# Forecasting Stock Price Accuracy for Vietnamese Pharmaceutical Firms through Ensemble Statistical, Machine Learning, and Deep Learning Models

**NGUYEN MINH DUY[1], NGUYEN CHI KHA [2], AND BUI DINH TRIEU[3]**

[1]Faculty of Information Systems, University of Information Technology, (e-mail: 21522005@gm.uit.edu.vn)
[2]Faculty of Information Systems, University of Information Technology, (e-mail: 21522179@gm.uit.edu.vn)
[3]Faculty of Information Systems, University of Information Technology, (e-mail: 21521576@gm.uit.edu.vn)

**ABSTRACT** Stock price prediction plays a crucial role in investment decision-making and portfolio management strategies. This study aims to forecast the stock prices of three major pharmaceutical companies listed on the Vietnamese stock exchange: **Hataphar (DHT)**, **Central Pharmaceutical JSC No3 (DP3)**, and **American Vietnamese Biotech Inc (AMV)**.

The research employs a ensemble of statistical and machine learning models, including Fast Fourier Transform (FFT), Linear Regression with Calendar/Fourier Terms, Deterministic Process Models, LightGBM, ARIMA, Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs), and Long Short-Term Memory (LSTM) networks. These diverse techniques are utilized to capture the complex patterns and dynamics inherent in stock price movements.

The research evaluates model performance and identifies key factors influencing price fluctuations in Vietnam's pharmaceutical sector. Findings provide insights for investment strategies and portfolio optimization in this industry. The accurate forecasting models enable informed decision-making by investors, analysts and stakeholders.

**INDEX TERMS** Vietnamese stock market, Machine learning models, Statistical models, LightGBM, Recurrent neural networks, LSTM, GRU, ARIMA, Linear regression, Calendar/Fourier terms, pharma companies, AMV, DP3, DHT, FFT, Deterministic process models.

## I. INTRODUCTION

The Covid-19 pandemic has severely disrupted global economies, with the pharmaceutical industry at the forefront combating this crisis. As demand for vaccines, treatments, and medical supplies spiked, pharmaceutical firms faced immense pressures, leading to significant volatility in their stock prices. In Vietnam, companies like **Hataphar**, **Central Pharmaceutical No3**, and **American Vietnamese Biotech** played vital roles supplying Covid-19 countermeasures amidst shifting market dynamics.

Traditional forecasting models may not fully capture the unprecedented pandemic's complexities impacting stock movements. This necessitates exploring advanced machine learning and deep learning techniques to effectively model intricate patterns and non-linearities. This study aims to leverage an ensemble of statistical methods (FFT, Linear Regression with Calendar/Fourier terms, ARIMA) and machine learning models (LightGBM, RNNs, GRUs, LSTMs) for forecasting stock prices of the three major Vietnamese pharmaceutical firms.

By combining diverse approaches, the research seeks insights into factors influencing price dynamics during Covid-19. Findings can contribute to financial forecasting knowledge, guiding investment strategies, portfolio optimization, and risk management for stakeholders in Vietnam's pharmaceutical sector. Reliable forecasts can support the industry's resilience and growth amidst future crises.

## II. RELATED WORKS

Xiao and Wang (2017) [1] applied ARIMA, LSTM and a hybrid ARIMA-LSTM model to forecast stock prices of two companies listed on Chinese stock exchanges. Using data from 2009-2016, they found the hybrid model outperformed individual ARIMA and LSTM across evaluation metrics like RMSE and MAPE, improving accuracy by 10-15%.

Nelson et al. (2017) [2] compared statistical (ARIMA, Exponential Smoothing) and machine learning models (ANN, LSTM, GBR) for forecasting stock volatility of the S&P

500 index. Using 60 years of daily data, they showed LSTM networks were most effective in capturing nonlinear patterns and long-range temporal dependencies.

Bao et al. (2017) [3] proposed a novel wavelet transformer combining wavelet analysis and self-attention for stock movement prediction on high-frequency intraday data of MSCI USA Index constituents in 2017. Evaluated against statistical baselines, ARIMA and Exponential Smoothing, the wavelet transformer demonstrated superior directional prediction accuracy.

Minh et al. (2018) [4] explored CNN, LSTM and hybrid CNN-LSTM architectures for forecasting stock prices of major companies listed on the Vietnamese stock exchange like VNM, VCB and BVH, using data from 2012-2017. Their CNN-LSTM model incorporating technical indicators outperformed individual CNN and LSTM networks.

