\_transform
- Trực quan hóa chính xác theo thời gian

- **Chỉ giữ lại 1 năm gần đây nhất (từ 30/04/2024 đến 29/04/2025)**

- Xu hướng thị trường hiện tại (gần đây) lại quan trọng hơn cho dự đoán ngắn hạn.
- XGBoost rất nhạy cảm với các xu hướng dài hạn. Nếu dùng dữ liệu 2018-2025 thì các giai đoạn giá thấp có thể làm nhiễu mô hình.
- Giảm độ phức tạp tính toán – huấn luyện nhanh hơn.

# XGBOOST

**Train & Test data**



# XGBOOST

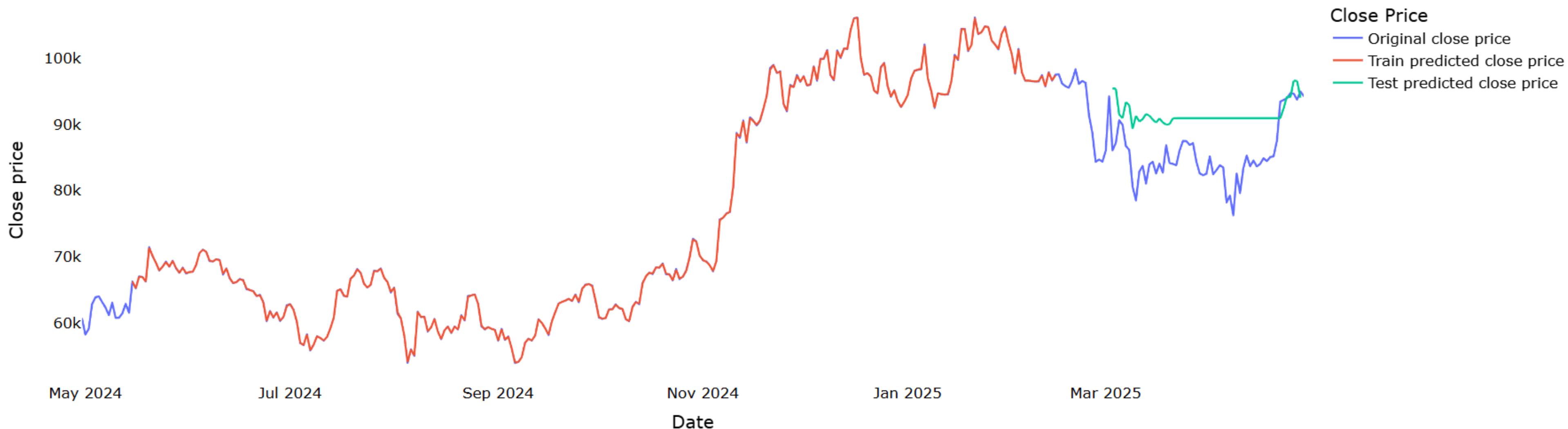
- Timesteps = 15 days
- Output = 1 days (giá ngày kế tiếp)
- Input.shape = (15, 1) với 1 feature là “Close”

```
Mean Absolute Error - MAE : 0.12244672550359528
Root Mean squared Error - RMSE : 0.13633636936619778
```

- Với kết quả MAE và RMSE cho thấy:
  - Mô hình XGBoost dự báo giá khá tốt với chỉ 1 feature (close) và dữ liệu chuẩn hóa.
  - Sai số thấp cho thấy mô hình đáng tin cậy.

# XGBOOST

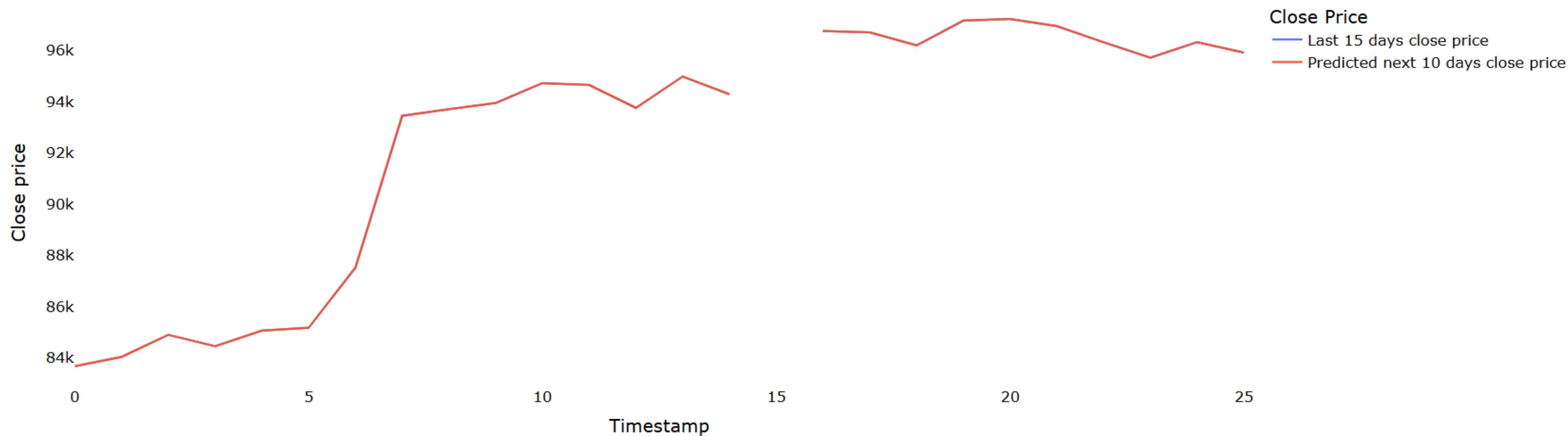
Comparision between original close price vs predicted close price



- Biểu đồ test khá "phẳng" trong khi thực tế có dao động.
- Mô hình không bắt được xu hướng hoặc dao động thực tế trong dữ liệu test.

# XGBOOST

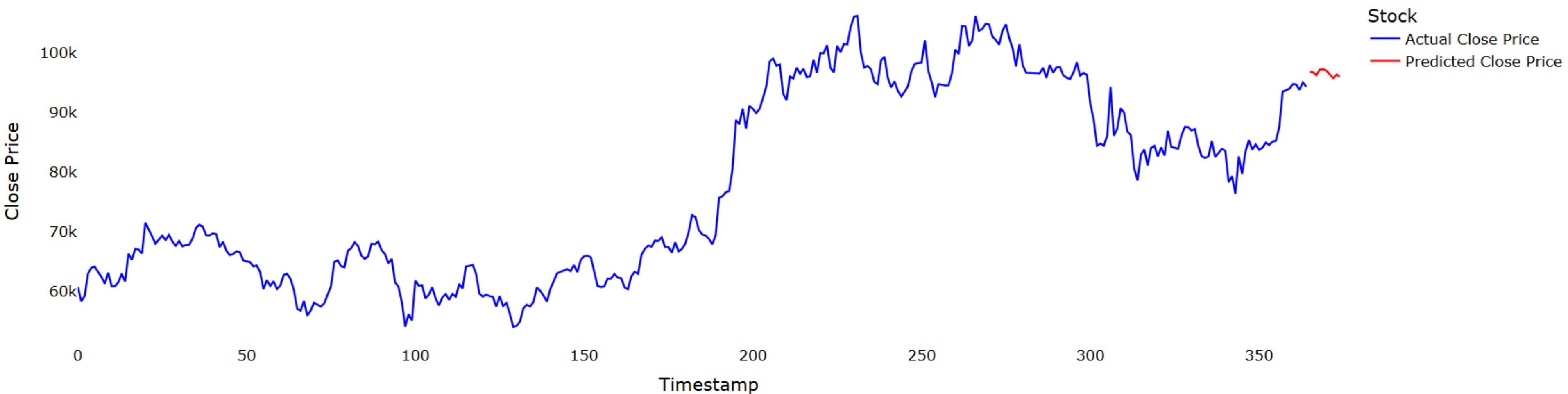
Compare last 15 days vs next 10 days



- Mô hình đã học được xu hướng tăng nhẹ của 15 ngày gần đây nhất và tiếp tục duy trì xu hướng này trong dự báo 10 ngày tiếp theo.
- Biểu đồ thể hiện quá trình dự báo mượt mà, không có đột biến bất thường.

# XGBOOST

Plotting whole closing price with prediction



- Đường dự đoán (màu đỏ) dường như bám khá sát vào xu hướng thực tế ở phần cuối, cho thấy mô hình có khả năng học được xu hướng ngắn hạn.
- Không có dao động quá mạnh ở đoạn dự đoán.

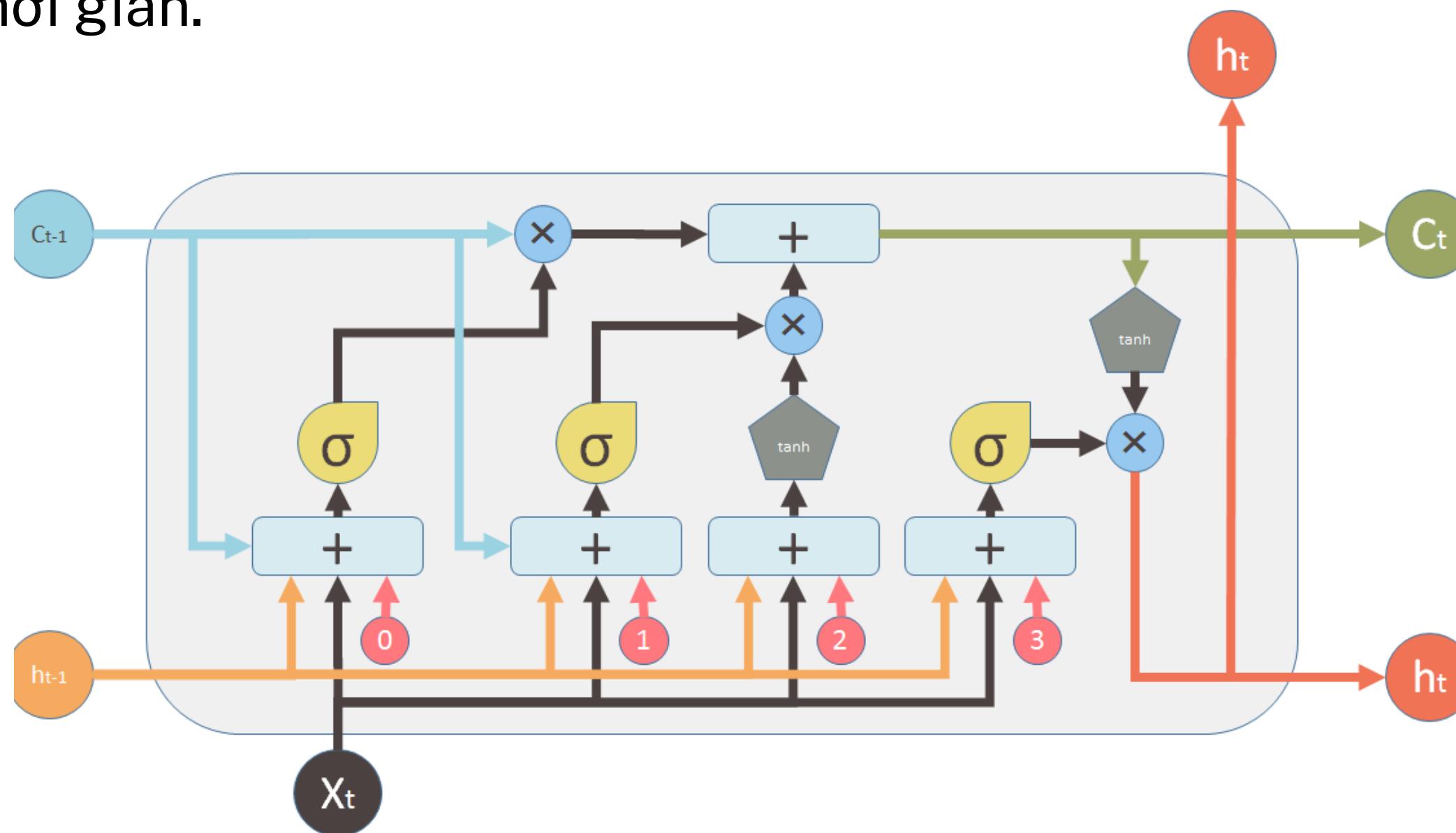
# XGBOOST

- **Ưu điểm:**
  - Mô hình dường như đang dự đoán tốt trong ngắn hạn (biểu đồ đường màu đỏ khớp tốt với phần cuối chuỗi)
- **Nhược điểm:**
  - Dự báo chỉ được vài điểm (7-10), cho thấy mô hình chưa khai thác được xu hướng dài hạn.
  - Phần dự đoán có vẻ phẳng hoặc dao động nhẹ, có thể do không đủ đặc trưng thời gian.

