



DỰ ĐOÁN GIÁ CỔ PHIẾU BA NGÂN HÀNG VIỆT NAM - BIDV, VIETCOMBANK, EXIMBANK

TRẦN HOÀNG PHÚC¹, NGUYỄN VIỆT HOÀNG², LÊ BÁ NHẤT LONG³, NGUYỄN HÙNG TUẤN⁴,
và LÊ ANH DUY⁵

¹Khoa Hệ thống thông tin, Trường Đại học Công nghệ Thông tin - UIT, (e-mail: 21522479@gm.uit.edu.vn)

²Khoa Hệ thống thông tin, Trường Đại học Công nghệ Thông tin - UIT, (e-mail: 21522095@gm.uit.edu.vn)

³Khoa Hệ thống thông tin, Trường Đại học Công nghệ Thông tin - UIT, (e-mail: 21522300@gm.uit.edu.vn)

⁴Khoa Hệ thống thông tin, Trường Đại học Công nghệ Thông tin - UIT, (e-mail: 21521633@gm.uit.edu.vn)

⁵Khoa Hệ thống thông tin, Trường Đại học Công nghệ Thông tin - UIT, (e-mail: 21521994@gm.uit.edu.vn)

TÓM TẮT Trước đây, những nhà nghiên cứu thị trường cho rằng việc dự đoán giá cổ phiếu là việc không thể, vì nó phụ thuộc nhiều vào sự biến động của thị trường. Nhưng ngày nay lại có những đề xuất chứng minh rằng việc sử dụng các mô hình thống kê, thuật toán học máy hay học sâu nếu được thiết kế thích hợp thì việc dự đoán giá cổ phiếu có thể được thực hiện rất chính xác. Trong nghiên cứu này, chúng em dựa trên các mô hình đã có sẵn để thực hiện dự đoán giá cổ phiếu của 3 ngân hàng lớn tại Việt Nam: Ngân hàng Thương mại cổ phần Ngoại thương Việt Nam (VCB), Ngân hàng Thương mại cổ phần Đầu tư và Phát triển Việt Nam (BIDV), Ngân hàng thương mại cổ phần Xuất Nhập khẩu Việt Nam (Eximbank). Để tìm ra các mô hình nào tốt nhất, chúng em lần lượt thực hiện các mô hình sau trên bộ dữ liệu của cả 3 ngân hàng để so sánh: Hồi quy tuyến tính (LR), Đường trung bình động tích hợp tự hồi quy (ARIMA), Mạng thần kinh tái phát (RNN), Đơn vị định kì có kiểm soát (GRU), Bộ nhớ ngắn hạn dài hạn (LSTM), thuật toán siêu học (Meta-Learning), Phân tích mở rộng cơ sở thần kinh (NBeats), Bộ lọc Kalman (Kalman Filter), trung bình động tự hồi quy vector (VARMA), Nội suy phân cấp thần kinh để dự báo chuỗi thời gian (N-HITS). Ngoài ra, cần đo hiệu quả của các mô hình bằng cách sử dụng các độ đo sau: Lỗi bình phương trung bình gốc (RMSE), Lỗi phần trăm tuyệt đối trung bình (MAPE), Lỗi tuyệt đối trung bình (MAE).

INDEX TERMS Stock price prediction, machine learning, deep learning.

I. GIỚI THIỆU

Hiện nay, nhóm cổ phiếu ngành ngân hàng có xu hướng tích cực trở lại, và nó được xem là nhóm cổ phiếu “vua” có thể đóng góp vai trò dẫn dắt đà tăng của thị trường chứng khoán. Giá cổ phiếu là những chỉ số quan trọng giúp các nhà đầu tư đưa ra quyết định đúng đắn.

Dựa trên xu thế đó, chúng em tập trung nghiên cứu để tìm ra mô hình dự đoán tốt nhất cho giá cổ phiếu của ngân hàng bằng cách sử dụng kết hợp giữa các kiến thức của học máy và học sâu. Các mô hình được sử dụng trong nghiên cứu này: Hồi quy tuyến tính (LR), Đường trung bình động tích hợp tự hồi quy (ARIMA), Mạng thần kinh tái phát (RNN), Đơn vị định kì có kiểm soát (GRU), Bộ nhớ ngắn hạn dài hạn (LSTM), thuật toán siêu học (Meta-Learning), Phân tích mở rộng cơ sở thần kinh (NBeats), Bộ lọc Kalman (Kalman Filter), trung bình động tự hồi quy vector (VARMA), Nội suy phân cấp thần kinh để dự báo chuỗi thời gian (N-HITS).

Hy vọng rằng, việc áp dụng các phương pháp này sẽ mở ra một phương pháp đánh giá mới về sự biến động giá của

thị trường chứng khoán, giúp các nhà đầu tư có cái nhìn sáng suốt và đưa ra các quyết định đúng đắn.

II. CÁC NGHIÊN CỨU LIÊN QUAN

Linear Regression: Trong nghiên cứu này, Arif Mudi Priyatno và những tác giả khác [1] (2023) đã so sánh hai phương pháp học máy Random Forest Regression và Linear Regression để dự báo cổ phiếu BBCA. Kết quả đánh giá các độ đo MSE, RMSE, MAE, MAPE của Linear Regression thấp hơn nhiều so với Random Forest Regression. Cho thấy Linear Regression hoạt động tốt hơn trong hiệu suất dự báo.

GRU: Mochamad Ridwan, Kusman Sadik, và Farit Mochamad Afendi [2] (2024) đã so sánh hiệu suất của hai mô hình ARIMA và GRU trong dự báo giá cổ phiếu ở tần suất cao từ ngân hàng HIMBARA. Sử dụng MAPE làm độ đo, họ thấy rằng mô hình GRU vượt trội hơn mô hình ARIMA.

LSTM: Shahzad Zaheer và những tác giả khác [3] (2023) sử dụng LSTM và những mô hình học sâu như RNN,

CNN,... lấy dữ liệu chứng khoán đầu vào, dự đoán hai thông số cổ phiếu là giá đóng và giá cao cho ngày hôm sau.

ARIMA: Prapanna Mondal, Labani Shit và Saptarsi Goswami [4] (2014) đã tiến hành một nghiên cứu trên 56 cổ phiếu từ các lĩnh vực khác nhau, độ chính xác của mô hình ARIMA trong dự đoán giá cổ phiếu cao hơn 85%, cho thấy ARIMA mang lại độ chính xác cao về sự dự đoán.

Meta-Learning: Cheng-Wen Hsu cùng các tác giả khác [5] đã nghiên cứu và đề xuất một khung học meta (meta-learning) dự đoán giá cổ phiếu trong ngắn hạn sử dụng mạng nơ-ron tích chập (CNN) bao gồm mạng nơ-ron tích chập thời gian, mạng nơ-ron tích chập hoàn toàn và mạng nơ-ron dư cho thấy sự cải thiện đáng kể về độ chính xác dự đoán và lợi nhuận.

RNN: Hansika Hewamalage, Christoph Bergmeir và Kasun Bandara đã tiến hành một nghiên cứu về việc sử dụng Mạng Nơ-ron Tái Phát (RNN) [6] trong dự báo chuỗi thời gian. Họ nhấn mạnh rằng RNN đã trở thành phương pháp dự báo cạnh tranh, nhưng vẫn cần cải thiện để đạt được tính tự động và hiệu quả.

NBEATS: Boris N. Oreshkin, Nicolas Chapados, Dmitri Carpov và Yoshua Bengio đã đưa ra NBEATS [7]. Công trình này đề xuất một phương pháp mới, kết hợp các yếu tố của deep learning và các khối dữ lượng để tạo ra một mô hình dự báo chuỗi thời gian có khả năng giải thích, áp dụng rộng rãi và nhanh chóng.