Chen et al. (2020) [5] developed an adaptive, dynamic method based on time series clustering for stock forecasting. Applied to constituent stocks of S&P 500 and CSI 300 indexes from 2013-2018, their approach using K-means clustering and Exponential Smoothing outperformed classical statistical models like ARIMA.

Mamon et al. (2021) [6] conducted a comparative analysis assessing the performance of ARIMA, VAR, CNN, LSTM and hybrid CNN-LSTM models on predicting Philippine stock index PSEi returns during the Covid-19 pandemic from 2020-2021 data. The CNN-LSTM exhibited the lowest error rates across metrics.

Jolly Masih [7] conducted a study that implemented algorithms such as SVM and LSTM on stock market data to observe whether major IT companies experienced growth or decline during the COVID-19 pandemic. The authors also utilized the ARIMA forecasting method to predict the stock prices of the mentioned four companies: Google, Microsoft, Apple, and Amazon. Regarding the current prediction of stock prices, the authors achieved an accuracy of 75.95% using the SVM algorithm and the best-fit forecast model of ARIMA. This decrease in accuracy was anticipated, as described earlier, due to increased volatility in the stock market resulting from the COVID-19 outbreak.

Ke et al. (2017) [20] introduced LightGBM, a highly efficient gradient boosting decision tree algorithm for classification and regression tasks. This method has gained significant attention for its superior performance, efficient handling of high-dimensional data, and effective parallelization. Several studies have successfully applied LightGBM to stock price prediction problems, leveraging its ability to capture nonlinear relationships and feature interactions.

## III. MATERIALS

### A. DATASET

We have compiled three datasets in the pharmaceutical field from three companies **Hataphar (DHT)**, **Central Pharmaceutical JSC No3 (DP3)**, and **American Vietnamese Biotech Inc (AMV)**. The collected data is the historical stock price of the three companies from 1-1-2019 to 17/4/2024. The data was collected from Investing.com website and includes columns:

**TABLE 1.** Data's attributes meaning

| Attributes | Description |
|---|---|
| Date | The date when the stock price data was recorded. |
| Price | Closing price of the stock for each date. |
| Open | Opening price of the stock for each date. |
| High | Highest price of the stock during the trading session. |
| Low | Lowest price of the stock during the trading session. |
| Vol. | Trading volume of the stock on each date. |
| Change % | Percentage change in the stock price compared to the previous trading day's closing price. |

The objective is to predict the price (close price), so only descriptive statistical data relating to the "Price" column will be processed.

### B. DESCRIPTIVE STATISTICS

**TABLE 2.** DHT, DP3, AMV's Descriptive Statistics

| | DHT | DP3 | AMV |
|---|---|---|---|
| Count | 1320 | 1146 | 1322 |
| Mean | 16,583 | 86,519 | 11,167 |
| Std | 4,149 | 21,584 | 5,811 |
| Min | 9,384 | 57,600 | 2,700 |
| 25% | 13,961 | 68,000 | 4,900 |
| 50% | 15,778 | 81,000 | 11,359 |
| 75% | 18,133 | 103,000 | 15,123 |
| Max | 29,400 | 157,500 | 24,589 |