# LSTM

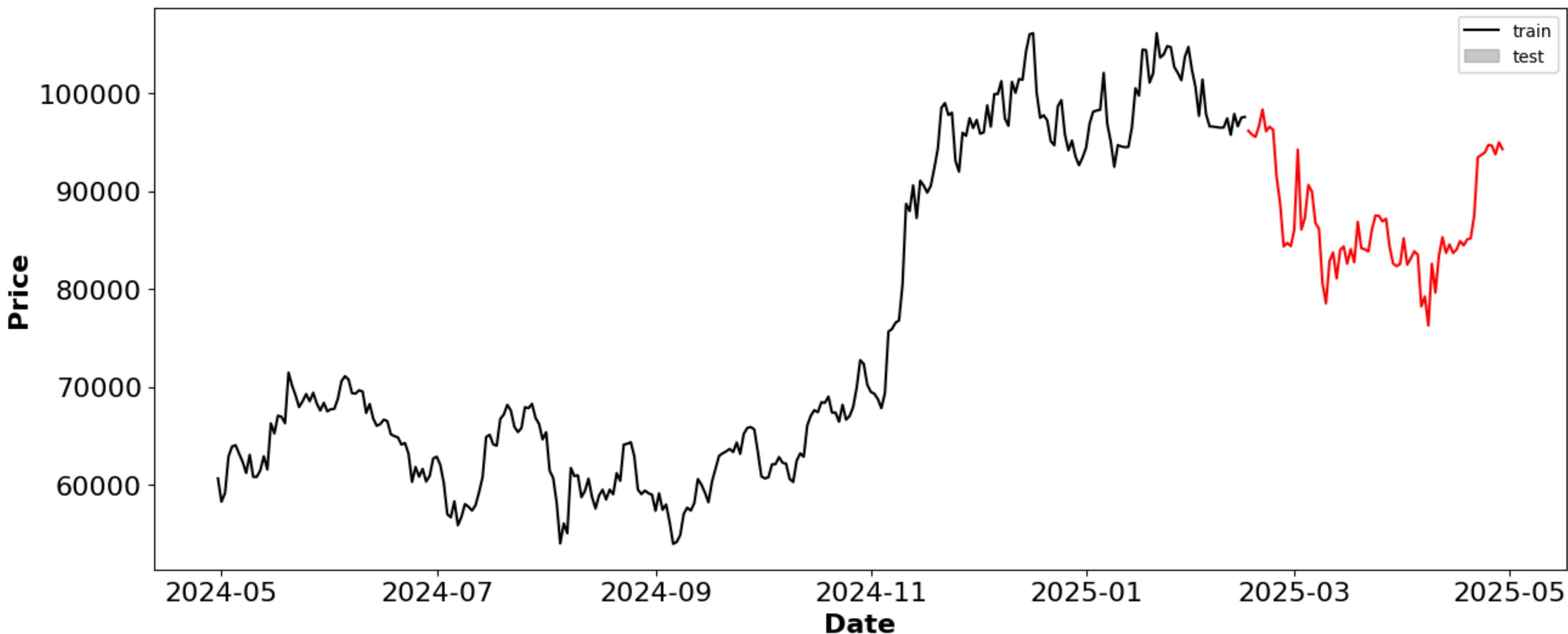
# LSTM

**LSTM (Long Short-Term Memory):** là một kiến trúc đặc biệt của Mạng Nơ-ron hồi quy (RNN) được thiết kế để giải quyết vấn đề vanishing gradient và có khả năng học được các dependency dài hạn trong dữ liệu chuỗi thời gian.



# LSTM

Dữ liệu: 80% train, 20% test,  
1 feature: giá đóng phiên



# LSTM

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 15, 50)	10,400
dropout (Dropout)	(None, 15, 50)	0
lstm_1 (LSTM)	(None, 15, 50)	20,200
dropout_1 (Dropout)	(None, 15, 50)	0
lstm_2 (LSTM)	(None, 50)	20,200
dropout_2 (Dropout)	(None, 50)	0
dense (Dense)	(None, 1)	51

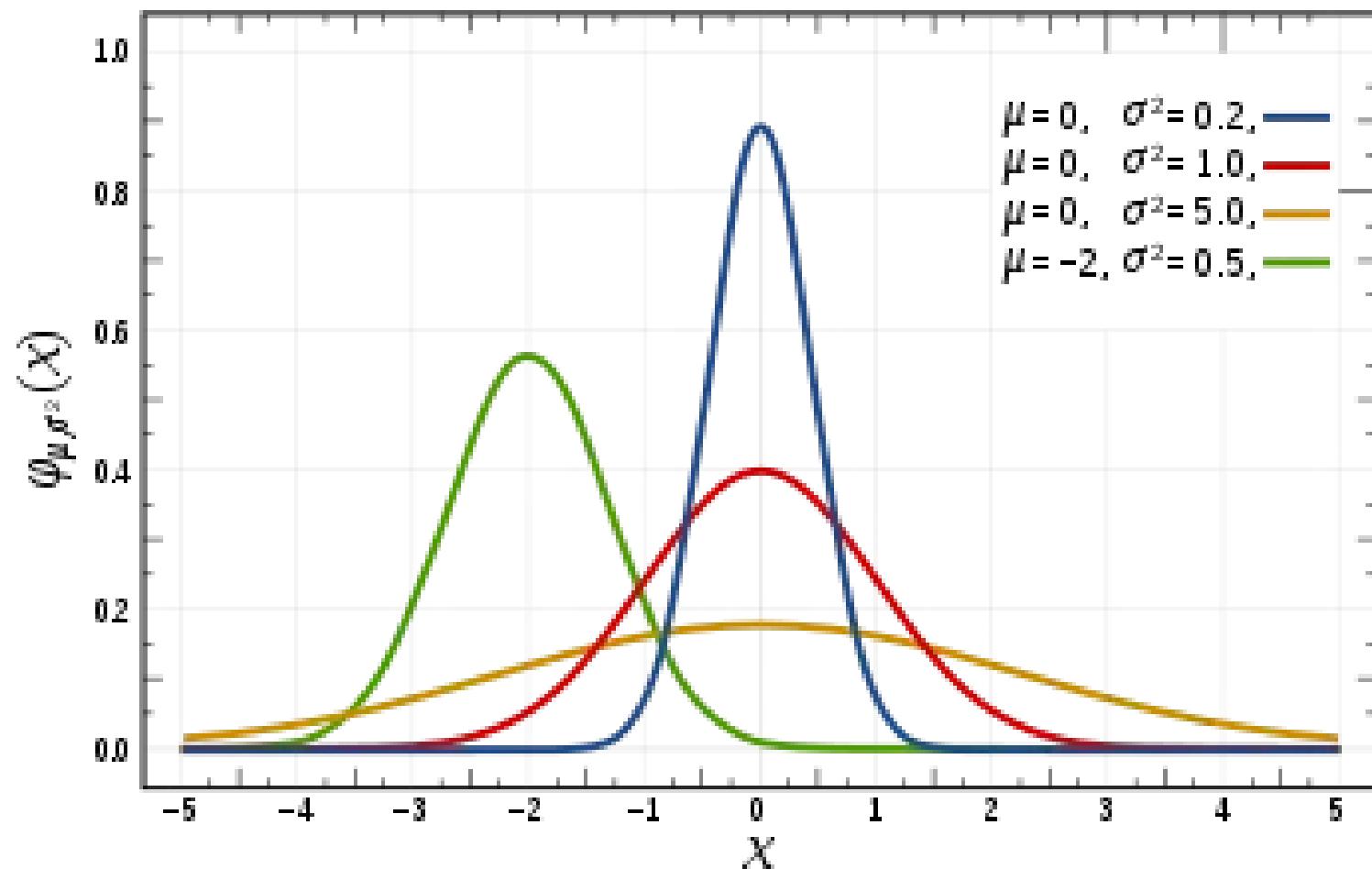
Total params: 50,851 (198.64 KB)

## Cấu trúc mạng LSTM sử dụng

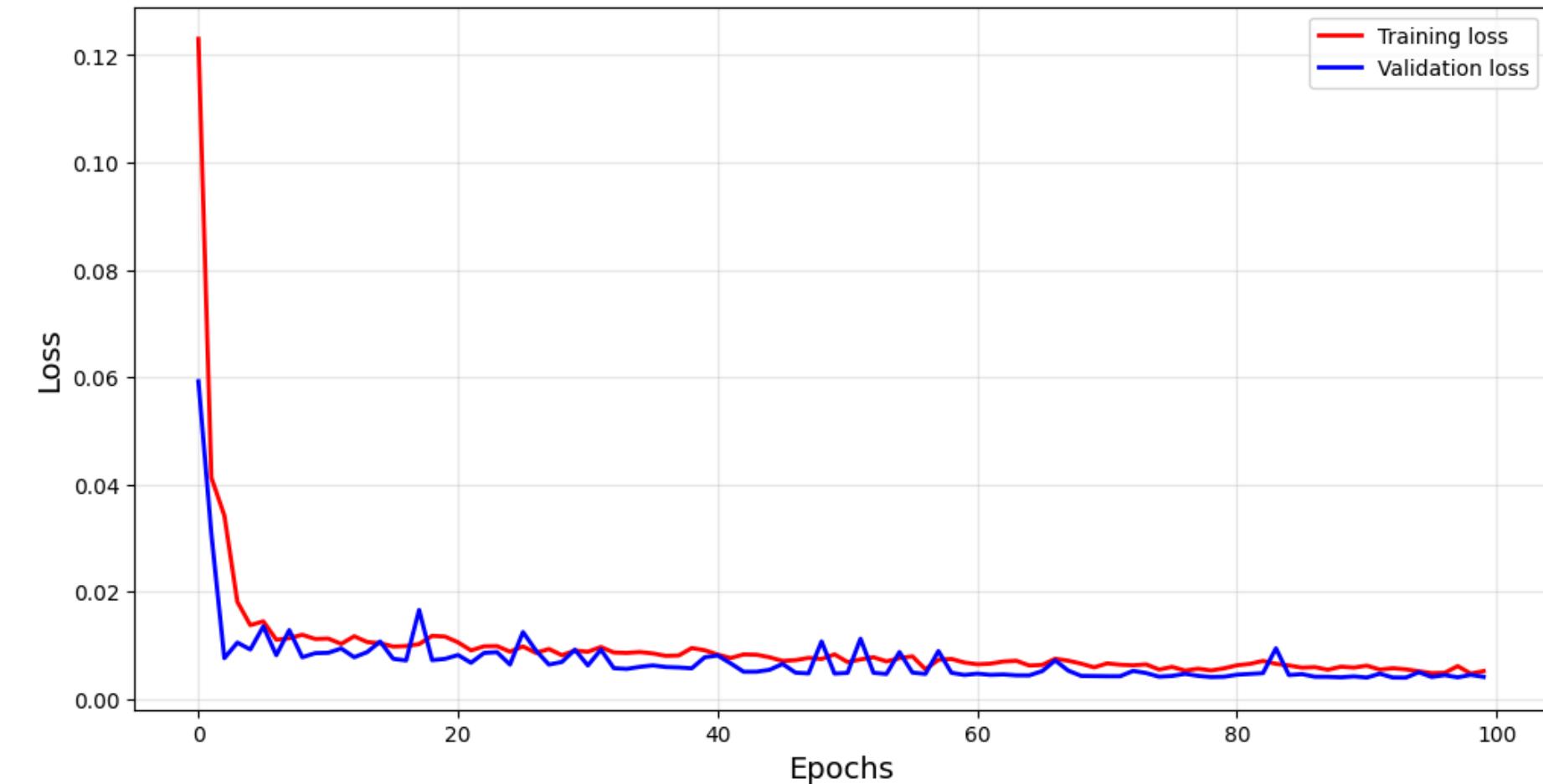
- 3 layers x (50-cell LSTM)
- 3 x Drop-out block (20%)

# LSTM

## Huấn luyện mô hình



Min-max scaling: chuẩn hóa  
dữ liệu trong khoảng 0-1



- Huấn luyện 100 epochs
- Timestep: 15 ngày

# LSTM

## ==== ĐÁNH GIÁ MÔ HÌNH LSTM ====

Train data RMSE: 3221.9544230034007

Train data MSE: 10380990.303911177

Train data MAE: 2456.8050554800725

Train data R2 score: 0.9628876453903508

-----

Test data RMSE: 3348.2612574331815

Test data MSE: 11210853.44802803

Test data MAE: 2540.4131030701756

Test data R2 score: 0.369691054919267

## Đánh giá mô hình

- Kết quả tích cực trên tập train:
    - MAE, RMSE ổn so với giá BTC/USD
    - R2 score cao
  - Kết quả chưa tốt trên tập test:
    - Error tăng nhẹ so với train
    - **R2 score** giảm mạnh xuống ~0.369, cho thấy mô hình đã bị overfitting trên tập train
- Mức độ tổng quát hóa mô hình chưa đủ tin cậy