N-HITS: Cristian Challu và các tác giả khác đã tiến hành nghiên cứu và giới thiệu N-HITS [8], một mô hình giải quyết cả hai thách thức này bằng cách kết hợp các kỹ thuật phân cấp (hierarchical interpolation) và lấy mẫu dữ liệu đa tốc độ (multi-rate data sampling). Họ tiến hành các thí nghiệm trên bộ dữ liệu quy mô lớn, N-HITS cung cấp độ chính xác trung bình được cải thiện gần 20% so với kiến trúc Transformer mới nhất trong khi giảm thời gian tính toán đáng kể (50 lần).

III. TÀI NGUYÊN

A. NGUỒN DỮ LIỆU

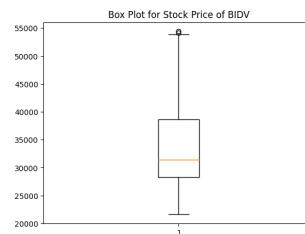
Tập dữ liệu được sử dụng trong nghiên cứu này được lấy từ <https://finance.yahoo.com/>, một trang tin tức thị trường chứng khoán thế giới đáng tin cậy. Tập dữ liệu bao gồm dữ liệu giá cổ phiếu của ba ngân hàng tại Việt Nam: BIDV, Eximbank và Vietcombank. Tập dữ liệu được lấy trong khoảng thời gian 01-03-2019 đến 01-06-2024. Mỗi mục trong tập dữ liệu bao gồm các chỉ số tài chính chính, bao gồm:

- **Date:** Ngày giao dịch.
- **Open:** Giá cổ phiếu mở cửa vào đầu ngày giao dịch.
- **High:** Giá cổ phiếu cao nhất ghi nhận được trong ngày.
- **Low:** Giá cổ phiếu thấp nhất ghi nhận được trong ngày.
- **Close:** Giá cổ phiếu đóng cửa vào một ngày nhất định.
- **Adj Close:** Giá đóng cửa được điều chỉnh.
- **Volume:** Khối lượng/tổng số cổ phiếu được giao dịch.

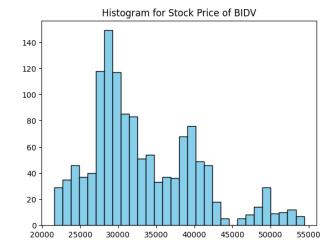
B. THỐNG KÊ MÔ TẢ

Bảng 1. Thống kê mô tả

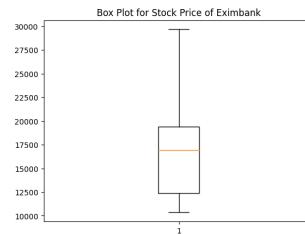
	BIDV	Eximbank	Vietcombank
Count	1306	1306	1306
Mean	33335.27142	16673.73704	66674.47299
Std	7058.435047	4257.231701	13618.83797
Min	21590.11133	10346.04492	37957.3125
Q1	28292.92188	12394.06738	56604.17578
Q2	31397.38281	16949.15234	65164.47656
Q3	38601.47266	19385.59375	75508.04297
Max	54400	29661.01758	97400
Mode	28222.36719	12146.89258	55077.91797
Median	31397.38281	16949.15234	65164.47656
Var	49821505.31	18124021.75	185472747.7
Kurtosis	0.23806089	-0.807813649	-0.610997452
Skewness	0.840622466	0.442884414	0.336313457
CV	0.21174074	0.255325587	0.204258652



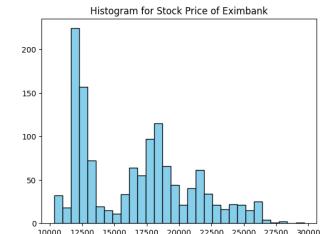
Hình 1. Biểu đồ boxplot của giá cổ phiếu BIDV



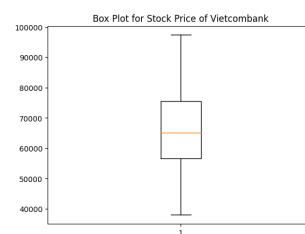
Hình 2. Biểu đồ histogram của giá cổ phiếu BIDV



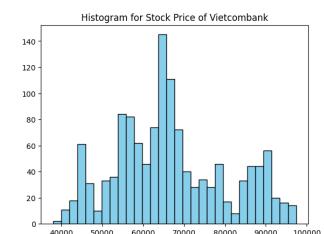
Hình 3. Biểu đồ boxplot của giá cổ phiếu Eximbank



Hình 4. Biểu đồ histogram của giá cổ phiếu Eximbank



Hình 5. Biểu đồ boxplot của giá cổ phiếu Vietcombank



Hình 6. Biểu đồ histogram của giá cổ phiếu Vietcombank

C. CÔNG CỤ

Trong nghiên cứu này, chúng em đã sử dụng các công cụ phân tích thống kê khác nhau trong Python bao gồm numpy, pandas, sklearn và matplotlib.pyplot.



D. TỶ LỆ PHÂN CHIA TẬP DỮ LIỆU

Phân chia dữ liệu thành hai tập: tập train và tập test theo các tỷ lệ 7:3, 8:2, 9:1. Tỷ lệ tiêu chuẩn là 7:3, tức là sử dụng 70% dữ liệu cho việc train và 30% cho việc test. Tỷ lệ này thường được sử dụng bởi vì với một bộ dữ liệu lớn và cần đạt được độ chính xác cao trong đánh giá mô hình. Bên cạnh đó, tỷ lệ 8:2 được coi là tỷ lệ tốt nhất cho việc phân chia tập train và tập test. Tỷ lệ này thường được sử dụng khi bộ dữ liệu có kích thước lớn và sẽ giúp mô hình được train với nhiều thông tin hơn từ dữ liệu. Cuối cùng, tỷ lệ 9:1 có train gần với test sẽ giúp mô hình học được nhiều thông tin hơn từ tập huấn luyện và đánh giá hiệu quả tốt trên tập kiểm tra.

E. CÁC CHỈ SỐ ĐÁNH GIÁ MÔ HÌNH

Để đánh giá độ chính xác của các mô hình, sử dụng ba tham số là Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE) và Root Mean Squared Error (RMSE). Các giá trị này càng nhỏ thì mô hình càng tốt.

RMSE được tính bằng cách lấy căn bậc hai của trung bình bình phương sai số giữa các giá trị dự đoán và các giá trị thực tế.

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

MAPE còn được gọi là độ lệch phần trăm tuyệt đối trung bình. Nó thường biểu thị độ chính xác dưới dạng tỷ lệ được xác định bởi công thức:

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

MAE được đo bằng cách tính trung bình giá trị tuyệt đối của sai số giữa giá trị thực tế và giá trị dự đoán.

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

IV. PHƯƠNG PHÁP NGHIÊN CỨU

A. LINEAR REGRESSION

Hồi quy tuyến tính là một kỹ thuật thống kê được sử dụng để mô hình hóa mối liên hệ giữa một biến phụ thuộc và một hoặc nhiều biến độc lập:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

Trong đó:

- y là giá trị dự đoán của biến phụ thuộc (y).
- x là các biến độc lập.
- β_0 là giá trị dự đoán của y khi X bằng 0 (intercept).
- β_1 là hệ số hồi quy.
- ε là sai số.

Công thức tính β_0 và β_1 :

$$\beta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}, \beta_0 = \bar{y} - \beta_1 \bar{x}$$

Trong đó:

- x_i, y_i là các giá trị cụ thể của biến độc lập, phụ thuộc.
- \bar{x}, \bar{y} là giá trị trung bình của x, y tương ứng.

B. ARIMA

Autoregressive Integrated Moving Average (ARIMA) là thuật toán thống kê phổ biến dùng để dự báo dữ liệu chuỗi thời gian, được kết hợp giữa Tự hồi quy (Autoregressive - AR), Trung bình trượt (Moving Average - MA) và tích hợp sai phân (Integrated - I).