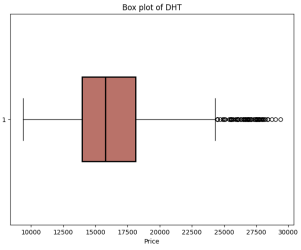


**FIGURE 1.** DHT stock price's boxplot

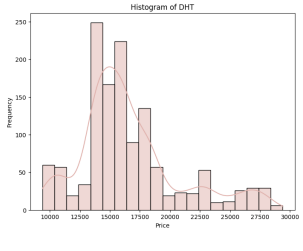


**FIGURE 2.** DHT stock price's histogram

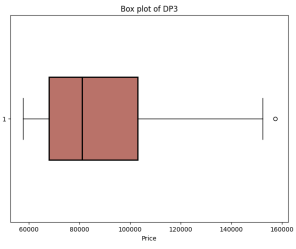


**FIGURE 3.** DP3 stock price's boxplot

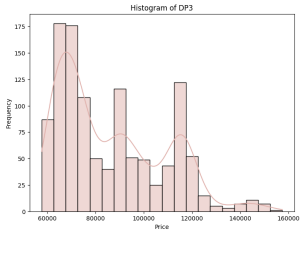


**FIGURE 4.** DP3 stock price's histogram

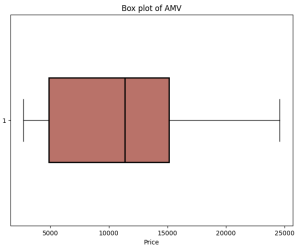


**FIGURE 5.** AMV stock price's boxplot

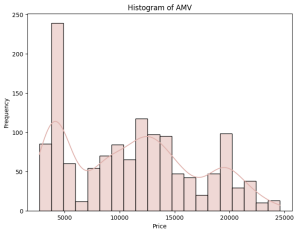


**FIGURE 6.** AMV stock price's histogram

## IV. METHODOLOGY

### A. LINEAR REGRESSION

Regression analysis is a tool for building mathematical and statistical models that characterize relationships between a dependent variable and one or more independent, or explanatory, variables, all of which are numerical. This statistical technique is used to find an equation that best predicts the y variable as a linear function of the x variables. A multiple linear regression model has the form:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon$$

Where:

- Y is the dependent variable (Target Variable).
- $X_1, X_2, \ldots, X_k$ are the independent (explanatory) variables.
- $\beta_0$ is the intercept term.
- $\beta_1, ..., \beta_k$ are the regression coefficients for the independent variables.
- $\varepsilon$ is the error term.

### B. FAST FOURIER TRANSFORM

The Fast Fourier Transform (FFT) is a method that allows computing the Discrete Fourier Transform (DFT) in $\mathcal{O}(n \log n)$ time. The basic idea of the FFT is to apply the divide-and-conquer approach. We divide the coefficient vector of the polynomial into two vectors, recursively compute the DFT for each of them, and combine the results to compute the DFT of the complete polynomial.

Let there be a polynomial $A(x)$ with degree $n-1$, where $n$ is a power of 2, and $n > 1$:

$$|A(x)| = a_0 x^0 + a_1 x^1 + \cdots + a_{n-1} x^{n-1}$$

Where:

- $|A(x)|$ is the absolute value of the polynomial $A(x)$.
- $a_0, a_1, \ldots, a_{n-1}$ are the coefficients of the polynomial.
- $x^0, x^1, \ldots, x^{n-1}$ are the corresponding powers of the variable $x$ for each coefficient.
- $n-1$ is the highest degree of the polynomial.
- The $\cdots$ (read as "and so on") represents the remaining terms in the sequence.

We divide it into two smaller polynomials, one containing only the coefficients of the even positions, and the other containing the coefficients of the odd positions:

$$A_0(x) = a_0 x^0 + a_2 x^1 + \cdots + a_{n-2} x^{\frac{n}{2}-1}$$
$$A_1(x) = a_1 x^0 + a_3 x^1 + \cdots + a_{n-1} x^{\frac{n}{2}-1}$$

In the FFT computation, we apply the divide-and-conquer approach recursively to the polynomials $A_0(x)$ and $A_1(x)$, and combine the results to obtain the DFT of the original polynomial $A(x)$. Dividing the polynomial into smaller polynomials and computing recursively significantly reduces the computational time compared to the traditional DFT computation method.

### C. ARIMA

The ARIMA model (an acronym for Auto-Regressive Integrated Moving Average), essentially creates a linear equation which describes and forecasts your time series data. This equation is generated through three separate parts which can be described as:

- AR — auto-regression: equation terms created based on past data points.
- I — integration or differencing: accounting for overall "trend" in the data.
- MA — moving average: equation terms of error or noise based on past data points.

An ARIMA model is characterized by 3 terms: p, d, q where,

- p is the order of the AR term.
- q is the order of the MA term.
- d is the number of differencing required to make the time series stationary.

A pure Auto Regressive (AR only) model is one where $Yt$ depends only on its own lags. That is, $Yt$ is a function of the 'lags of $Yt$'.

$$Y_t = a + B_1 Y_{t-1} + B_2 Y_{t-2} + \ldots + B_p Y_{t-p} + \varepsilon_t$$

Where:

- $Y_t$: The value of the time series at time $t$.
- $a$: A constant term.
- $B_1, B_2, \ldots, B_p$: Autoregressive coefficients representing the relationship between the current observation and its past values at lag intervals.
- $p$: The order of the autoregressive component.
- $\varepsilon_t$: A white noise error term at time $t$.

Likewise a pure Moving Average (MA only) model is one where Yt depends only on the lagged forecast errors.

$$Y_t = a + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + \ldots + \phi_q \varepsilon_{t-q} + \varepsilon_t$$

Where:

- $q$: The order of the moving average component.