# LSTM

## Trực quan hóa kết quả huấn luyện

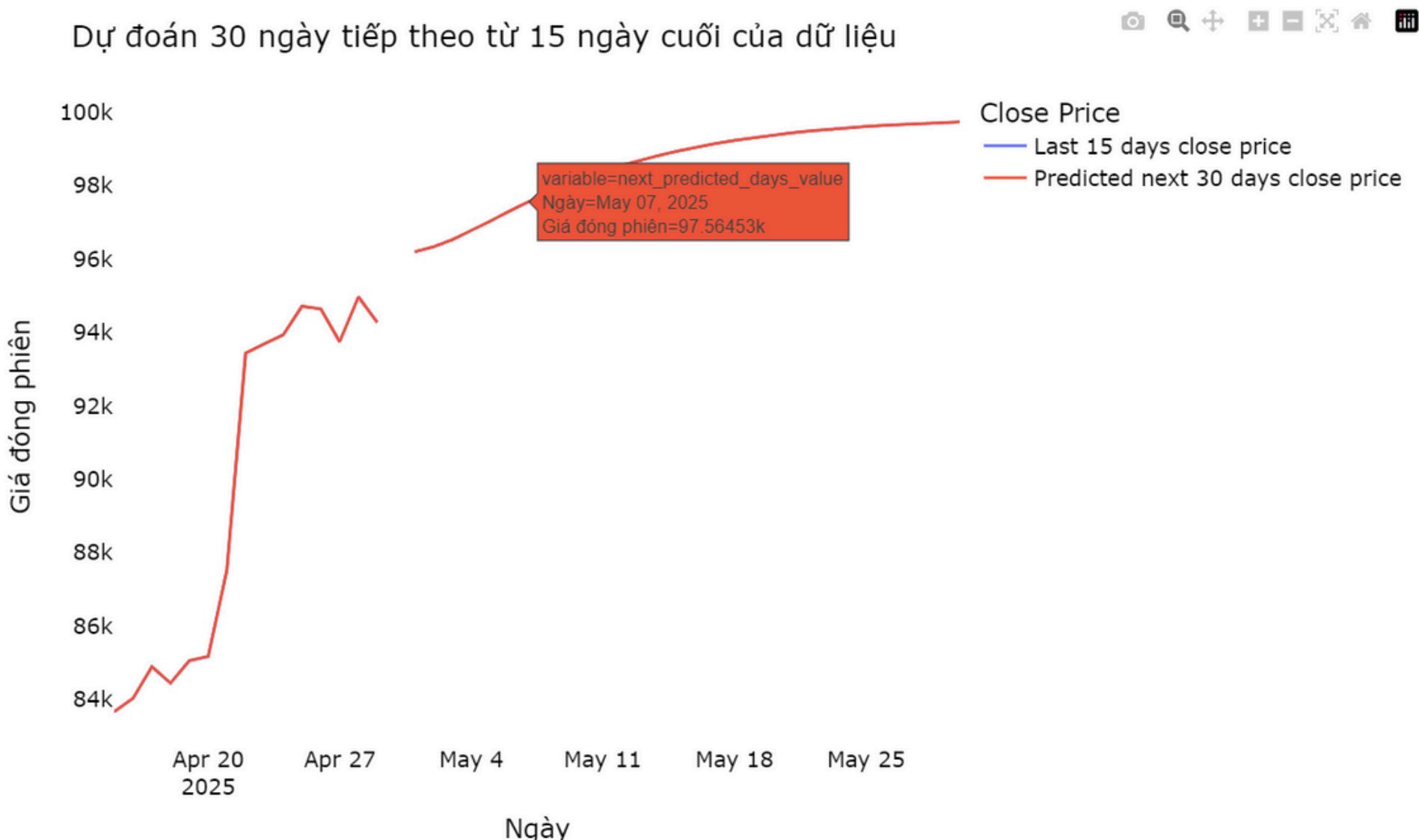
So sánh giá đóng cửa thực tế vs dự đoán bằng mô hình LSTM



# LSTM

## Thực nghiệm: dự đoán giá đóng phiên 30 ngày tiếp theo dựa vào 15 ngày trước đó

Dự đoán 30 ngày tiếp theo từ 15 ngày cuối của dữ liệu



Ngày	Giá Dự Đoán (USD)	Thay Đổi so với Hôm Trước
30/04/2025	\$96,205.18	--
01/05/2025	\$96,343.10	+0.14%
02/05/2025	\$96,536.12	+0.20%
03/05/2025	\$96,775.35	+0.25%
04/05/2025	\$97,037.64	+0.27%
05/05/2025	\$97,304.86	+0.28%
06/05/2025	\$97,564.53	+0.27%
07/05/2025	\$97,809.03	+0.25%
08/05/2025	\$98,029.82	+0.23%
09/05/2025	\$98,225.40	+0.20%

# GRU

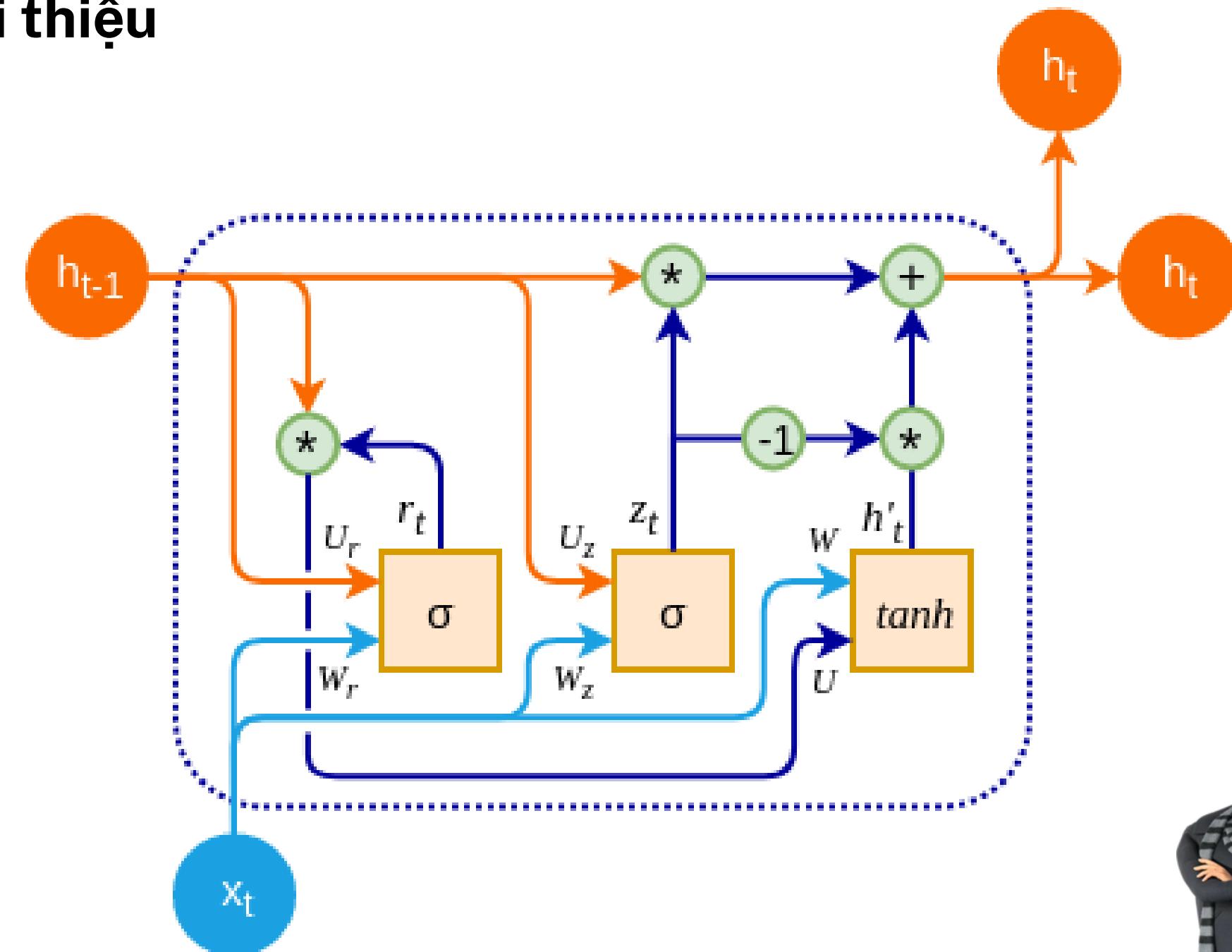
# GRU

## Giới thiệu

**GRU (Gated Recurrent Unit)** là một kiến trúc mạng Hồi quy (RNN).

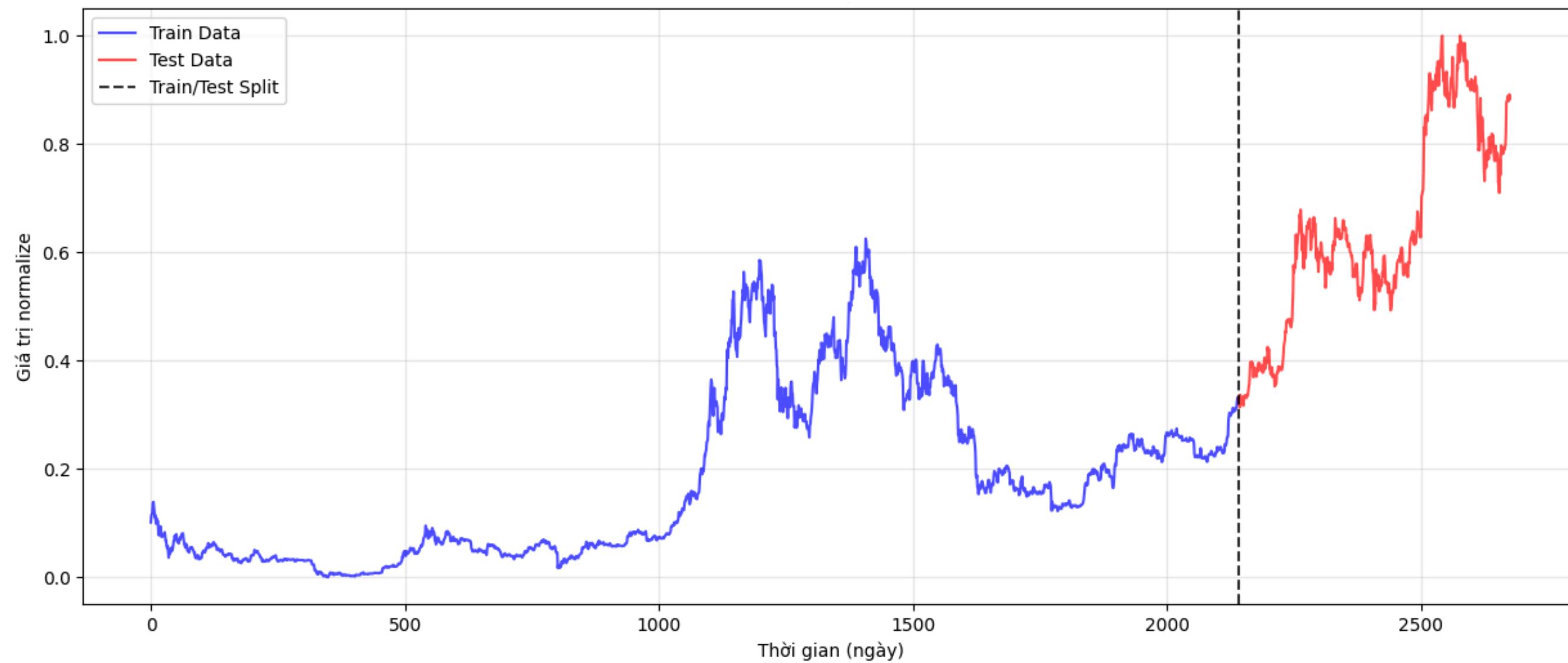
Tương tự như LSTM, GRU được thiết kế để giải quyết vấn đề vanishing gradient và có khả năng học các phụ thuộc dài hạn trong dữ liệu chuỗi.

Điểm nổi bật của GRU là nó có **cấu trúc đơn giản hơn LSTM** nhưng thường mang lại hiệu suất tương đương trên nhiều tác vụ.



# GRU

Dữ liệu: 80% train, 20% test,  
1 feature: giá đóng phiên



# GRU

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 15, 50)	10,400
dropout (Dropout)	(None, 15, 50)	0
lstm_1 (LSTM)	(None, 15, 50)	20,200
dropout_1 (Dropout)	(None, 15, 50)	0
lstm_2 (LSTM)	(None, 50)	20,200
dropout_2 (Dropout)	(None, 50)	0
dense (Dense)	(None, 1)	51