Mô hình AR có phương trình như sau:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \varepsilon_t$$

Trong đó:

- y_t là quan sát tại thời điểm t
- c là hệ số chặn
- ϕ_1, \dots, ϕ_p là các tham số cần ước tính
- y_{t-1}, \dots, y_{t-p} là các giá trị dự báo tại p bước trước đó
- ε_t là nhiễu trắng Sai phân (Integrated - I).

$$y_t = Y_t - Y_{t-d}$$

Trong đó:

- y_t là sai phân tại thời điểm t
- Y_t, \dots, Y_{t-d} là các quan sát tại hiện tại và trước đó

Trung bình trượt (MA) là mô hình thay vì sử dụng các quan sát trong quá khứ của các biến dự báo hồi quy, trung bình trượt sử dụng các lỗi dữ báo trong quá khứ.

$$y_t = c + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \dots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Trong đó:

- y_t là trung bình trượt tại thời điểm t
- c là hệ số chặn
- $\theta_1, \dots, \theta_p$ là các tham số cần ước tính
- $\varepsilon_{t-1}, \dots, \varepsilon_{t-p}$ là các lỗi dữ báo tại q bước trước đó
- ε_t là nhiễu trắng

Kết hợp ba mô hình trên được phương trình ARIMA:

$$y_t = c + \phi_1 y_{t-1} + \dots + \phi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

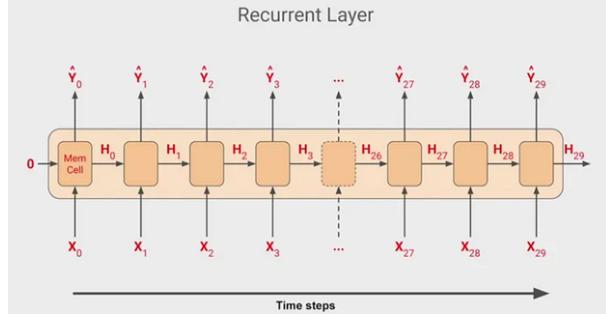
Trong đó:

- y_t là quan sát tại thời điểm t
- c là giá trị hằng
- ϕ_p là tham số của AR
- θ_q là tham số của MA
- ε_t là nhiễu trắng

Các giá trị tham số của mô hình ARIMA (p, d, q , AR(p) có thể được tính bằng Hàm tự tương quan (Autocorrelation Function – ACF), I(d) có thể sử dụng các loại kiểm định như Dickey Fuller (DF) hoặc Augmented Dickey Fuller (ADF) nếu như chuỗi thời gian chưa dừng và MA(q) có thể sử dụng Hàm tự tương quan một phần (Partial Autocorrelation Function PACF).



C. RNN



$$h_t = \sigma(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t = \sigma(W_y h_t + b_y)$$

- h_t : Vector lấp ẩn tại thời điểm t
- X_t : vector đầu vào tại thời điểm t
- \hat{Y}_t : vector đầu ra tại thời điểm t
- W_h, U_h, W_y : Là ma trận trọng số ngẫu nhiên
- b_h, b_y : là các bias
- σ_h, σ_y : là các hàm kích hoạt

Một số hàm kích hoạt phổ biến là: Sigmoid, Tanh, RELU.
Đầu vào: chuỗi thời gian $x_1, x_2, x_3, \dots, x_t$, nhãn thực tế tương ứng $y_1, y_2, y_3, \dots, y_t$.

Đầu ra: Tính toán hàm mất mát (Loss Function) và cập nhật các trọng số W_h, U_h, W_y, b_h, b_y

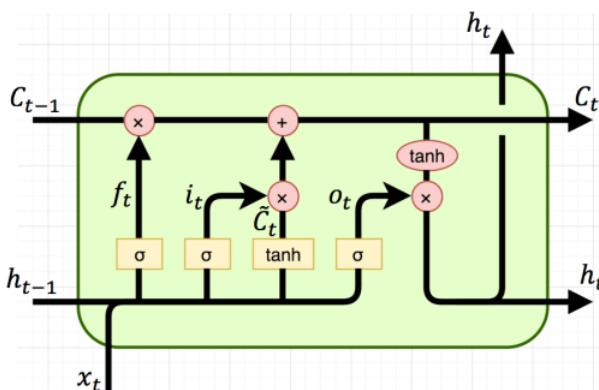
$$\text{Loss Function} = \frac{1}{N} \sum_{i=1}^N (-y_i + \hat{y}_i)^2$$

Lặp lại quá trình trên cho đến khi hàm mất mát giảm đủ (không thay đổi trong một số bước lặp nhất định, không thể giảm được nữa)

D. LSTM

LSTM là một mạng nơ-ron tuần tự sâu trong học sâu, cho phép thông tin tồn tại lâu dài. Đây là một loại đặc biệt của Mạng Nơ-ron Tái Phát, có khả năng xử lý vấn đề vanishing gradient gấp phai bởi RNN.

- Input: c_{t-1}, h_{t-1} . Trong đó, x_t là đầu vào tại trạng thái thứ t của mô hình. c_{t-1}, h_{t-1} là đầu ra từ lớp trước. h chơi vai trò tương tự như s trong RNN, trong khi c là điểm mới của LSTM.
- Output: c, h_t, c_t . Ở đây, c biểu diễn trạng thái của ô (cell state) và h biểu diễn trạng thái ẩn (hidden state).



Kí hiệu σ , tanh là các hàm kích hoạt.

f_t, i_t, o_t : forget gate, input gate và output gate.

• Forget gate: $f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b_f)$

• Input gate: $i_t = \sigma(U_i * x_t + W_i * h_{t-1} + b_i)$

• Output gate: $o_t = \sigma(U_o * x_t + W_o * h_{t-1} + b_o)$

Nhận xét: $0 < f_t, i_t, o_t < 1$; b_f, b_i, b_o là các hệ số bias.

$c_t = \tanh(U_c * x_t + W_c * h_{t-1} + b_c)$.

$c_t = f_t * c_{t-1} + i_t * c_t$, forget gate quyết định cần lấy bao nhiêu từ cell state trước và input gate sẽ quyết định lấy bao nhiêu từ input của state và hidden layer của layer trước.

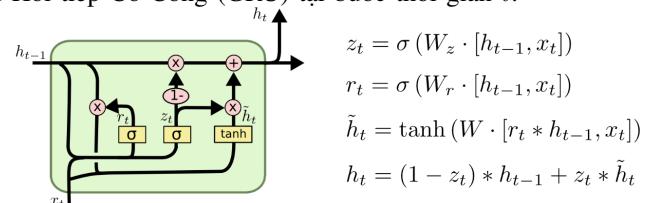
$h_t = o_t * \tanh(c_t)$, output gate quyết định cần lấy bao nhiêu từ cell state để trở thành output của hidden state. Ngoài ra h_t cũng được dùng để tính output y_t cho state t.

E. GRU

Ý tưởng cơ bản của GRU là sử dụng cơ chế cổng để lựa chọn cập nhật trạng thái ẩn của mạng tại mỗi bước thời gian. Các cơ chế cổng được sử dụng để kiểm soát luồng thông tin vào và ra khỏi mạng. GRU có hai cơ chế cổng, gọi là cổng thiết lập lại (reset gate) và cổng cập nhật (update gate).

Cổng thiết lập lại xác định mức độ của trạng thái ẩn trước đó nên được quên, trong khi cổng cập nhật xác định mức độ của đầu vào mới nên được sử dụng để cập nhật trạng thái ẩn.