An ARIMA model is one where the time series was differenced at least once to make it stationary and you combine the AR and the MA terms. So the equation becomes:

$$Y_t = a + \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t + \sum_{j=1}^{q} B_j \varepsilon_{t-j}$$

To find the orders of AR, I, and MA components in an ARIMA model, analysts use ACF and PACF plots. A decreasing ACF suggests a MA component, with the lag where it drops below a threshold indicating its order. Significant spikes in PACF imply an AR component, with the lag where PACF cuts off after a spike revealing its order. By interpreting these plots, analysts select suitable orders for ARIMA modeling.

### D. LSTM

Long short-term memory is a type of recurrent neural network (RNN) aimed at dealing with the vanishing gradient problem present in traditional RNNs. Its relative insensitivity to gap length is its advantage over other RNNs, hidden Markov models and other sequence learning methods. It aims to provide a short-term memory for RNN that can last thousands of timesteps, thus "long short-term memory". It is applicable to classification, processing and predicting data based on time series, such as in handwriting, speech recognition, machine translation, speech activity detection, robot control, video games, and healthcare.

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. Forget gates decide what information to discard from a previous state by assigning a previous state, compared to a current input, a value between 0 and 1. A (rounded) value of 1 means to keep the information, and a value of 0 means to discard it. Input gates decide which pieces of new information to store in the current state, using the same system as forget gates. Output gates control which pieces of information in the current state to output by assigning a value from 0 to 1 to the information, considering the previous and current states. Selectively outputting relevant information from the current state allows the LSTM network to maintain useful, long-term dependencies to make predictions, both in current and future time-steps.

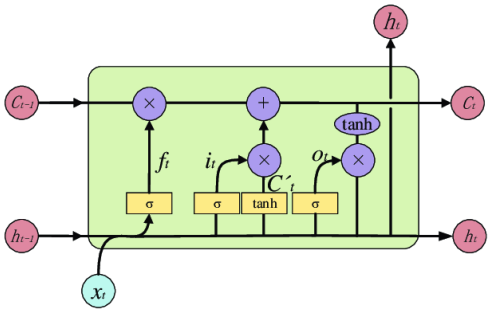


FIGURE 7. LSTM model's

### E. LINEAR REGRESSION COMBINED WITH CALENDARFOURIER AND DETERMINISTICPROCESS

Linear Regression combined with CalendarFourier and DeterministicProcess is an extended linear regression method used for modeling and forecasting time series with recurring cycles and growth trends. This method combines the following elements:

**Linear Regression**:

- Linear Regression is a common statistical modeling technique used to explore the linear relationship between an output variable (dependent variable) and one or more input variables (independent variables).
- Linear Regression is often used as a foundation for other time series forecasting techniques due to its ability to effectively model linear relationships.

**CalendarFourier**:

- CalendarFourier is a time feature used to model recurring patterns in time series data.
- It transforms time into Fourier functions (sine and cosine), allowing the model to learn cyclical patterns such as daily, weekly, monthly, or yearly cycles.
- When combined with Linear Regression, CalendarFourier enhances the ability to forecast recurring cycles in time series data.

**DeterministicProcess**:

- DeterministicProcess is a time feature used to model trends or growth patterns in time series data.
- It transforms time into a growth function (e.g., straight line, curve, polynomial, etc.), allowing the model to learn the long-term growth or decline trends of the data.
- When combined with Linear Regression, DeterministicProcess improves the ability to forecast long-term trends in time series data.

Suppose we have a time series `y` and input variables `X`. The Linear Regression model combined with CalendarFourier and DeterministicProcess takes the form:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n + f(t) + g(t) + \epsilon$$

Where:

- `y` is the output variable (the time series to be predicted).
- $\beta_0$ is the constant term.
- $\beta_1, \beta_2, ..., \beta_n$ are the regression coefficients for the input variables $X_1, X_2, ..., X_n$.
- `f(t)` is the CalendarFourier component, representing the recurring cycles in the time series, with `t` being the time.
- `g(t)` is the DeterministicProcess component, representing the growth trend in the time series, with `t` being the time.
- $\epsilon$ is the random error term.

The CalendarFourier component `f(t)` can be represented as a sum of sine and cosine functions with different cycle frequencies:

$$f(t) = \alpha_1 \sin(\omega t) + \beta_1 \cos(\omega t) + \alpha_2 \sin(2\omega t) + \beta_2 \cos(2\omega t) + ... + \alpha_k \sin(k\omega t) + \beta_k \cos(k\omega t)$$

Where:

- $\omega$ is the cycle frequency (e.g., $2\pi/365$ for an annual cycle).
- $k$ is the number of Fourier terms used to model the cycles.
- $\alpha_1, \beta_1, \alpha_2, \beta_2, ..., \alpha_k, \beta_k$ are the coefficients to be estimated from the data.