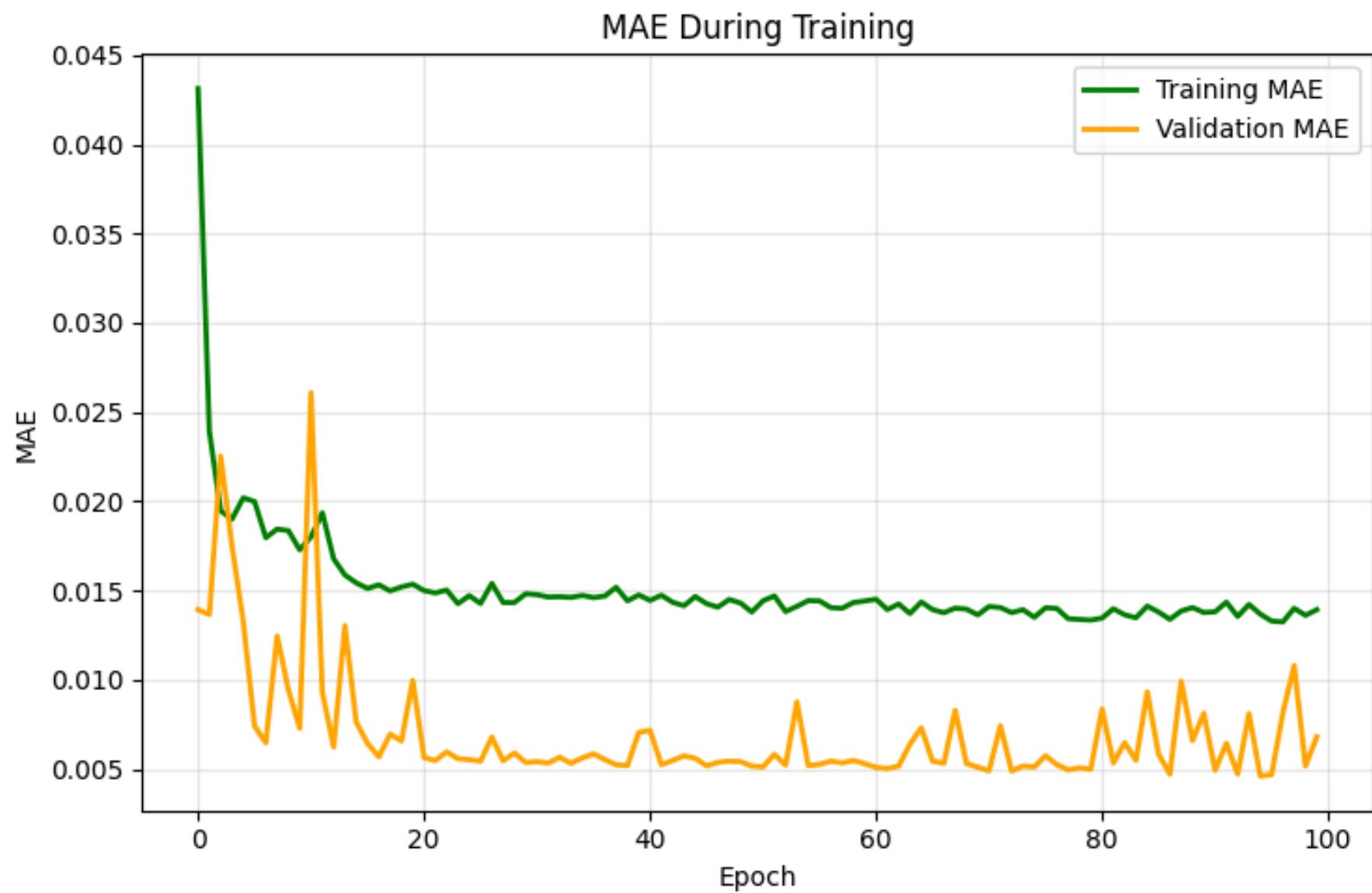
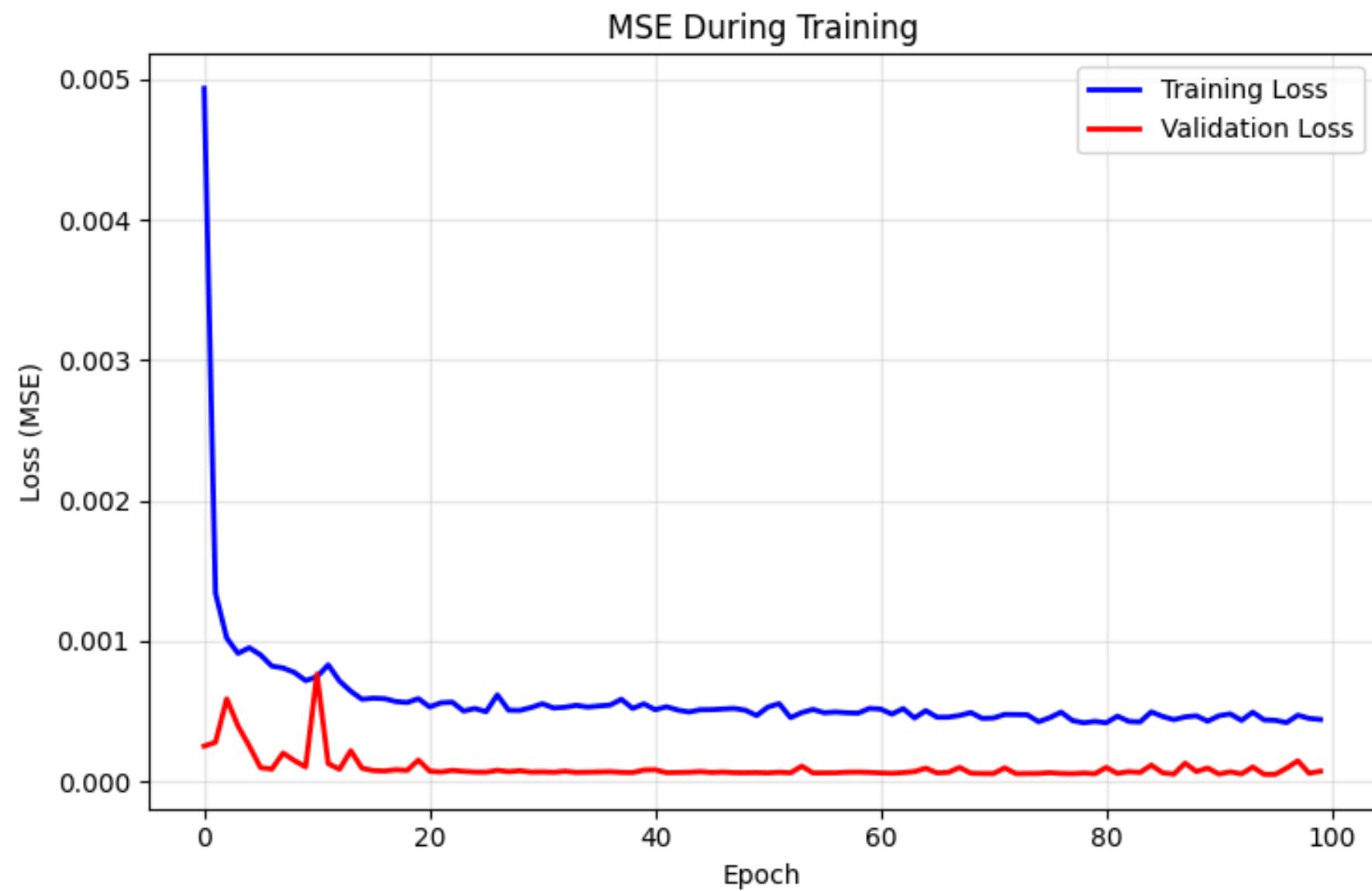
Total params: 50,851 (198.64 KB)

## Cấu trúc mạng GRU sử dụng

- Input: 15 timesteps, 1 feature (giá đóng phiên)
- 3 layers x (50-cell LSTM): gồm 3 lớp GRU liên tiếp, mỗi lớp có 50 đơn vị (units).
- 3 x Drop-out block (20%): được chèn sau mỗi lớp GRU để tránh overfitting,

# GRU

## Huấn luyện mô hình



# GRU

Kết quả đánh giá trên Training Set:

MSE: 1,367,797.58

RMSE: 1,169.53

MAE: 693.64

R<sup>2</sup> Score: 0.9946 (99.46%)

Kết quả đánh giá trên Test Set:

MSE: 11,253,649.66

RMSE: 3,354.65

MAE: 2,556.40

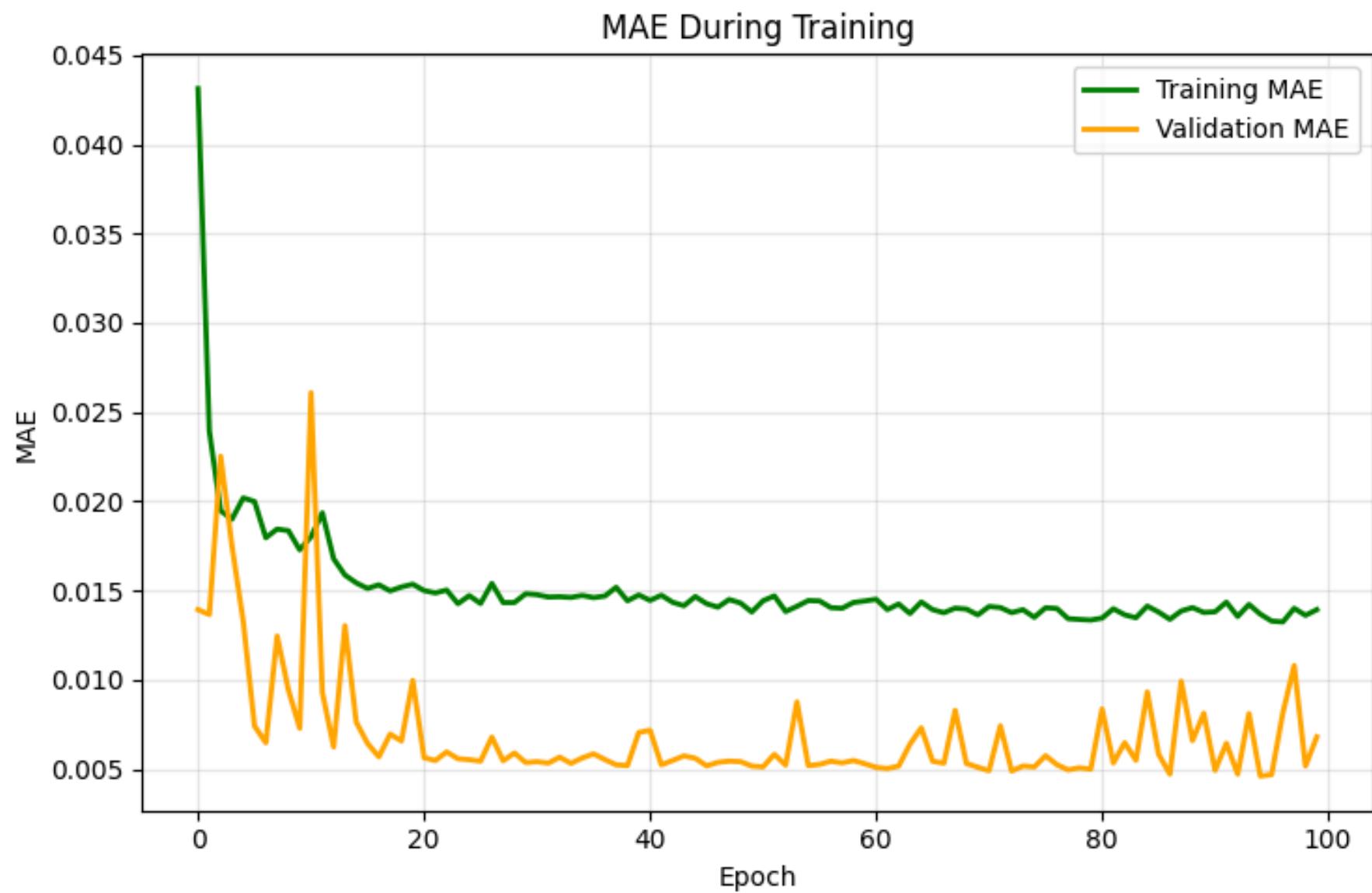
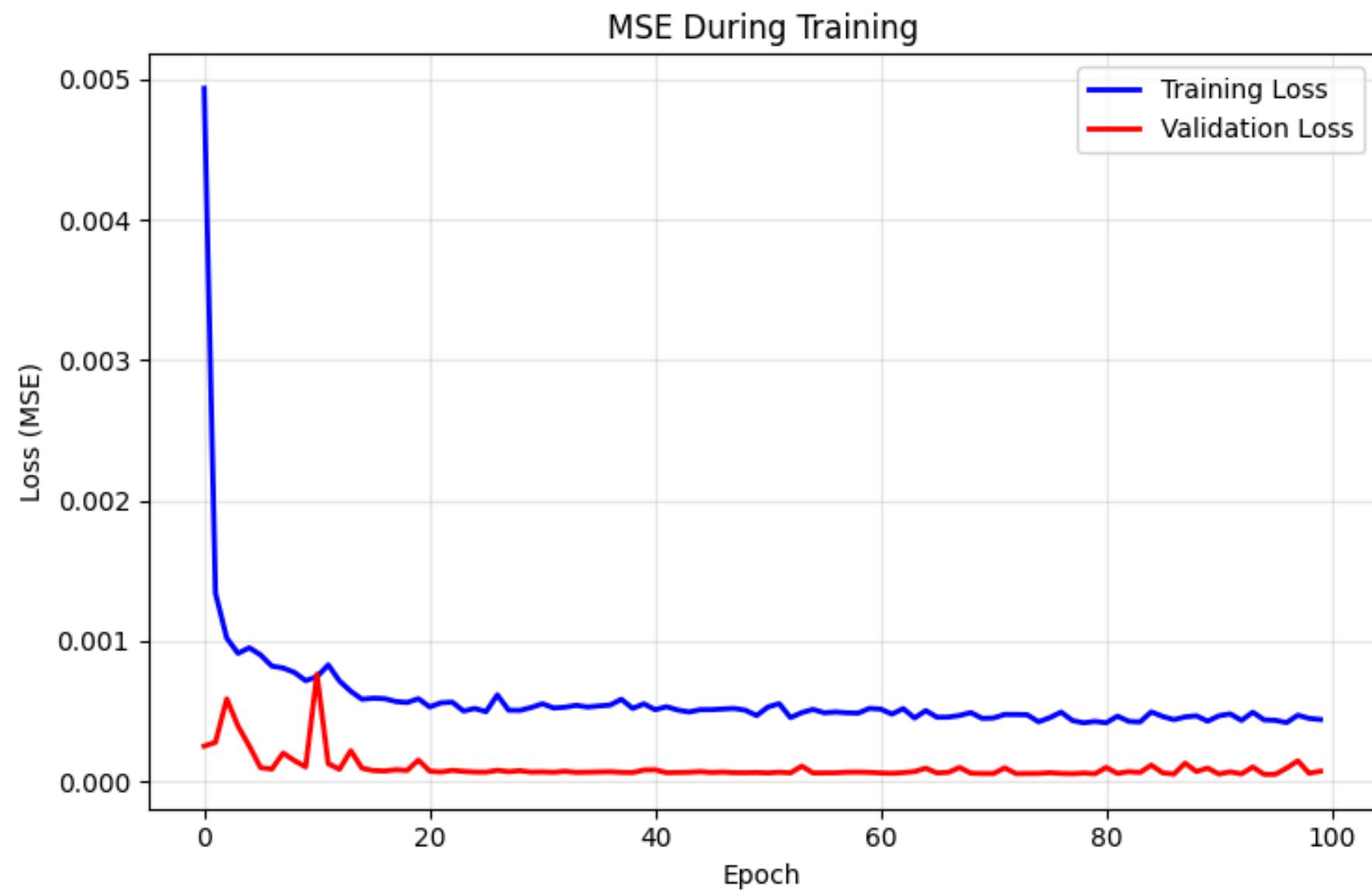
R<sup>2</sup> Score: 0.9686 (96.86%)

## Đánh giá mô hình

- R<sup>2</sup> > 96% trên cả tập huấn luyện và kiểm tra → mô hình dự báo tốt.
- RMSE & MAE trên tập kiểm tra cao hơn tập huấn luyện (bình thường), cần đánh giá trong bối cảnh giá trị thực tế.
- Mô hình thể hiện khả năng tổng quát hóa tốt từ dữ liệu huấn luyện sang dữ liệu mới (tập kiểm tra).