Phương trình được sử dụng để tính toán cổng thiết lập lại (r_t), cổng cập nhật (z_t), và trạng thái ẩn (h_t) của một Đơn vị Hồi tiếp Có Cổng (GRU) tại bước thời gian t:



Trong đó:

• x_t đại diện cho đầu vào tại bước thời gian t

• h_{t-1} là trạng thái ẩn trước đó

• W_r, W_z và W là các ma trận trọng số mà mạng học được trong quá trình huấn luyện

• b_r, b_z và b là các vector độ lệch

• σ biểu thị cho hàm sigmoid

F. VARMA

Vector Autoregressive Moving Average (VARMA) là sự kết hợp giữa hai mô hình: Vector tự hồi quy (VAR) là mô hình dùng để kiểm tra các mối quan hệ giữa các biến tương tác với nhau. Trung bình trượt (MA) là mô hình thay vì sử dụng các quan sát trong quá khứ của các biến dự báo hồi quy, trung bình trượt sử dụng các lỗi dữ báo trong quá khứ.

Mô hình VARMA được định nghĩa như sau:

$$y_t = c + \phi_1 y_{t-1} + \dots + \phi_p y_{t-p} + C D_t + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_{t-p} + \varepsilon_t$$

Trong đó:

• y_t là vector của các biến chuỗi thời gian tại thời điểm t



- c là vector độ lệch không đổi trong mỗi phương trình
- ϕ_1, \dots, ϕ_p : ma trận tham số của AR ($i = 1, \dots, p$)
- C : ma trận hệ số của các biến hồi quy
- D_t là vector chứa các biến hồi quy xác định thích hợp như biến không đổi, xu hướng hoặc mùa vụ
- $\theta_1, \dots, \theta_p$: ma trận tham số của MA ($j = 1, \dots, q$)
- ε_t là vector lỗi ngẫu nhiên tại thời điểm t

G. KALMAN FILTER

1) Định nghĩa

Kalman Filter là một thuật toán tổng quát được sử dụng để ước lượng các tham số hệ thống. Nó có thể sử dụng các đo lường không chính xác hoặc nhiễu để ước lượng trạng thái của biến đó hoặc một biến không quan sát được khác với độ chính xác cao hơn.

Sức mạnh thực sự của Kalman Filter không phải là làm mịn các đo lường. Mà đó là khả năng ước lượng các tham số hệ thống mà không thể được đo lường hoặc quan sát với độ chính xác

2) Công thức

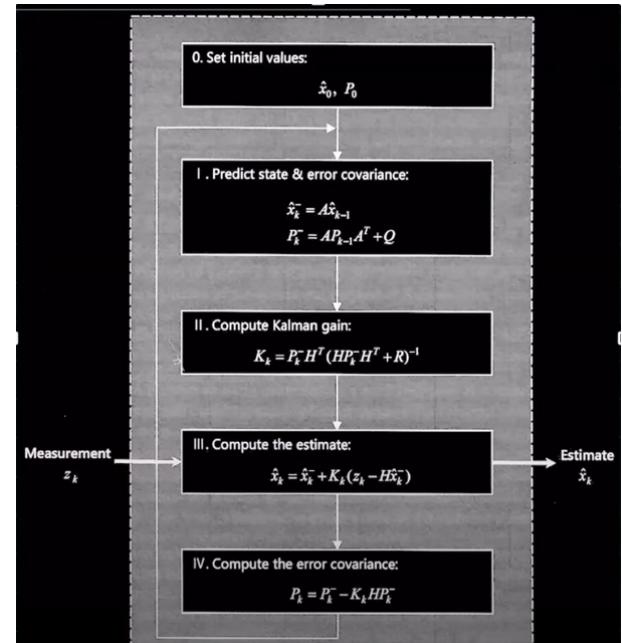
Description	Equation
Kalman Gain	$K_k = P'_k H^T (H P'_k H^T + R)^{-1}$
Update Estimate	$\hat{x}_k = \hat{x}'_k + K_k (z_k - H \hat{x}'_k)$
Update Covariance	$P_k = (I - K_k H) P'_k$
Project into $k+1$	$\hat{x}'_{k+1} = \Phi \hat{x}_k$ $P_{k+1} = \Phi P_k \Phi^T + Q$

Trong đó:

- K là Kalman Gain
- P là Ma trận hiệp phương sai
- H là ma trận quan sát
- H^T là ma trận chuyển vị của H
- Z_k là giá trị đo được (tại thời điểm k)
- X_k là giá trị dự đoán (tại thời điểm k)

3) Cách thuật toán hoạt động

- Bước 1: Khởi tạo giá trị ban đầu của X và P bằng với giá trị đầu tiên của bộ dữ liệu
- Bước 2: Sau đó sẽ tiến hành dự đoán X_{k-} và P_{k-} (Giá trị trước khi đo đạc)
- Bước 3: Tính toán lại Kalman Gain
- Bước 4: Sau khi đo lường được giá trị Z_k thì sẽ tính toán lại X_k (Giá trị dự đoán)
- Bước 5: Sau đó sẽ tính toán lại P và lặp lại từ bước 2



H. META-LEARNING

Một trong những thuật toán nổi tiếng nhất trong meta-learning là Model-Agnostic Meta-Learning (MAML).

Model-Agnostic Meta-Learning (MAML)

Mục tiêu: MAML nhằm tìm một khái niệm tốt cho các tham số của mô hình, giúp mô hình có thể được tinh chỉnh nhanh chóng trên một nhiệm vụ mới chỉ với một vài bước cập nhật gradient.

Các bước của thuật toán MAML

- 1. Khởi tạo tham số mô hình: Bắt đầu với các tham số khởi tạo θ .
- 2. Lấy mẫu một loạt các nhiệm vụ: Lấy mẫu một loạt các nhiệm vụ T_i từ phân phối nhiệm vụ $p(T)$
- 3. Đổi với mỗi nhiệm vụ T_i

Lấy mẫu bộ dữ liệu hỗ trợ $D_{T_i}^{\text{train}}$ và bộ dữ liệu truy vấn $D_{T_i}^{\text{test}}$.

Tính gradient của hàm mất mát trên bộ dữ liệu hỗ trợ với tham số θ :

$$\nabla_{\theta} \mathcal{L}_{T_i}(f_{\theta})$$

Cập nhật các tham số sử dụng gradient:

$$\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{T_i}(f_{\theta})$$

- 4. Cập nhật Meta

Tính hàm mất mát trên bộ dữ liệu truy vấn sử dụng tham số đã cập nhật θ'_i

$$\mathcal{L}_{T_i}(f_{\theta'_i})$$

Tính gradient của hàm mất mát trên bộ dữ liệu truy vấn đối với các tham số khởi tạo θ

$$\nabla_{\theta} \mathcal{L}_{T_i}(f_{\theta'})$$

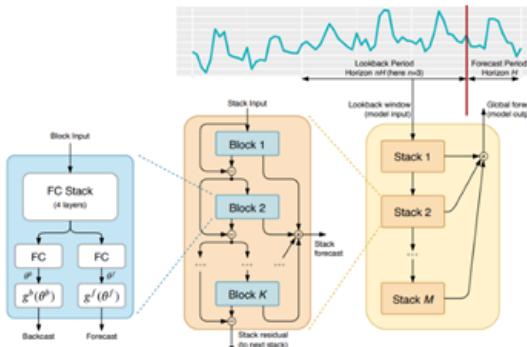


Cập nhật các tham số khởi tạo θ bằng cách trung bình các gradient qua loạt nhiệm vụ:

$$\theta \leftarrow \theta - \beta \sum_i \nabla_{\theta} \mathcal{L}_{T_i} (f_{\theta'_i})$$

với β là tốc độ học của meta.

I. N-BEATS



Cấu trúc mô hình gồm 3 phần:

Kiến trúc của Block Input: Đầu vào sẽ là đầu vào của mô hình (chuỗi thời gian). Mỗi block có 2 đầu ra: Một đầu ra (Backcast) sẽ được làm đầu vào cho block tiếp theo, đầu còn lại là Forecast được dùng để tổng hợp cho kết quả dự báo cuối cùng. Các lớp FC (Fully connected) sử dụng các hàm kích hoạt để học, nó ánh xạ tuyến tính với các hàm kích hoạt này.