The DeterministicProcess component `g(t)` can be represented by a growth function, such as a polynomial, exponential, or logarithmic function:

$$g(t) = \gamma_1 t + \gamma_2 t^2 + \gamma_3 t^3 + ... \quad \text{(polynomial)}$$

$$g(t) = \gamma_1 + \gamma_2 \log(t) \quad \text{(logarithmic)}$$

$$g(t) = \gamma_1 \exp(\gamma_2 t) \quad \text{(exponential)}$$

Where $\gamma_1, \gamma_2, \gamma_3, ...$ are the coefficients to be estimated from the data.

The estimation of the coefficients $\beta_0, \beta_1, \beta_2, ..., \beta_n, \alpha_1, \beta_1, \alpha_2, \beta_2, ..., \alpha_k, \beta_k, \gamma_1, \gamma_2, \gamma_3, ...$ can be performed using the Ordinary Least Squares (OLS) method or other optimization techniques.

By combining these components, the Linear Regression model with CalendarFourier and DeterministicProcess can effectively model and forecast both recurring cycles and growth trends in the time series, thereby achieving higher forecasting accuracy compared to simpler models.

### F. GATED RECURRENT UNIT

The Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) architecture similar to Long Short-Term Memory (LSTM) but with a simpler design and fewer parameters. GRU employs two gates: the reset gate and the update gate. The reset gate controls the forgetting of the previous hidden state, while the update gate manages the incorporation of a candidate activation vector into the new hidden state.

The GRU processes sequential data one element at a time, updating its hidden state based on the current input and the previous hidden state. At each time step, it computes a "candidate activation vector" that combines information from the input and previous hidden state. This vector is updated using the reset gate, which controls how much of the previous hidden state to forget, and the update gate, which determines how much of the candidate activation vector to incorporate into the new hidden state.

Here's the math behind the GRU architecture:

1. The reset gate $r$ and update gate $z$ are computed using the current input $x$ and the previous hidden state $h_{t-1}$:

$$r_t = \text{sigmoid}(\mathbf{W}_r \cdot [h_{t-1}, x_t])$$

$$z_t = \text{sigmoid}(\mathbf{W}_z \cdot [h_{t-1}, x_t])$$

where $\mathbf{W}_r$ and $\mathbf{W}_z$ are weight matrices that are learned during training.

2. The candidate activation vector $\tilde{h}_t$ is computed using the current input $x$ and a modified version of the previous hidden state that is "reset" by the reset gate:

$$\tilde{h}_t = \frac{e^{r_t * h_{t-1}} + e^{-r_t * h_{t-1}}}{e^{r_t * h_{t-1}} - e^{-r_t * h_{t-1}}} \cdot x_t$$

where $\mathbf{W}_h$ is another weight matrix.

3. The new hidden state $h_t$ is computed by combining the candidate activation vector with the previous hidden state, weighted by the update gate:

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Overall, the reset gate controls the retention of the previous hidden state, and the update gate determines how much of the candidate activation vector updates the new hidden state. This compact architecture selectively updates its hidden state without needing a separate memory cell like LSTM.

### G. LIGHTGBM

LightGBM, short for Light Gradient Boosting Machine, is a gradient boosting framework developed by Microsoft. It's renowned for its efficiency and speed. The fundamental representation of how LightGBM combines predictions from different trees to make a final prediction can be represented by the following formula:

$$Y = \text{Base\_tree}(X) - \text{lr} \times \text{Tree}_1(X) - \text{lr} \times \text{Tree}_2(X) - \text{lr} \times \text{Tree}_3(X)$$

Here, Y is the final prediction, Base tree(X) is the prediction of the base model, Tree1(X), Tree2(X), and Tree3(X) are the predictions of subsequent models, and lr is the learning rate

LightGBM introduces two key techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB).

- GOSS focuses on sampling instances based on their gradients, prioritizing those with larger gradients for training. It also randomly drops instances with small gradients, thereby reducing the computational burden.
- EFB optimizes memory usage by bundling exclusive features together, allowing for more efficient processing and faster training without compromising the performance significantly.

LightGBM employs histogram-based algorithms to find the best split points during tree construction1. Instead of exhaustively searching through all possible split points, LightGBM discretizes continuous features into discrete bins and constructs histograms to approximate the optimal split points. This approach significantly reduces computational complexity and memory usage, especially for datasets with a large number of unique feature values.