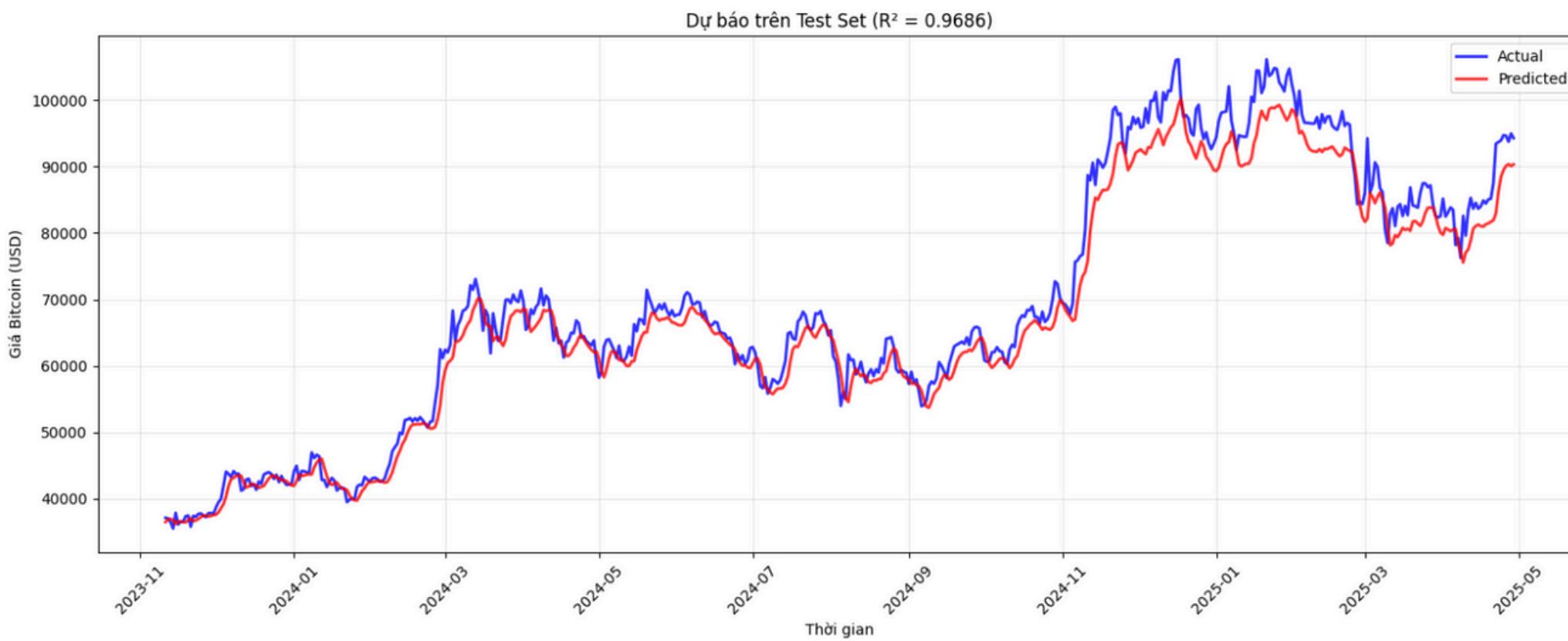
# GRU

## Huấn luyện mô hình



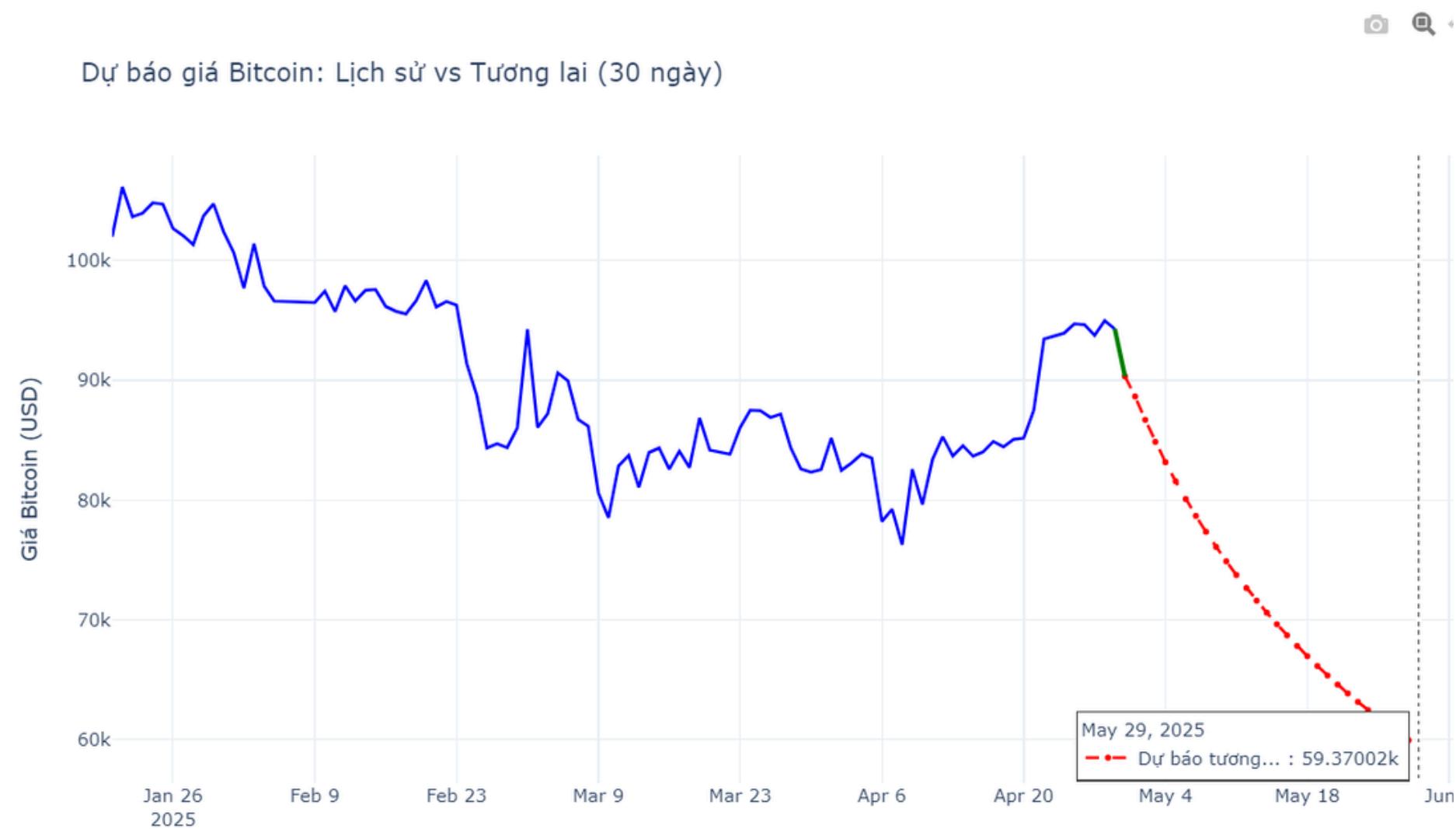
# GRU

## Trực quan hóa kết quả huấn luyện



# GRU

## Thực nghiệm: dự đoán giá đóng phiên 30 ngày tiếp theo dựa vào 15 ngày trước đó



Dự báo chi tiết (10 ngày đầu):

Ngày	Giá dự báo (USD)	Thay đổi từ hiện tại (%)
0 2025-04-30	\$90,329.16	-4.20%
1 2025-05-01	\$88,650.22	-5.98%
2 2025-05-02	\$86,700.18	-8.04%
3 2025-05-03	\$84,861.29	-9.99%
4 2025-05-04	\$83,158.40	-11.80%
5 2025-05-05	\$81,570.78	-13.48%
6 2025-05-06	\$80,081.88	-15.06%
7 2025-05-07	\$78,678.71	-16.55%
8 2025-05-08	\$77,349.98	-17.96%
9 2025-05-09	\$76,088.75	-19.30%

# TRANSFORMER

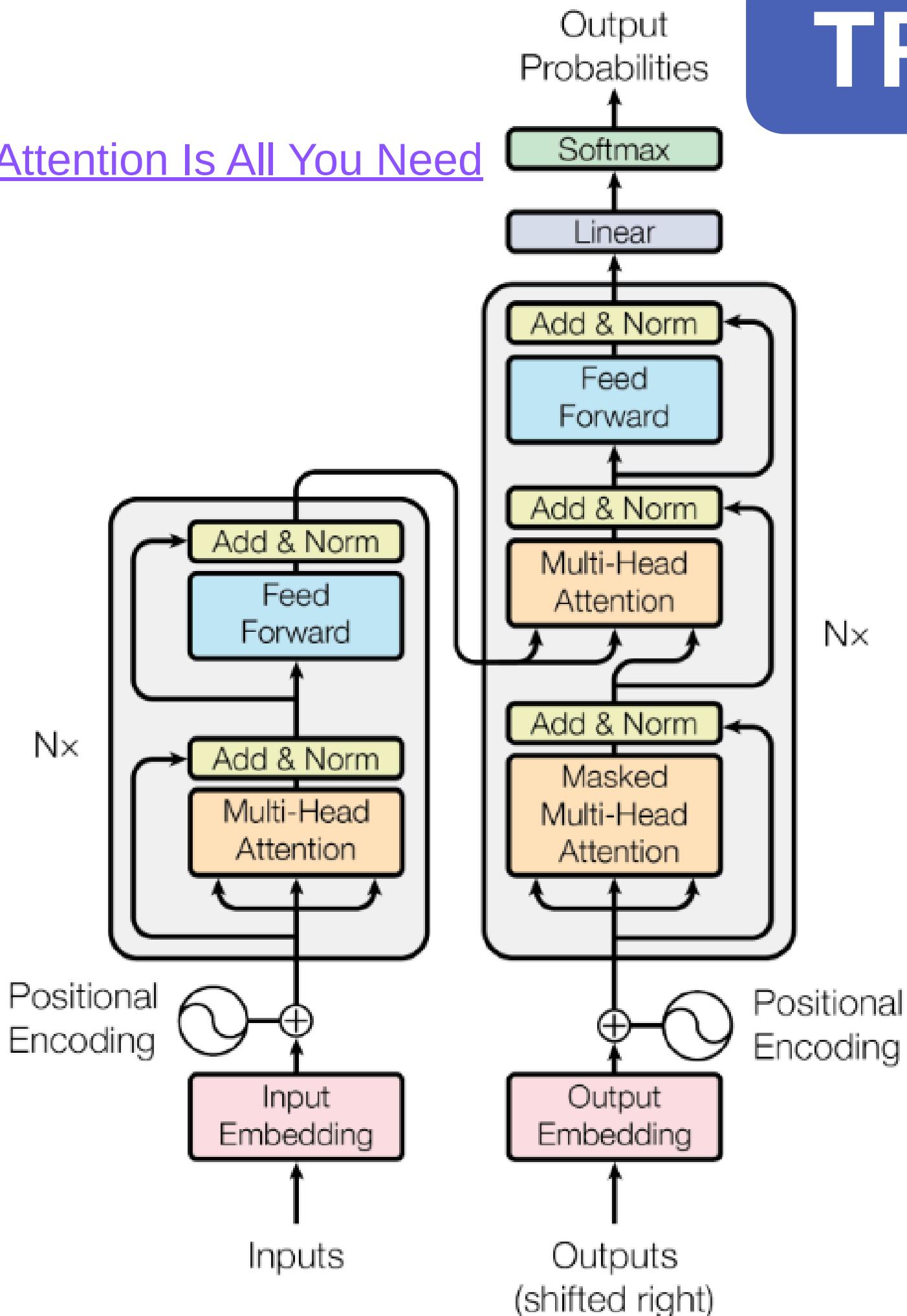


× × ×



# TRANSFORMER

[Attention Is All You Need](#)