Kiến trúc của Stack Input: Gồm các block, mỗi block sẽ cho ra một đầu ra forecast, các đầu ra forecast được tổng hợp lại để tạo đầu ra Stack forecast.

Tổng hợp tạo thành mô hình cuối cùng: Mô hình cuối cùng được tạo ra bao gồm nhiều stack, kết quả dự báo của mỗi stack được tổng hợp lại để cho ra kết quả dự báo cuối cùng.

Forecast: Mỗi khi đưa ra dự báo cho tương lai và các dự báo này được tổng hợp lại để tạo ra dự báo cuối cùng
Phương pháp tính:

$$FC(x_i) = RELU(Wx_i + \text{bias})$$

Trong đó, W và bias là các ma trận trọng số được khởi tạo ngẫu nhiên

Trong FC Stack sẽ có 4 lớp ẩn, được tính dựa vào công thức ở trên:

$$h_{\ell,1} = FC_{\ell,1}(x_{\ell}), \quad h_{\ell,2} = FC_{\ell,2}(h_{\ell,1}),$$

$$h_{\ell,3} = FC_{\ell,3}(h_{\ell,2}), \quad h_{\ell,4} = FC_{\ell,4}(h_{\ell,3}).$$

Sau đó nó chia đầu ra của lớp FC ở trên thành 2 nhánh riêng để tạo các vector trọng số θ^f và θ^b , các vector này dùng làm dự báo quá khứ (Backcast) và dự báo tương lai (Forecast). θ^f và θ^b được tính bằng cách ánh xạ tuyến tính

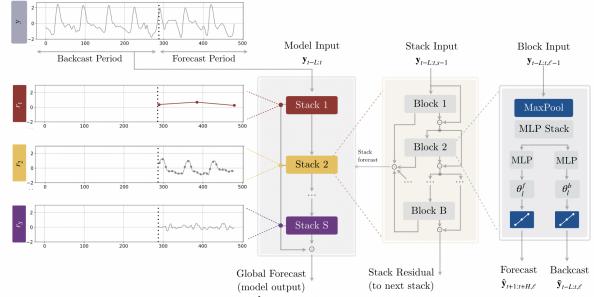
các layer ở trên:

$$\theta_{\ell}^b = LINEAR_{\ell}^b(h_{\ell,4}), \quad \theta_{\ell}^f = LINEAR_{\ell}^f(h_{\ell,4}).$$

Cuối cùng nó sẽ cho đầu ra dự báo backcast và forecast:

$$\hat{y}_{\ell} = \sum_{i=1}^{\dim(\theta_{\ell}^f)} \theta_{\ell,i}^f \vec{v}_i^f, \quad \hat{x}_{\ell} = \sum_{i=1}^{\dim(\theta_{\ell}^b)} \theta_{\ell,i}^b \vec{v}_i^b.$$

J. N-HITS



Nguyên lý hoạt động: Giống như N-BEATS, N-HITS thực hiện các phép chiếu phi tuyến cục bộ lên các hàm cơ sở qua nhiều khối, mỗi khối bao gồm một multilayer perceptron (MLP) học cách tạo ra các hệ số cho đầu ra backcast và forecast. Các khối được nhóm thành các stack, mỗi stack học một đặc điểm khác nhau của dữ liệu. Đầu vào của toàn bộ mạng là chuỗi tín hiệu $y_{t-L:t}$ từ L độ trễ.

Lấy mẫu tín hiệu đa tỷ lệ: Tại đầu vào của mỗi khối ℓ , một lớp MaxPool với kích thước kernel k_{ℓ} được sử dụng để phân tích các thành phần tín hiệu ở một tỷ lệ cụ thể. Kích thước k_{ℓ} lớn hơn cắt bớt các thành phần tần số cao, giúp khôi phục trung vào phân tích các nội dung quy mô lớn. Điều này giúp giảm số lượng tham số cần học, giảm thiểu overfitting và duy trì trường tiếp nhận ban đầu.

$$y_{t-L:t,\ell}^{(p)} = \text{MaxPool}(y_{t-L:t,\ell}, k_{\ell})$$

Hồi quy phi tuyến: Sau khi lấy mẫu, khối thực hiện hồi quy phi tuyến để tạo ra các hệ số MLP tiến và lùi. Các hệ số này được sử dụng để tổng hợp đầu ra backcast và forecast của khối. Quá trình này giúp tạo ra dự báo chính xác hơn bằng cách sử dụng các phép chiếu phi tuyến.

$$h_{\ell} = \text{MLP}_{\ell} \left(y_{t-L:t,\ell}^{(p)} \right)$$

$$\theta_{\ell}^f = LINEAR^f(h_{\ell})$$

$$\theta_{\ell}^b = LINEAR^b(h_{\ell})$$

Nội suy phân cấp (Hierarchical Interpolation): N-HITS sử dụng nội suy thời gian. Các hệ số nội suy được xác định theo tỷ lệ biểu đạt (expressiveness ratio) r_{ℓ} điều chỉnh số lượng tham số trên mỗi đơn vị thời gian đầu ra, $|\theta_{\ell}^f| = \lceil r_{\ell} H \rceil$. Để khôi phục lại tốc độ lấy mẫu ban đầu và dự báo tất cả H điểm trong tầm nhìn, chúng ta sử dụng nội suy thời gian thông qua hàm nội suy g :

$$\hat{y}_{\tau,\ell} = g(\tau, \theta_{\ell}^f), \quad \forall \tau \in \{t+1, \dots, t+H\},$$

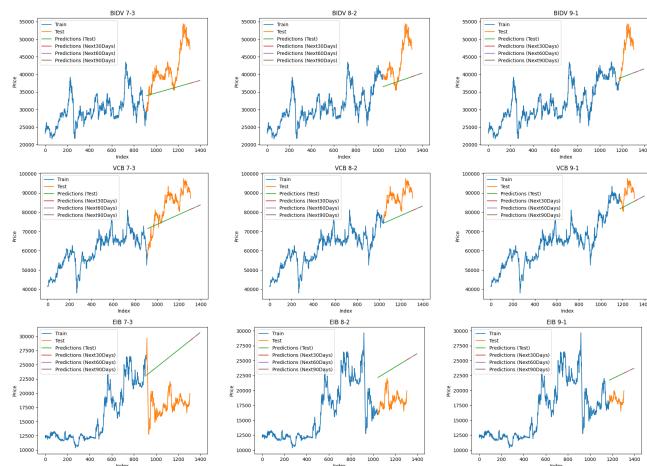
$$\tilde{y}_{\tau,\ell} = g(\tau, \theta_{\ell}^b), \quad \forall \tau \in \{t-L, \dots, t\}.$$