### H. RECURRENT NEURAL NETWORK

Recurrent Neural Network (RNN) is a neural network architecture that enables processing sequential data in O(n) time complexity. The core idea of RNNs is to apply the divide-and-conquer approach. We divide the input sequence into time steps, recursively compute for each step, and combine the results to process the complete input sequence. Let $X$ be an input sequence of length $n$:

$$|X| = x_1, x_2, ..., x_n \tag{1}$$

Where:

- $|X|$ is the length of the input sequence $X$.
- $x_1, x_2, ..., x_n$ are the elements of the sequence.
- ... (read as "and so on") represents the remaining elements in the sequence.

We divide the input sequence into time steps, processing one element at each step:

$$h_1 = f(x_1, h_0)\ h_2 \quad = f(x_2, h_1)\ ...\ h_n \quad = f(x_n, h_{n-1}) \tag{2}$$

In which:

- $h_t$ is the hidden state at time step $t$.
- $f$ is a computation function, typically a non-linear transformation (e.g., tanh, ReLU) of the linear combination of the input $x_t$ and the previous hidden state $h_{t-1}$.
- $h_0$ is the initial hidden state, usually initialized as zeros.

In the RNN computation, we apply the divide-and-conquer approach recursively at each time step, computing the hidden state $h_t$ based on the input $x_t$ and the previous hidden state $h_{t-1}$. The final result is the hidden state $h_n$, which encapsulates the contextual information from the entire input sequence.

The output $y_t$ at each time step $t$ is computed based on the current hidden state $h_t$ and input $x_t$, typically through an output layer:

$$y_t = g(h_t, x_t) \tag{3}$$

Where $g$ is a function, often a linear transformation or a neural network.

Recurrent structure allows RNNs to capture long-range dependencies within the sequential data, making them suitable for various tasks such as language modeling, machine translation, speech recognition, and time series prediction.

## V. RESULT

### A. EVALUATION METHODS

**Mean Percentage Absolute Error** (MAPE): is the average percentage error in a set of predicted values.

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

**Root Mean Squared Error** (RMSE): is the square root of average value of squared error in a set of predicted values.

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

**Mean Absolute Error** (MSLE):is the relative difference between the log-transformed actual and predicted values.

$$MSLE = \frac{1}{n} \sum_{i=1}^{n} (log(1 + \hat{y_i}) - log(log(1 + y_i))^2$$

Where:

- $n$ is the number of observations in the dataset.
- $y_i$ is the true value.
- $\hat{y_i}$ is the predicted value.

### B. AMV DATASET

| AMV Dataset's Evaluation | | | | |
|---|---|---|---|---|
| Model | Training:Testing | RMSE | MAPE (%) | MSLE |
| LN | 7:3 | 8480.63 | 54.59% | 22.52% |
| | 8:2 | | | |
| | **9:1** | | | |
| SVR | 7:3 | | | |
| | 8:2 | | | |
| | **9:1** | | | |
| GRU | **7:3** | | | |
| | 8:2 | | | |
| | 9:1 | | | |
| ARIMA | 7:3 | | | |
| | 8:2 | | | |
| | **9:1** | | | |
| FFT | **7:3** | 11243.57 | 70.86% | 1.67 |
| | 8:2 | 9313.02 | 67.90% | 1.35 |
| | 9:1 | 9666.76 | 71.17% | 1.56 |
| LN-CF-DP | 7:3 | 6841.46 | 165.18 | 0.96 |
| | 8:2 | 2031.85 | 44.21 | 0.15 |
| | **9:1** | 1441.58 | 33.21 | 0.10 |
| LSTM | 7:3 | 691.72 | 2.68% | 0.0021 |
| | 8:2 | 437.58 | 2.67% | 0.00098 |
| | **9:1** | 485.37 | 2.72% | 0.0011 |
| LightGBM | 7:3 | 0.17 | 291.51 | 0.023 |
| | 8:2 | 0.007 | 22.24 | 4.78 |
| | **9:1** | 0.006 | 20.78 | 4.16 |