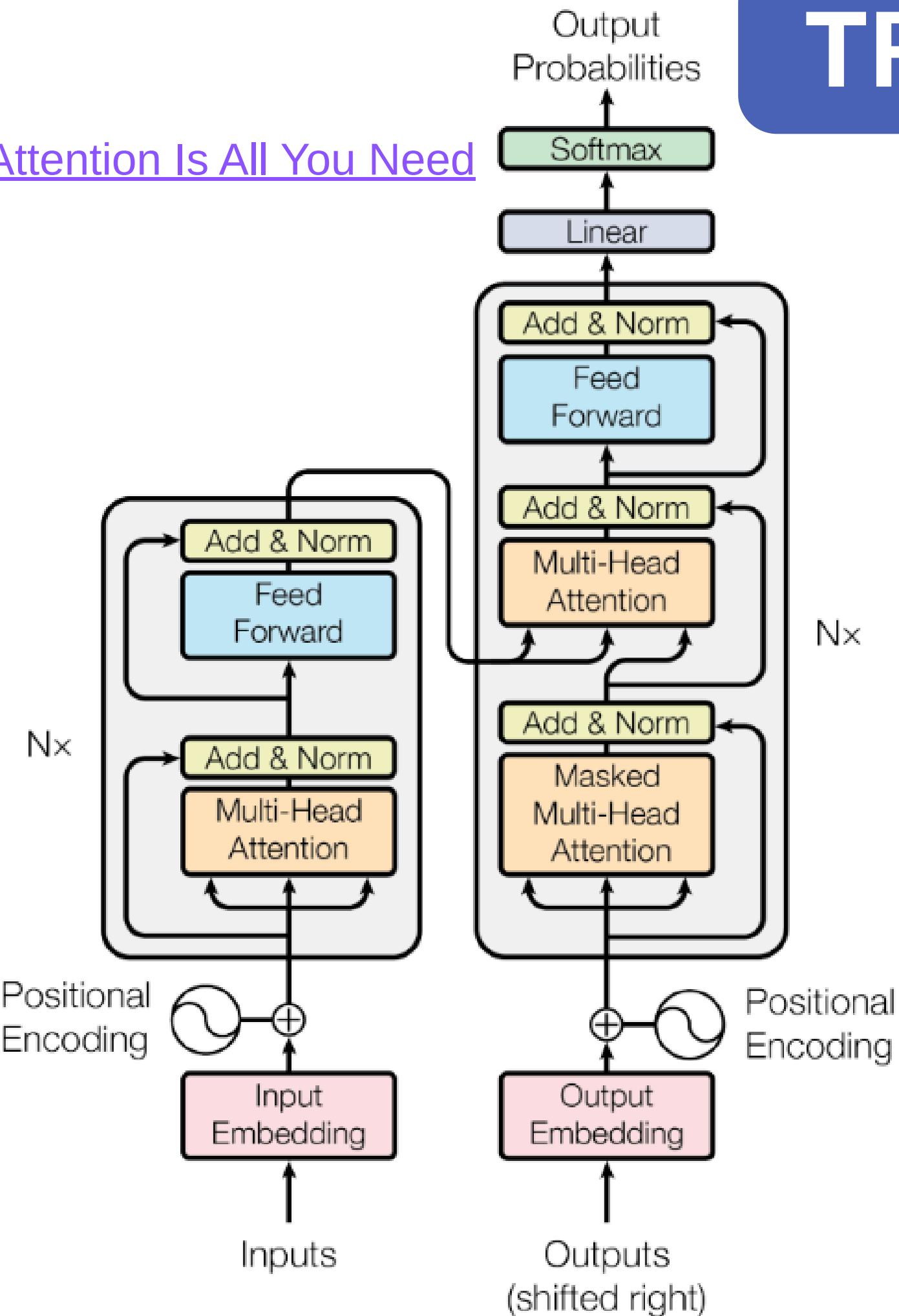
**Transformer có kiến trúc encoder - decoder:**

## 1. Bộ mã hóa (Encoder):

- Mục tiêu: Hiểu toàn bộ chuỗi đầu vào và mã hóa nó thành một biểu diễn ngữ cảnh phong phú.
- Quá trình:
  - Đầu vào (chuỗi token) được nhúng thành vector (embedding).
  - Thêm thông tin vị trí (positional encoding) để mô hình biết được thứ tự các token.
  - Dữ liệu đi qua một loạt các Encoder Layer, mỗi lớp gồm:
    - Multi-Head Self-Attention: Giúp mỗi token "nhìn" toàn bộ chuỗi đầu vào.
    - Feed-Forward Network: Xử lý phi tuyến tính độc lập trên từng vị trí.
    - Residual Connection + Layer Normalization.

# TRANSFORMER

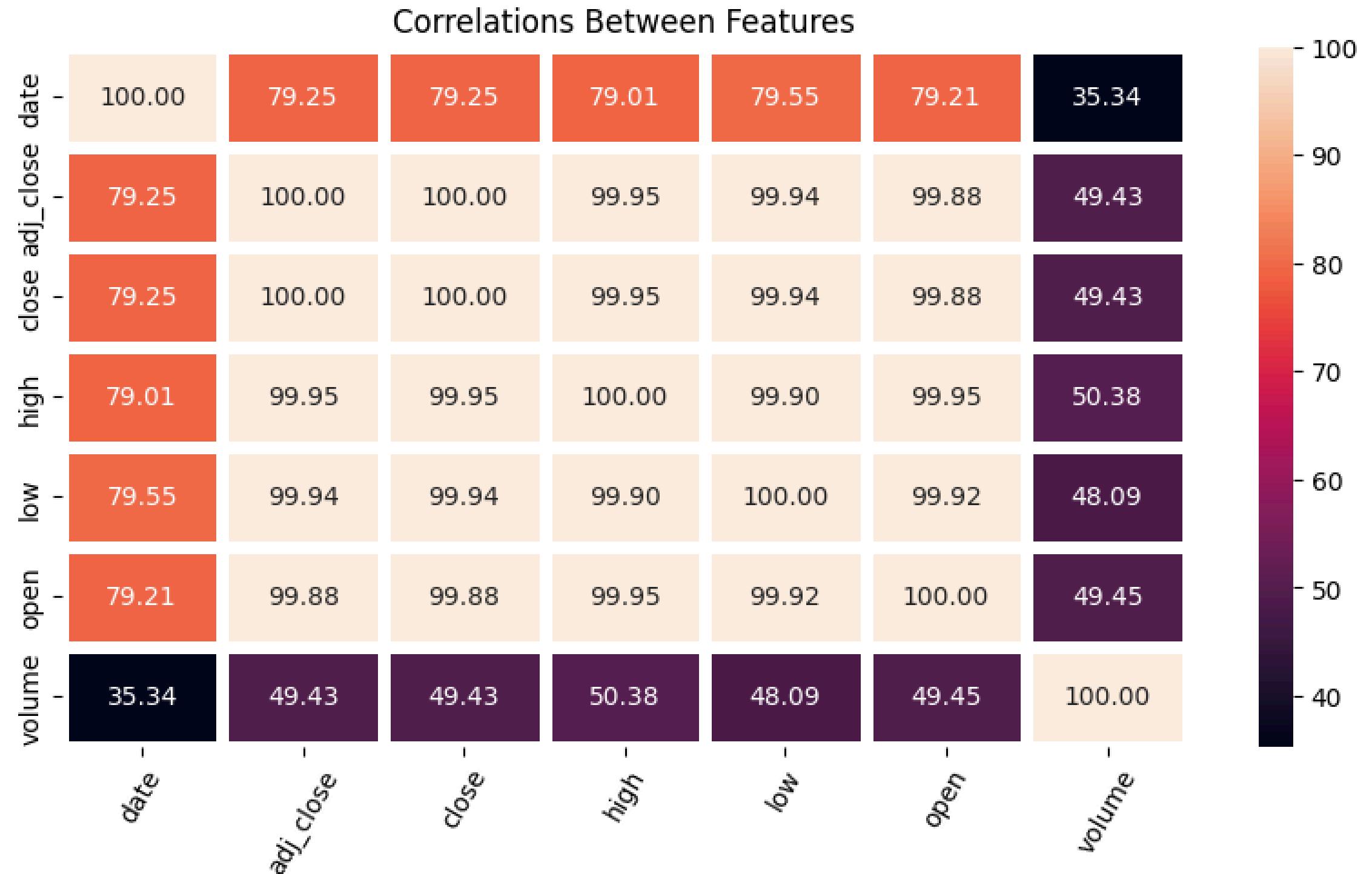
## Attention Is All You Need



## 2. Bộ giải mã (Decoder):

- Mục tiêu: Tạo ra chuỗi đầu ra từng token một cách tự hồi quy (autoregressive).
- Quá trình:
  - Nhận biểu diễn liên tục từ Encoder.
  - Giống như Encoder, mỗi Decoder Layer gồm:
    - Masked Multi-Head Self-Attention: Chỉ cho phép "nhìn" những token trước đó để tránh "nhìn trước" (future tokens).
    - Encoder–Decoder Attention: Giúp Decoder truy cập thông tin từ biểu diễn Encoder.
    - Feed-Forward Network, Residual, và Layer Norm như trên.
- Ở mỗi bước, Decoder sinh ra một token và sử dụng nó làm đầu vào cho bước tiếp theo.

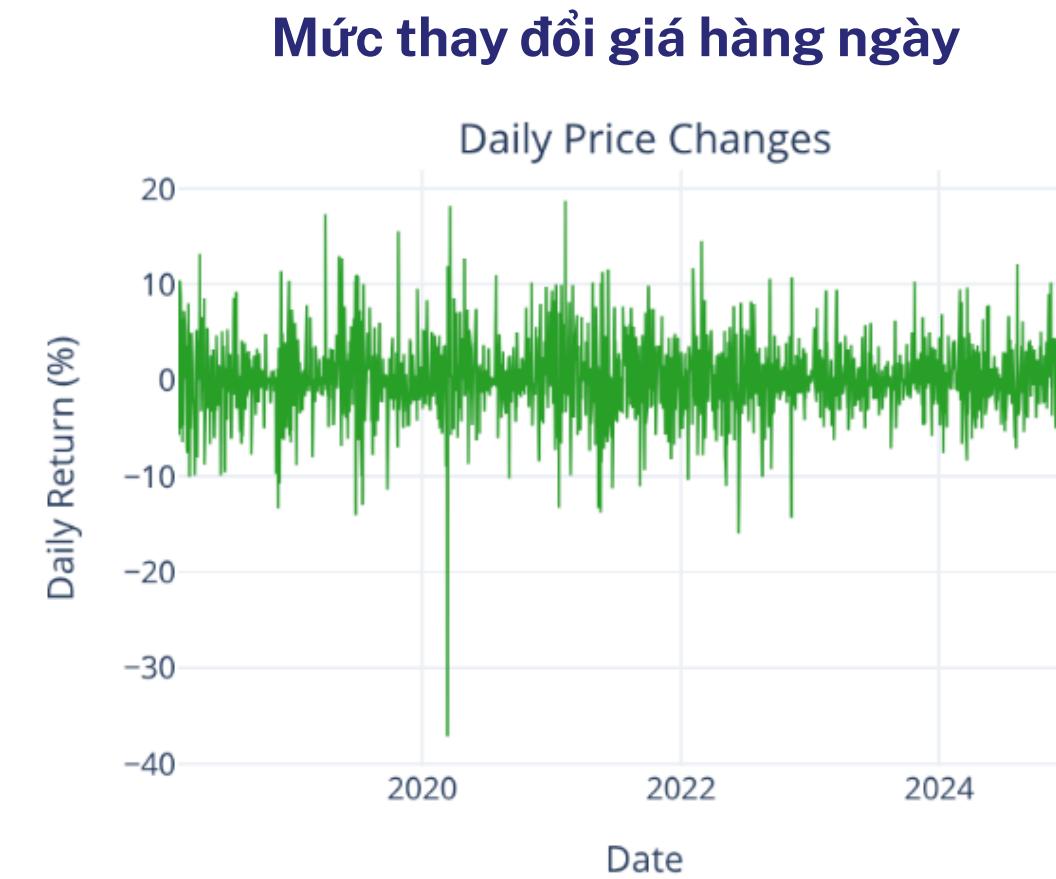
# TRANSFORMER



## Mức độ tương quan của bộ dữ liệu về giá:

- Các cặp đặc trưng giá Close, High, Low, Open có hệ số tương quan gần 100% (~ 99.9)
- Khối lượng giao dịch (Volume) có mối liên hệ yếu hơn với giá, dao động khoảng 35 - 50%
- Hệ số tương quan của Date và Volume chỉ khoảng 35.34%

# TRANSFORMER



# TRANSFORMER

```
RangeIndex: 2676 entries, 0 to 2675
Data columns (total 24 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   date             2676 non-null    datetime64[ns]
 1   adj_close        2676 non-null    float64
 2   close            2676 non-null    float64
 3   high             2676 non-null    float64
 4   low              2676 non-null    float64
 5   open             2676 non-null    float64
 6   volume           2676 non-null    int64  
 7   RSI              2675 non-null    float64
 8   MACD_MACD       2676 non-null    float64
 9   MACD_SIGNAL     2676 non-null    float64
 10  SMA              2636 non-null    float64
 11  EMA              2676 non-null    float64
 12  VWAP             2676 non-null    float64
 13  ATR              2663 non-null    float64
 14  ADX              2663 non-null    float64
 15  BBANDS_BB_UPPER 2657 non-null    float64
 16  BBANDS_BB_MIDDLE 2657 non-null    float64
 17  BBANDS_BB_LOWER  2657 non-null    float64
 18  STOCH            2663 non-null    float64
 19  WILLIAMS         2663 non-null    float64
 20  CCI              2675 non-null    float64
 21  MFI              2663 non-null    float64
 22  ROC              2664 non-null    float64
 23  TRIX             2675 non-null    float64
```

## Ngoài ra thêm vào các đặc trưng:

- Các chỉ số kỹ thuật: RSI, MACD, Bollinger Bands, EMA, SMA v.v... từ thư viện FinTA trong Python → phản ánh **tín hiệu thị trường** và **tâm lý giao dịch**