V. KẾT QUẢ

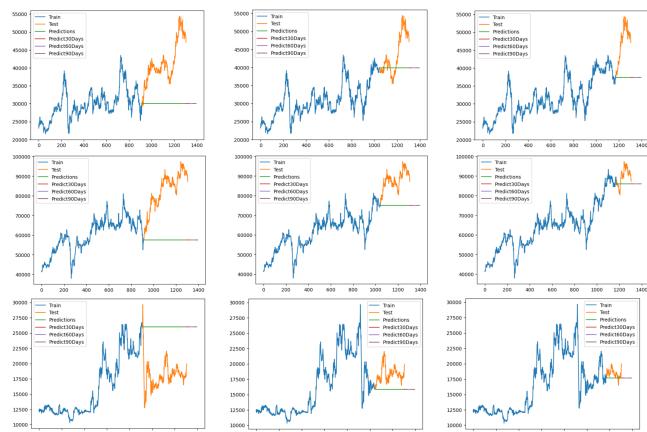
A. THIẾT LẬP MÔ HÌNH

1) Linear Regression



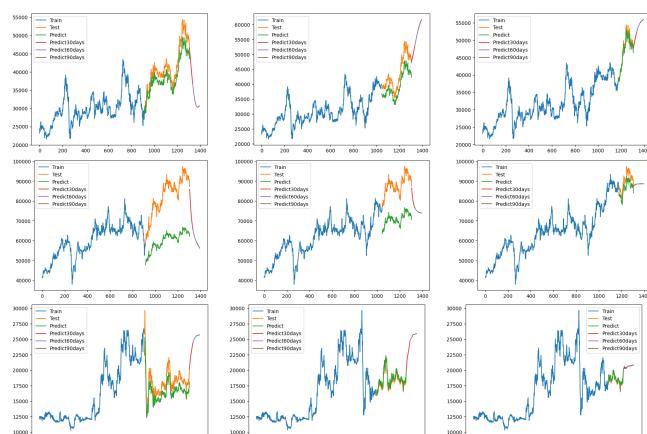
Hình 7. Kết quả chạy của mô hình Linear Regression

2) ARIMA



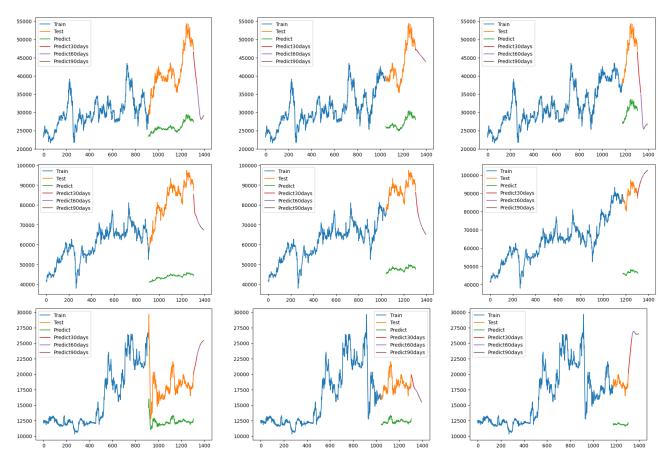
Hình 8. Kết quả chạy của mô hình ARIMA

3) RNN



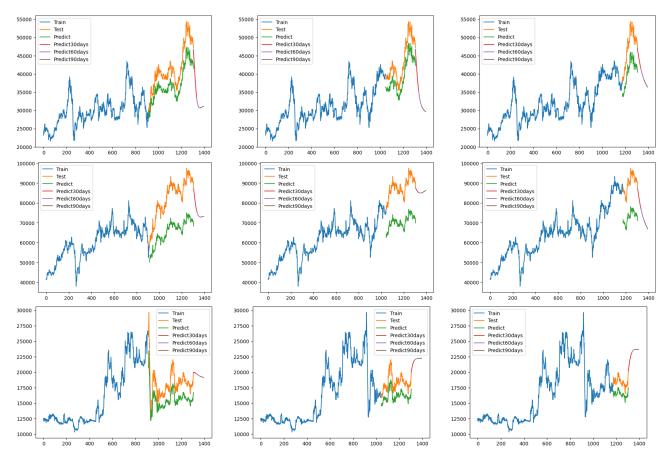
Hình 9. Kết quả chạy của mô hình RNN

4) LSTM



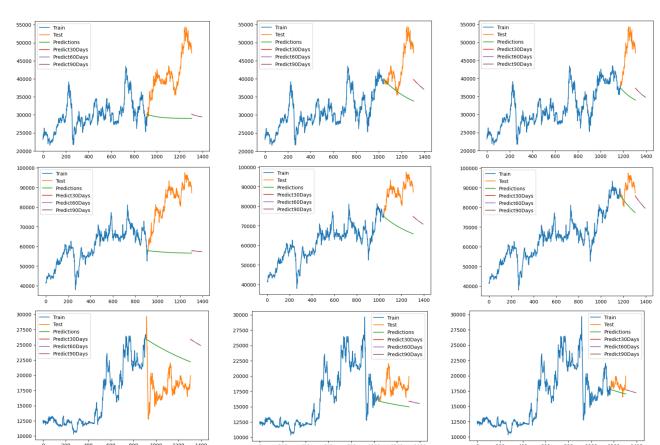
Hình 10. Kết quả chạy của mô hình LSTM

5) GRU



Hình 11. Kết quả chạy của mô hình GRU

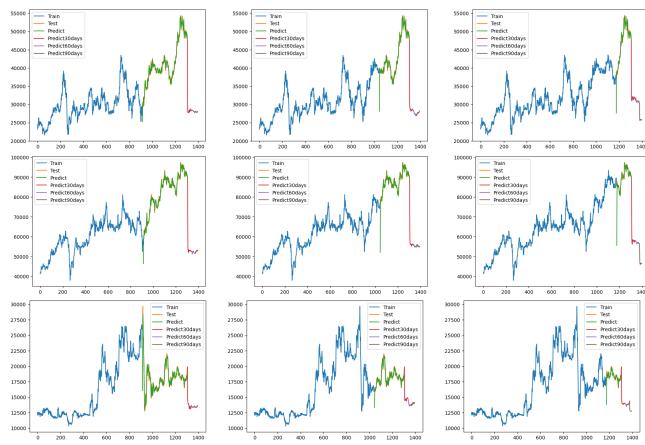
6) VARMA



Hình 12. Kết quả chạy của mô hình VARMA

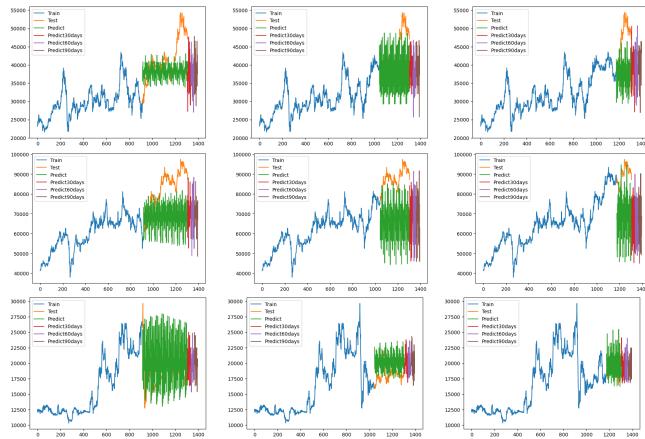


7) Kalman Filter



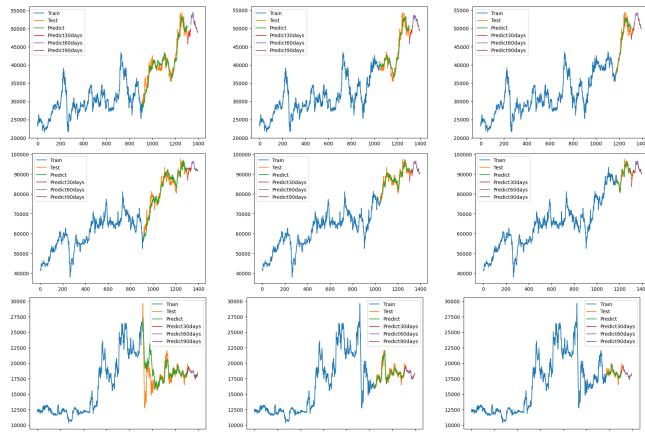
Hình 13. Kết quả chạy của mô hình Kalman Filter