TABLE 3. AMV Dataset's Evaluation

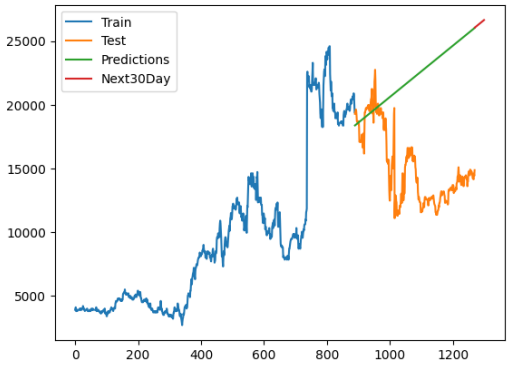


FIGURE 8. Linear model's result with 7:3 splitting proportion

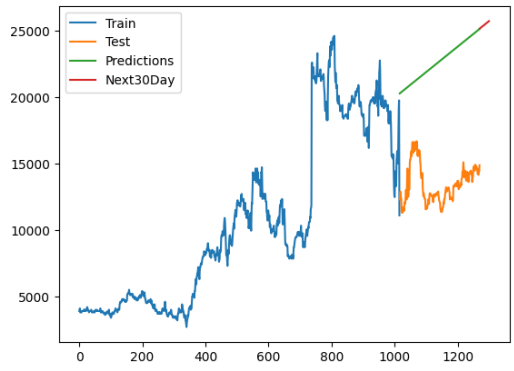


**FIGURE 9.** Linear model's result with 8:2 splitting proportion

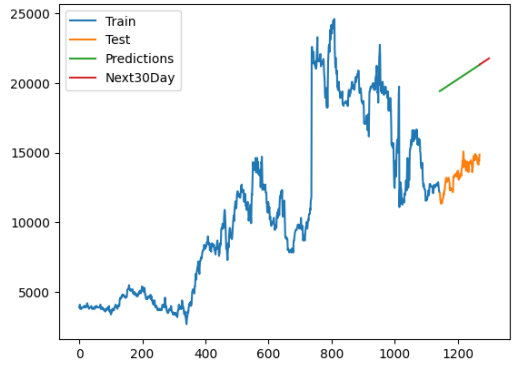


**FIGURE 10.** Linear model's result with 9:1 splitting proportion

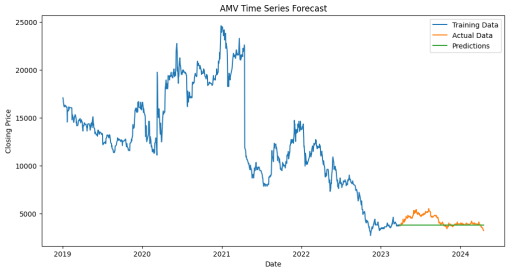


**FIGURE 11.** Arima model's result with 8:2 splitting proportion

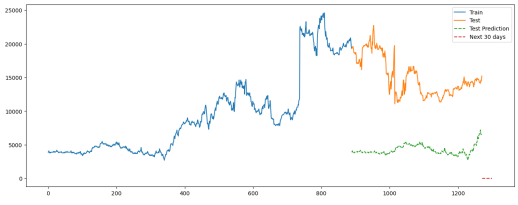


**FIGURE 12.** FFT model's result with 7:3 splitting proportion

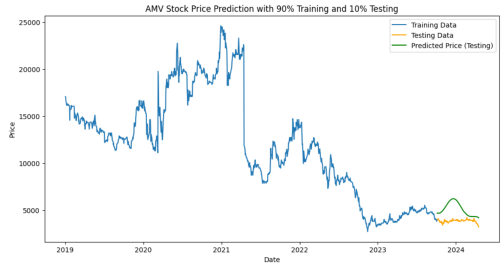


**FIGURE 13.** LN-CF-DP model's result with 9:1 splitting proportion

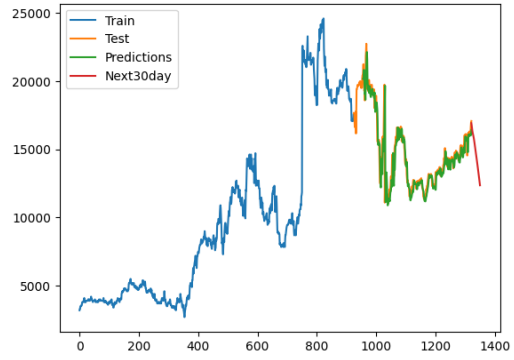


**FIGURE 14.** LSTM model's result with 7:3 splitting proportion

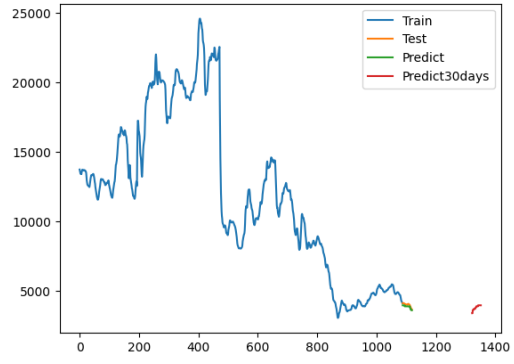


**FIGURE 15.** LightGBM model's result with 9:1 splitting proportion

### C. DP3 DATASET

| DP3 Dataset's Evaluation | | | | |
|---|---|---|---|---|
| Model | Training:Testing | RMSE | MAPE (%) | MSLE |
| LN | 7:3 | | | |
| | 8:2 | | | |
| | **9:1** | | | |
| SVR | 7:3 | | | |
| | 8:2 | | | |
| | **9:1** | | | |