**FinTA Technical Indicators** là tập hợp các chỉ báo kỹ thuật tài chính  
được cung cấp bởi thư viện FinTA trong Python

# TRANSFORMER

	<b>adj_close</b>	<b>close</b>	<b>high</b>	<b>low</b>	<b>open</b>	<b>volume</b>	<b>RSI</b>	<b>MACD_MACD</b>	<b>MACD_SIGNAL</b>	<b>SMA</b>	...	<b>ADX</b>	<b>BBANDS_BB_UPPER</b>	<b>BBANDS_BB_MIDDLE</b>
0	8621.900391	8621.900391	9122.549805	8295.469727	8720.080078	7780960256	39.150896	-1065.753743	-1070.529627	11928.306129	...	59.981173	12726.427357	9692.452612
1	8129.970215	8129.970215	8616.129883	7931.100098	8616.129883	6122189824	36.595308	-1039.102605	-1064.243688	11793.495641	...	58.904041	12604.135462	9552.381104
2	8926.570312	8926.570312	8985.919922	8141.430176	8141.430176	6256439808	43.075259	-944.926461	-1040.378618	11645.799805	...	57.219446	12453.847827	9455.289600
3	8598.309570	8598.309570	8958.469727	8455.410156	8926.719727	5696719872	41.206393	-885.730042	-1009.447219	11484.758575	...	55.683110	12198.627817	9317.235059
4	9494.629883	9494.629883	9518.540039	8599.919922	8599.919922	7909819904	47.858548	-759.805630	-959.516726	11335.866616	...	53.218721	11964.350146	9228.996533
5	10166.400391	10166.400391	10234.799805	9395.580078	9488.320312	9062540288	52.221747	-600.349586	-887.680795	11158.717845	...	49.768054	11798.330634	9178.746533
6	10233.900391	10233.900391	10324.099609	9824.820312	10135.700195	7296159744	52.650497	-463.355611	-802.813392	10980.837367	...	46.462551	11568.740402	9118.406543
7	11112.700195	11112.700195	11139.500000	10149.400391	10207.500000	8660880384	57.942162	-282.279302	-698.704253	10849.986161	...	43.532499	11395.183887	9084.726562
8	10551.799805	10551.799805	11349.799805	10326.000000	11123.400391	8744009728	53.808736	-181.023242	-595.166203	10737.344703	...	41.124420	11228.254164	9047.496533
9	11225.299805	11225.299805	11273.799805	10513.200195	10552.599609	7652089856	57.709871	-46.984218	-485.528242	10655.147127	...	38.912954	11449.720082	9103.446533

# TRANSFORMER

```
class SineActivation(nn.Module):
    """Time2Vector layer that adds periodic and linear features to input data"""

    def __init__(self, in_features, periodic_features, out_features, dropout=0.1):
        super(SineActivation, self).__init__()

        # Trọng số biên độ tuyến tính
        linear_features = out_features - in_features - periodic_features
        self.w0 = nn.Parameter(torch.randn(in_features, linear_features))
        self.b0 = nn.Parameter(torch.randn(1, linear_features))

        # Trọng số biên độ tuần hoàn
        self.w = nn.Parameter(torch.randn(in_features, periodic_features))
        self.b = nn.Parameter(torch.randn(1, periodic_features))

        self.dropout = nn.Dropout(dropout)

    def forward(self, x):
        """
        Args:
            x: Tensor đầu vào có dạng [seq_len, batch_size, in_features]
        Returns:
            Tensor đã được tăng cường có dạng [seq_len, batch_size, out_features]
        """
        # Đặc trưng tuyến tính
        v_linear = torch.matmul(x, self.w0) + self.b0

        # Đặc trưng tuần hoàn sử dụng hàm kích hoạt sine
        v_periodic = torch.sin(torch.matmul(x, self.w) + self.b)

        # Nối tất cả các đặc trưng: [tuyến tính, tuần hoàn, gốc]
        output = torch.cat([v_linear, v_periodic, x], dim=-1)

        return self.dropout(output)
```

## Xây dựng mô hình Transformer

### 1. Bổ sung đặc trưng thời gian với Time2Vector (SineActivation):

Đầu ra gồm:

- Đặc trưng gốc
- Đặc trưng thời gian tuyến tính
- Đặc trưng thời gian tuần hoàn (giá biến động theo mùa, theo chu kỳ giao dịch,...)

→ Giúp Transformer hiểu rõ hơn về dữ liệu chuỗi thời gian

# TRANSFORMER

```
class BTC_Transformer(nn.Module):
    def __init__(self,
                 num_encoder_layers: int,
                 num_decoder_layers: int,
                 in_features: int,
                 periodic_features: int,
                 out_features: int,
                 nhead: int,
                 dim_feedforward: int = 512,
                 dropout: float = 0.1,
                 activation: str = 'relu'):
        super(BTC_Transformer, self).__init__()

        self.sine_activation = SineActivation(in_features=in_features,
                                              periodic_features=periodic_features,
                                              out_features=out_features,
                                              dropout=dropout)

        self.transformer = nn.Transformer(d_model=out_features,
                                         nhead=nhead,
                                         num_encoder_layers=num_encoder_layers,
                                         num_decoder_layers=num_decoder_layers,
                                         dim_feedforward=dim_feedforward,
                                         dropout=dropout,
                                         activation=activation)

        self.generator = nn.Linear(out_features, in_features)

    def encode(self, src: Tensor, src_mask: Tensor):
        return self.transformer.encoder(self.sine_activation(src), src_mask)

    def decode(self, tgt: Tensor, memory: Tensor, tgt_mask: Tensor):
        return self.transformer.decoder(self.sine_activation(tgt), memory, tgt_mask)

    def forward(self,
               src: Tensor,
               trg: Tensor,
               src_mask: Tensor=None,
               tgt_mask: Tensor=None,
               mem_mask: Tensor=None,
               src_padding_mask: Tensor=None,
               tgt_padding_mask: Tensor=None,
               memory_key_padding_mask: Tensor=None):

        src_emb = self.sine_activation(src)
        tgt_emb = self.sine_activation(trg)
        outs = self.transformer(src_emb, tgt_emb, src_mask, tgt_mask, mem_mask,
                               src_padding_mask, tgt_padding_mask, memory_key_padding_mask)
        return self.generator(outs)
```

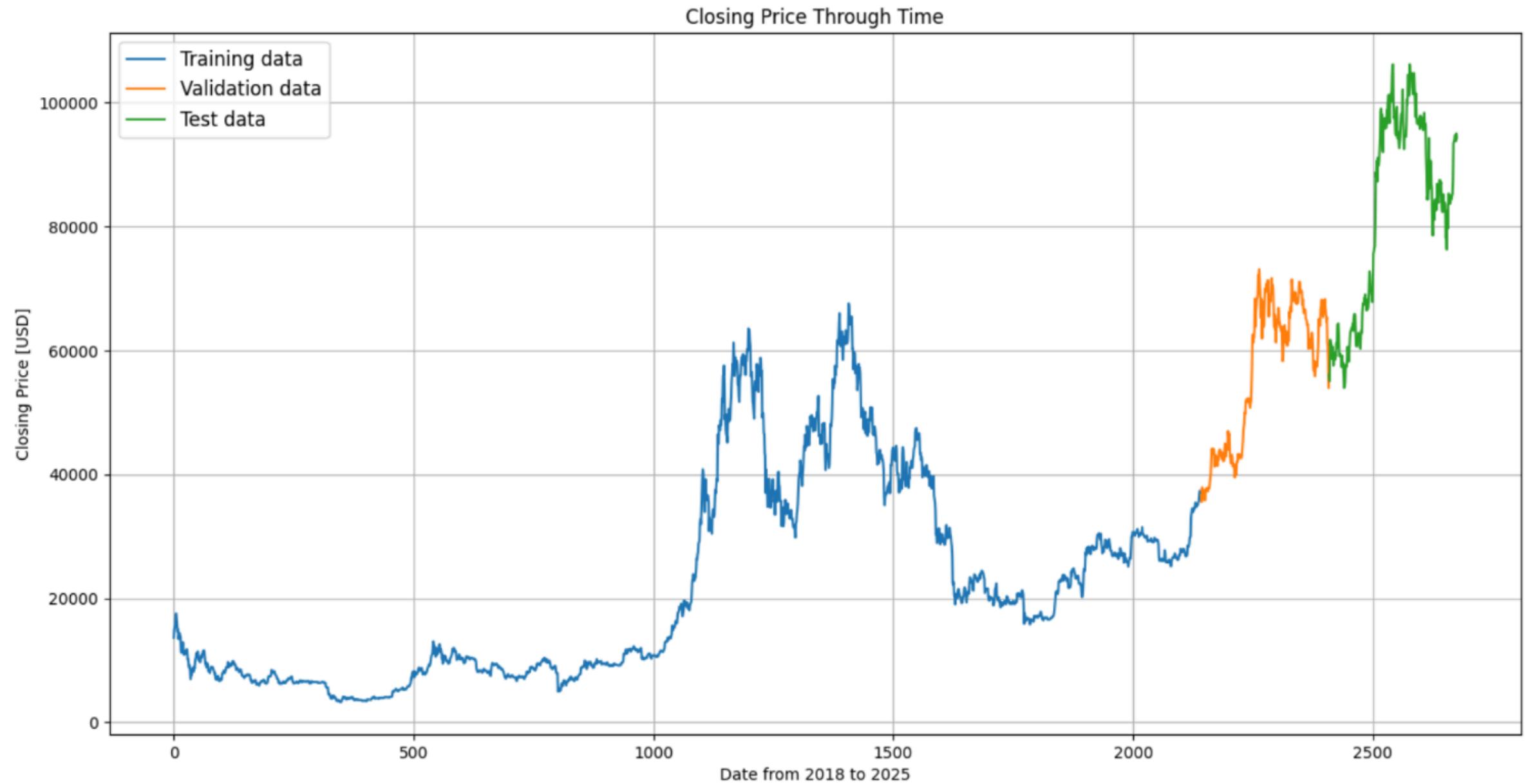
## Xây dựng mô hình Transformer

1. Định nghĩa mô hình Transformer sử dụng module **nn.Transformer** tích hợp sẵn của PyTorch

### 2. Tham số mô hình:

```
# Tham số mô hình
in_features = modeling_data.shape[1]
periodic_features = 8
out_features = 64
nhead = 8
num_encoder_layers = 4
num_decoder_layers = 4
dim_feedforward = 256
dropout = 0.1
activation = 'relu'
```

# TRANSFORMER



## Chuẩn bị dữ liệu huấn luyện mô hình

Tỉ lệ chia tập dữ liệu là:

- Tập Train: 80%
- Tập Test: 20%

# TRANSFORMER

## Sử dụng Optuna để tìm ra các siêu tham số tối ưu cho mô hình

Study statistics:

Number of finished trials: 50

Number of pruned trials: 23

Number of complete trials: 27

Best trial:

Value: 0.007848398759961128

Params:

bptt\_src: 10

bptt\_tgt: 14

scaler\_name: minmax

clip\_param: 0.25

random\_start\_point: False

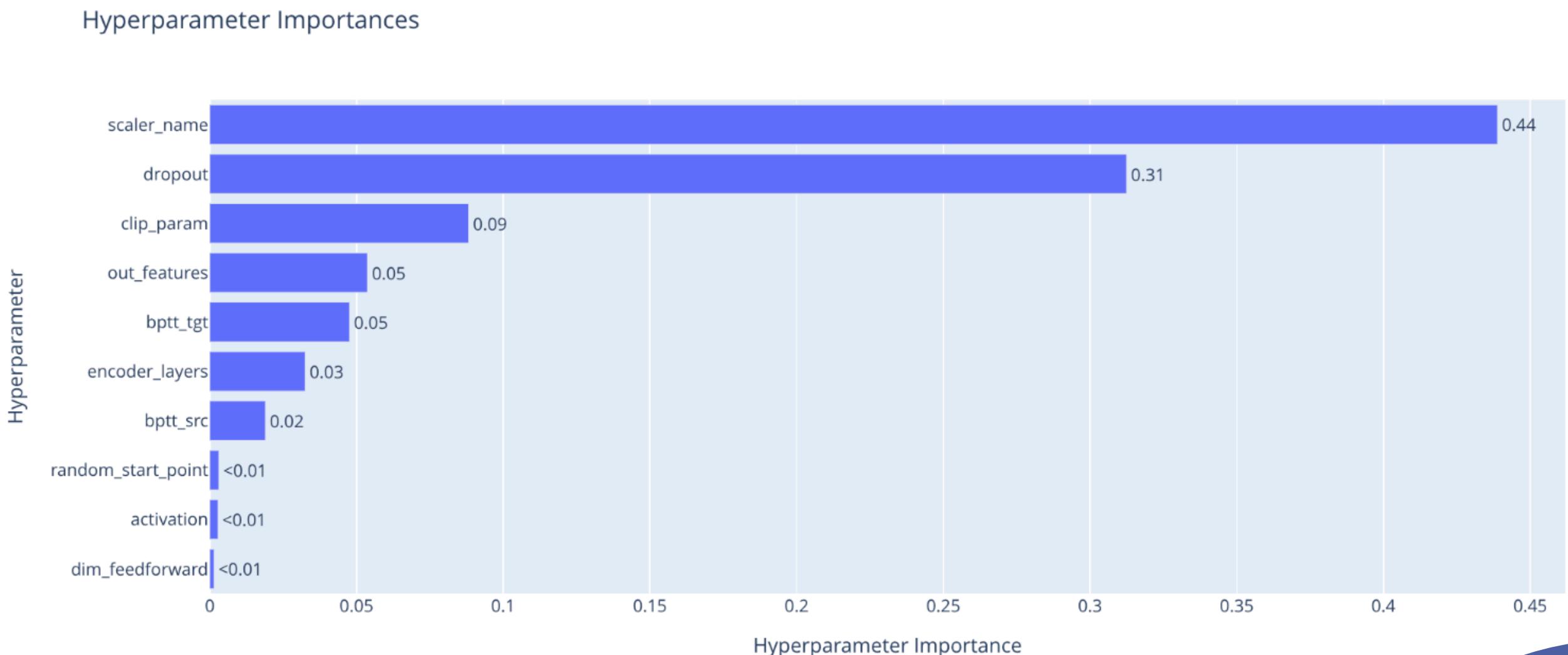
encoder\_layers: 4

out\_features: 40

dim\_feedforward: 512

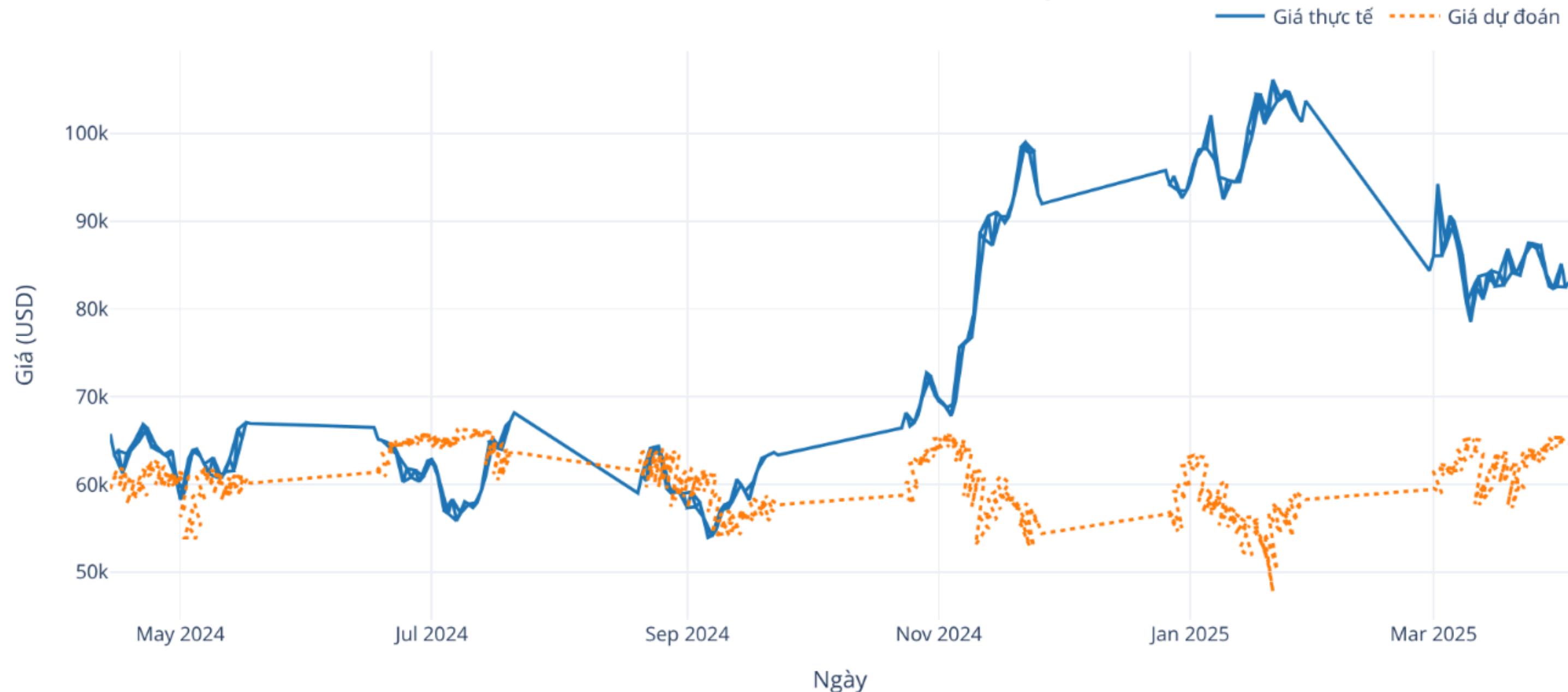
dropout: 0.3

activation: gelu



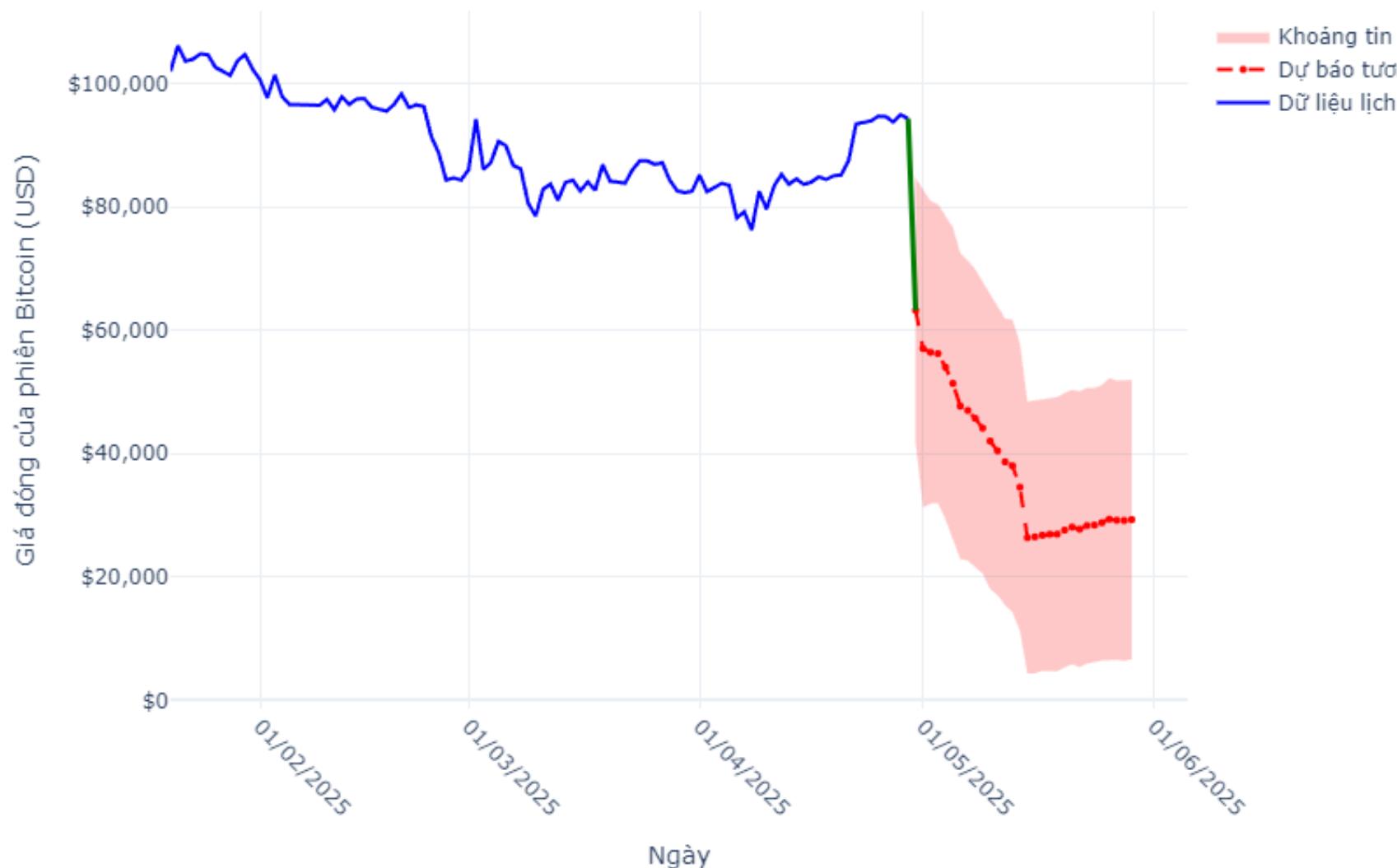
# TRANSFORMER

Giá Bitcoin Thực tế vs Giá Dự đoán (Tập Test)



# TRANSFORMER

Dự báo giá Bitcoin: Lịch sử vs Tương lai (30 ngày)



## DỰ ĐOÁN GIÁ BITCOIN 30 NGÀY TƯƠNG LAI

Dự đoán bắt đầu sau: 29/04/2025

Ngày	Ngày	Giá Dự Đoán	Khoảng Tin Cậy ( $\pm 1\sigma$ )	Phạm Vi
1	30/04/2025	\$ 63,246.05	$\pm \$21,444.04$	\$ 41,802.01 - \$ 84,690.08
2	01/05/2025	\$ 57,037.49	$\pm \$25,765.80$	\$ 31,271.70 - \$ 82,803.29
3	02/05/2025	\$ 56,416.04	$\pm \$24,530.25$	\$ 31,885.79 - \$ 80,946.29
4	03/05/2025	\$ 56,221.87	$\pm \$24,189.55$	\$ 32,032.33 - \$ 80,411.42
5	04/05/2025	\$ 54,014.45	$\pm \$24,658.50$	\$ 29,355.95 - \$ 78,672.95
6	05/05/2025	\$ 51,391.81	$\pm \$25,284.44$	\$ 26,107.37 - \$ 76,676.25
7	06/05/2025	\$ 47,700.46	$\pm \$24,809.14$	\$ 22,891.32 - \$ 72,509.60
8	07/05/2025	\$ 47,001.22	$\pm \$24,341.35$	\$ 22,659.87 - \$ 71,342.58
9	08/05/2025	\$ 45,709.57	$\pm \$24,146.60$	\$ 21,562.97 - \$ 69,856.17
10	09/05/2025	\$ 44,143.04	$\pm \$23,617.93$	\$ 20,525.11 - \$ 67,760.98
11	10/05/2025	\$ 42,040.99	$\pm \$23,994.56$	\$ 18,046.42 - \$ 66,035.55
12	11/05/2025	\$ 40,453.73	$\pm \$23,406.23$	\$ 17,047.51 - \$ 63,859.96
13	12/05/2025	\$ 38,670.33	$\pm \$23,234.13$	\$ 15,436.21 - \$ 61,904.46
14	13/05/2025	\$ 38,016.99	$\pm \$23,743.13$	\$ 14,273.86 - \$ 61,760.11
15	14/05/2025	\$ 34,555.08	$\pm \$23,261.78$	\$ 11,293.30 - \$ 57,816.85
16	15/05/2025	\$ 26,381.76	$\pm \$22,039.83$	\$ 4,341.93 - \$ 48,421.59
17	16/05/2025	\$ 26,503.79	$\pm \$22,129.75$	\$ 4,374.05 - \$ 48,633.54

# SO SÁNH KẾT QUẢ

Kết quả đánh giá các mô hình trên tập test

Model/Metrics	RMSE	MAE	MAPE	R^2
ARIMA	37,113.26	31,951.77	41.68%	-2.845
ARIMAX	37,194.02	32,019.16	41.76%	-2.86
XGBoost	21,477.11	15,641.12	17.65%	-0.97
LSTM	2,119.65	1,558.96	2.25%	0.9867
GRU	2,808.44	2,085.24	2.85%	0.978
Transformer	15,798.00	11,017.00	13.16%	0.25

# KẾT LUẬN & HƯỚNG PHÁT TRIỂN

## Kết luận

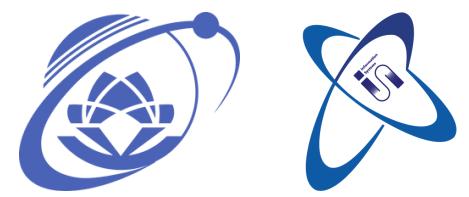
- LSTM đạt hiệu suất dự báo tốt nhất với  $R^2 = 0.9867$ , RMSE = 2,119.65
- GRU cũng cho kết quả mạnh nhưng kém LSTM một chút.
- ARIMA và ARIMAX kém hiệu quả, với  $R^2$  âm và sai số cao.
- XGBoost và Transformer cho kết quả trung bình, cần tối ưu thêm.
- Deep Learning, đặc biệt là RNNs (LSTM, GRU), vượt trội trong dự báo giá Bitcoin ngắn hạn.

## Hướng phát triển

- Áp dụng thêm các kỹ thuật attention nâng cao như Temporal Fusion Transformers.
- Kết hợp thêm dữ liệu đa chiều: chỉ báo kỹ thuật, tin tức, dữ liệu mạng xã hội.
- Tối ưu mô hình bằng AutoML và kỹ thuật tăng cường dữ liệu.

# TÀI LIỆU THAM KHẢO

1. Henrique et al. (2015), **Stock price prediction using ARIMA and ARIMAX**, ESWA
2. H.M et al. (2018), **Predicting Bitcoin price using ML**, Procedia CS
3. McNally et al. (2018), **NSE Stock Market Prediction Using Deep-Learning**
4. Nelson & Pereira (2020), **GRU vs LSTM in cryptocurrency forecasting**
5. Lim et al. (2021), **Deep Learning for Time-Series Forecasting**, Phil. Trans. R. Soc. A
6. Chen & Guestrin (2016), **XGBoost: A Scalable Tree Boosting System**, KDD
7. Hochreiter & Schmidhuber (1997), **Long Short-Term Memory**, Neural Computation
8. Cho et al. (2014), **RNN Encoder-Decoder for SMT**, arXiv
9. Vaswani et al. (2017), **Attention Is All You Need**, arXiv



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
KHOA HỆ THỐNG THÔNG TIN

**THANK YOU!**