8) Meta-Learning



Hình 14. Kết quả chạy của mô hình Meta-Learning

9) NBeats



Hình 15. Kết quả chạy của mô hình NBeats

10) N-HiTS



Hình 16. Kết quả chạy của mô hình N-HiTS

B. ĐÁNH GIÁ MÔ HÌNH

1) Đánh giá mô hình trên bộ dữ liệu BIDV

Bảng 2. Đánh giá trên bộ dữ liệu BIDV cho 5 thuật toán đầu

Đánh giá trên bộ dữ liệu BIDV				
Model	Training:Testing	RMSE	MAE	MAPE (%)
Linear Regression	7:3	7545.22	6125.68	13.75
	8:2	7336.11	5831.44	12.41
	9:1	8937.58	7874.29	15.81
ARIMA	7:3	12711.53	11427.93	26.26
	8:2	6425.28	4707.98	9.91
	9:1	11133.30	10093.96	20.43
RNN	7:3	2735.05	2571.72	6.01
	8:2	4153.80	3986.16	8.97
	9:1	836.06	719.74	1.44
LSTM	7:3	16066.17	15502.88	36.67
	8:2	16865.72	16435.48	37.27
	9:1	17021.76	16788.22	35.11
GRU	7:3	4512.19	4337.92	10.24
	8:2	4242.52	4140.48	9.39
	9:1	6816.77	6713.13	14.02

Bảng 3. Đánh giá trên bộ dữ liệu BIDV cho 5 thuật toán sau

Đánh giá trên bộ dữ liệu BIDV				
Model	Training:Testing	RMSE	MAE	MAPE (%)
VARMA	7:3	13533.87	12240.34	28.18
	8:2	9806.60	7267.22	15.19
	9:1	13211.38	12005.67	24.31
Kalman Filter	7:3	347.07	183.11	0.46
	8:2	808.84	224.45	0.57
	9:1	1023.80	304.91	0.78
Meta Learning	7:3	6687.32	4953.23	11.22
	8:2	8905.57	7149.11	115.72
	9:1	11619.81	10248.67	20.89
NBeats	7:3	1568.92	1225.05	3.02
	8:2	1333.41	993.17	2.29
	9:1	1452.04	1122.96	2.31
N-HiTS	7:3	761.21	528.28	1.28
	8:2	731.64	506.20	1.15
	9:1	822	575.08	1.19



2) Đánh giá mô hình trên bộ dữ liệu Vietcombank (VCB)

Bảng 4. Đánh giá trên bộ dữ liệu VCB cho 5 thuật toán đầu

Đánh giá trên bộ dữ liệu VCB				
Model	Training:Testing	RMSE	MAE	MAPE (%)
Linear Regression	7:3	9193.28	8187.25	9.7
	8:2	10964.97	10382.15	11.63
	9:1	7267.68	6470.07	7.03
ARIMA	7:3	26725.46	25101.68	29.39
	8:2	13823.10	13075.91	14.63
	9:1	5841.48	5018.32	5.46
RNN	7:3	23247.02	22751.47	27.17
	8:2	17205.25	17119.41	19.40
	9:1	3804.23	3737.46	4.12
LSTM	7:3	39672.67	38853.78	46.43
	8:2	40424.90	40272.32	45.67
	9:1	43470.64	43337.60	48.06
GRU	7:3	17528.59	17192.13	20.56
	8:2	17643.62	17581.13	19.94
	9:1	17085.98	17035.85	18.89

Bảng 5. Đánh giá trên bộ dữ liệu VCB cho 5 thuật toán sau

Đánh giá trên bộ dữ liệu VCB				
Model	Training:Testing	RMSE	MAE	MAPE (%)
VARMA	7:3	27394.80	25693.65	30.07
	8:2	19404.06	18300.77	20.48
	9:1	10751.45	9174.86	9.94
Kalman Filter	7:3	719.21	317.71	0.42
	8:2	1639.78	396.68	0.53
	9:1	3006.35	602.69	0.85
Meta Learning	7:3	17732.50	15385.31	17.92
	8:2	24116.94	22014.26	24.83
	9:1	24780.37	22537.83	24.92
NBeats	7:3	2489.45	1960.15	2.45
	8:2	1903.88	1444.53	1.64
	9:1	1565.97	1121.88	1.24
N-HiTS	7:3	1139.77	795.18	0.97
	8:2	1107.53	805.27	0.91
	9:1	1032.32	714.59	0.79

3) Đánh giá mô hình trên bộ dữ liệu Eximbank (EIB)

Bảng 6. Đánh giá trên bộ dữ liệu EIB cho 5 thuật toán đầu

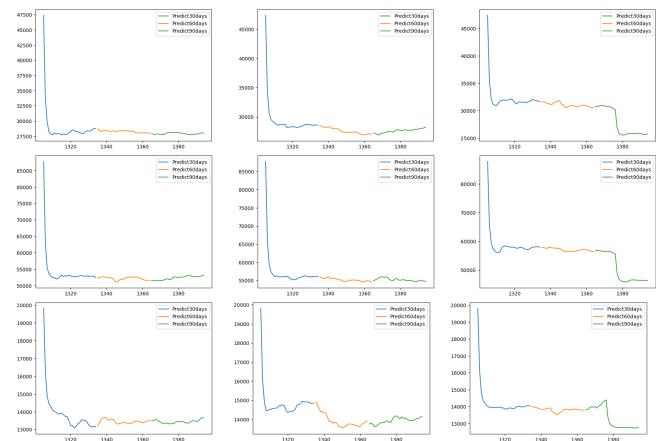
Đánh giá trên bộ dữ liệu EIB				
Model	Training:Testing	RMSE	MAE	MAPE (%)
Linear Regression	7:3	8518.18	8267.90	46.95
	8:2	5503.29	5322.43	29.54
	9:1	3966.94	3875.98	21.20
ARIMA	7:3	8173.63	7950.99	45.61
	8:2	2768.02	2480.93	13.16
	9:1	997.83	823.84	4.38
RNN	7:3	1606.59	1535.58	8.32
	8:2	309.26	308.76	1.69
	9:1	139.01	137.69	0.75
LSTM	7:3	5967.58	5772.39	31.41
	8:2	6008.25	5937.59	32.22
	9:1	6553.26	6531.35	35.36
GRU	7:3	2625.95	2526.05	13.71
	8:2	2183.42	2150.93	11.65
	9:1	1978.61	1969.27	10.65

Bảng 7. Đánh giá trên bộ dữ liệu EIB cho 5 thuật toán sau

Đánh giá trên bộ dữ liệu EIB				
Model	Training:Testing	RMSE	MAE	MAPE (%)
VARMA	7:3	6311.07	5876.39	34.13
	8:2	3202.22	2928.74	15.61
	9:1	13211.38	12005.67	24.32
Kalman Filter	7:3	593.36	179.24	0.98
	8:2	230.39	117.18	0.66
	9:1	415.36	135.54	0.79
Meta Learning	7:3	4699.01	3753.18	21.28
	8:2	2554.86	2115.52	11.81
	9:1	2573.81	2034.43	11.12
NBeats	7:3	1923.79	1151.80	6.84
	8:2	556.92	414.03	2.22
	9:1	467.71	353.59	1.91
N-HiTS	7:3	461.21	318.75	1.81
	8:2	372.65	255.28	1.38
	9:1	294.03	205.03	1.11

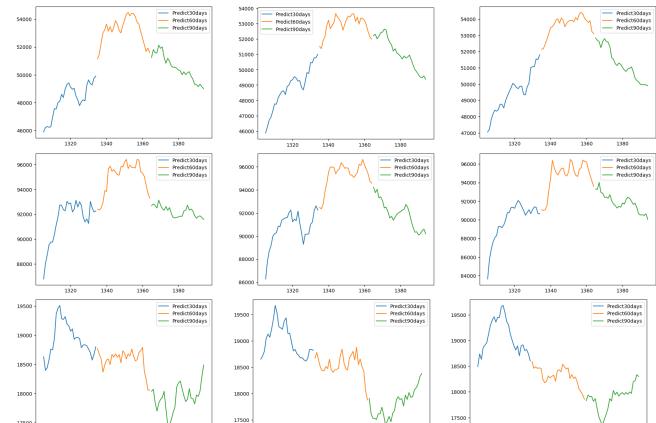
C. DỰ ĐOÁN GIÁ CHO 90 NGÀY TỚI CỦA 3 MÔ HÌNH TỐT NHẤT

1) Kalman Filter



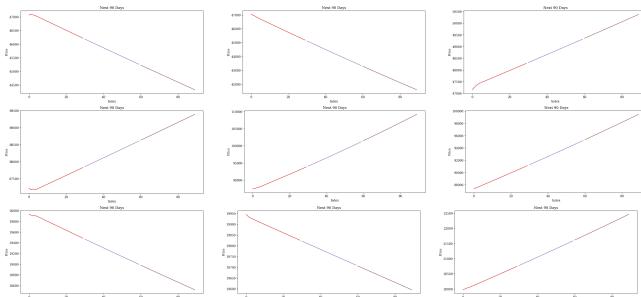
Hình 17. Dự đoán 90 ngày tới của mô hình Kalman Filter

2) NBeats



Hình 18. Dự đoán 90 ngày tới của mô hình NBeats

3) N-HiTS



Hình 19. Dự đoán 90 ngày tới của mô hình N-HiTS

VI. KẾT LUẬN

Thông qua các cuộc thử nghiệm ở cả 10 mô hình được thực hiện với bộ dữ liệu của 3 ngân hàng BIDV, Vietcombank, Eximbank, ta thấy có 3 mô hình có được kết quả dự báo giá cổ phiếu tốt nhất là Kalman Filter, NBeats và N-HiTS. Từ đó, ta thấy việc phát triển các mô hình thống kê, mô hình máy học thực sự đã tạo ra các mô hình dự đoán có độ chính xác cao hơn. Trong tương lai, chúng tôi dự tính sẽ nghiên cứu thêm các mô hình mới khác, để xem xét nó có được áp dụng cho dữ liệu giá cổ phiếu hay không.

LỜI CẢM ƠN

Nhóm chúng em xin được gửi lời cảm ơn sâu sắc đến PGS.TS Nguyễn Đình Thuân, người đã cung cấp chi tiết những kiến thức từ cơ bản đến nâng cao, giúp chúng em có được sự nhìn nhận sâu sắc về việc phân tích dữ liệu trong kinh doanh. Chúng em nhận thấy việc hiểu được lý thuyết cơ bản tốt thì sẽ giúp ích cho việc thực hiện các nghiên cứu cao hơn.

Ngoài ra, nhóm chúng em cũng xin được gửi lời cảm ơn đến Trợ giảng Nguyễn Minh Nhựt đã tận hình hỗ trợ chúng em trong việc thực hiện các nghiên cứu này. Anh Nguyễn Minh Nhựt đã hỗ trợ chúng em về các tài liệu để nghiên cứu, cũng như kiểm tra và đánh giá các thuật toán mà chúng em đã làm.

TÀI LIỆU

- [1] A. M. Priyatno, L. S. Tanjung, W. F. Ramadhan, P. Cholidhazia, P. Z. Jati, and F. I. Firmananda, "Comparison Random Forest Regression and Linear Regression For Forecasting BBCA Stock Price," *Jurnal Teknik Industri Terintegrasi*, vol. 6, no. 3, pp. 718-732, Jul. 2023.
- [2] M. Ridwan, K. Sadik, and F. M. Afendi, "Comparison of ARIMA and GRU Models for High-Frequency Time Series Forecasting," *Scientific Journal of Informatics*, vol. 10, no. 3, pp. 389-400, Jan. 2024.
- [3] S. Zaheer et al., "A Multi Parameter Forecasting for Stock Time Series Data Using LSTM and Deep Learning Model," vol. 11, no. 3, p. 590, Jan. 2023.
- [4] P. Mondal, L. Shit, and S. Goswami, "Study of Effectiveness of Time Series Modeling (Arima) in Forecasting Stock Prices," *International Journal of Computer Science, Engineering and Applications*, vol. 4, no. 2, p. 13, Apr. 2014.
- [5] S.-H. Chang et al., "Short-Term Stock Price-Trend Prediction Using Meta-Learning," in *IEEE International Conference on Systems, Man and Cybernetics*, 2021, doi: 10.48550/arXiv.2105.13599.

- [6] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions," Faculty of Information Technology, Monash University, Melbourne, Australia, arXiv:1909.00590v5 [cs.LG], Dec. 23, 2020.
- [7] B. N. Oreshkin, N. Chapados, D. Carpio, and Y. Bengio, "N-BEATS: Neural Basis Expansion Analysis for Interpretable Time Series Forecasting," arXiv:1905.10437v4 [cs.LG], Feb. 20, 2020.
- [8] C. Challu, K. G. Olivares, B. N. Oreshkin, F. Garza, M. Mergenthaler-Canseco, A. Dubrawski, "N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting," arXiv:2201.12886v6 [cs.LG], Nov. 29, 2022.
- [9] Prakhar Prasad, "Time Series - VARMA," [Online], Available: <https://www.kaggle.com/code/prakharprasad/time-series-varma>.
- [10] A. H. Al-Nasser and L. T. Abdullah, "The Estimators of Vector Autoregressive Moving Average Model VARMA of Lower Order: VARMA (0,1), ARMA (1,0), VARMA (1,1), VARMA (1,2), VARMA (2,1), VARMA (2,2) with Forecasting," *Iraqi Academics Syndicate International Conference for Pure and Applied Sciences, Journal of Physics: Conference Series*, vol. 1818, p. 012145, 2021. doi: 10.1088/1742-6596/1818/1/012145.
- [11] Michael Fuchs, "Time Series Analysis - Regression Extension Techniques for Multivariate Time Series," [Online], Available: <https://michael-fuchs-python.netlify.app/2020/10/29/time-series-analysis-regression-extension-techniques-for-forecasting-multivariate-variables/#varma>.
- [12] sabankara, "Recurrent Neural Networks for Time Series," May 25, 2023, [Online]. Available: <https://sabankara.medium.com/recurrent-neural-networks-for-time-series-b3132a6afb6a>.
- [13] Afshine Amidi and Shervine Amidi, "Recurrent Neural Networks cheat sheet," [Online]. Available: <https://sabankara.medium.com/recurrent-neural-networks-for-time-series-b3132a6afb6a>.
- [14] Yuqiang Pan, "Data label for time series mining (Part 4) Interpretability Decomposition Using Label Data," [Online]. Available: <https://www.mql5.com/en/articles/13218>.
- [15] Daniel Bourke, "Using the N-BEATS algorithm to predict the price of Bitcoin (time series with TensorFlow)," [Online Video], Available: <https://www.youtube.com/watch?v=pUUhQyUEZSk>.
- [16] Daniel Bourke, "Milestone Project 3: Time series forecasting in TensorFlow (BitPredict)," [Online], Available: https://github.com/mrdbourke/tensorflow-deep-learning/blob/main/10_time_series_forecasting_in_tensorflow.ipynb.
- [17] Cristian Challu, "NHITS Long-Horizon Forecast," [Online], Available: https://github.com/Nixtla/neuralforecast/blob/main/nbs/examples/LongHorizon_with_NHITS.ipynb.
- [18] Nixtla, "Neuralforecast NHITS and AutoNHITS," [Online], Available: <https://github.com/Nixtla/neuralforecast>.
- [19] Jonte Dancker, "N-HiTS — Making Deep Learning for Time Series Forecasting More Efficient," May 31, 2024, [Online]. Available: <https://towardsdatascience.com/n-hits-making-deep-learning-for-time-series-forecasting-more-efficient-d00956fc3e93>.