| GRU | **7:3** | | | |
| | 8:2 | | | |
| | 9:1 | | | |
| ARIMA | 7:3 | | | |
| | 8:2 | | | |
| | **9:1** | | | |
| FFT | 7:3 | 27995.79 | 29.10% | 0.10 |
| | 8:2 | 21824.13 | 13.88% | 0.05 |
| | **9:1** | 10698.11 | 14.28% | 0.02 |
| LN-CF-DP | 7:3 | 54681.37 | 69.13 | 0.31 |
| | 8:2 | 48774.18 | 67.56 | 0.28 |
| | **9:1** | 45471.00 | 66.45 | 0.26 |
| LSTM | 7:3 | 2690.97 | 2.48% | 0.0013 |
| | 8:2 | 2066.61 | 1.7% | 0.00079 |
| | **9:1** | 2835.08 | 2.24% | 0.0013 |
| LightGBM | 7:3 | 0.08 | 186.63 | 0.003 |
| | 8:2 | 0.013 | 71.27 | 0.0001 |
| | **9:1** | 0.008 | 7.12 | 6.01 |

**TABLE 4.** DP3 Dataset's Evaluation

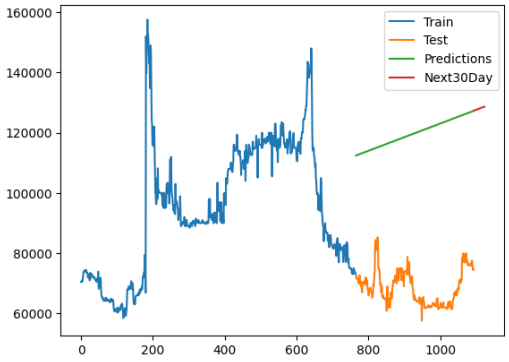


**FIGURE 16.** Linear model's result with 7:3 splitting proportion

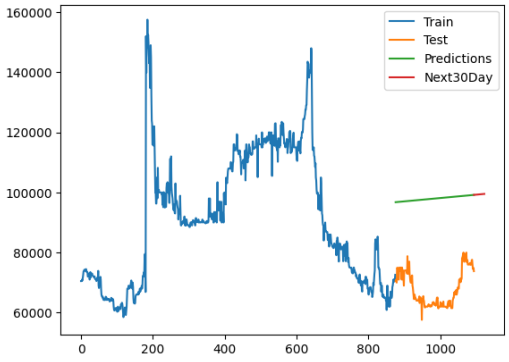


**FIGURE 17.** Linear model's result with 8:2 splitting proportion

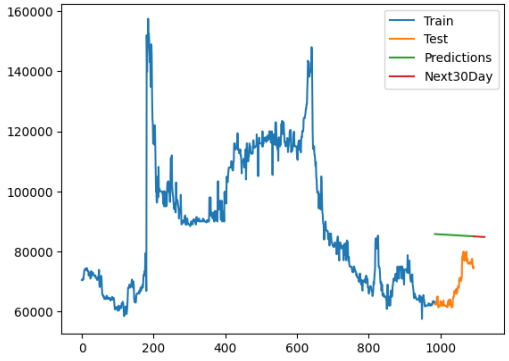


**FIGURE 18.** Linear model's result with 9:1 splitting proportion

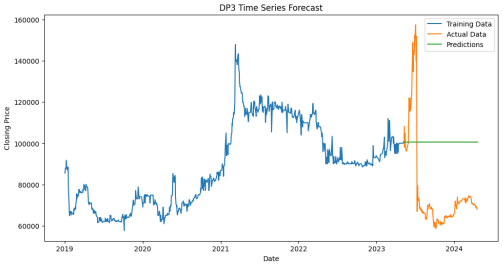


**FIGURE 19.** ARIMA model's result with 8:2 splitting proportion

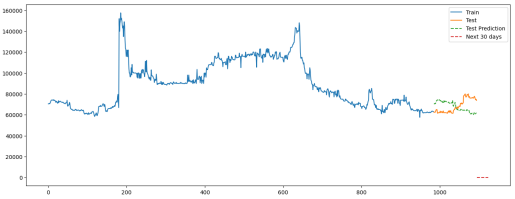

**FIGURE 20.** FFT model's result with 9:1 splitting proportion

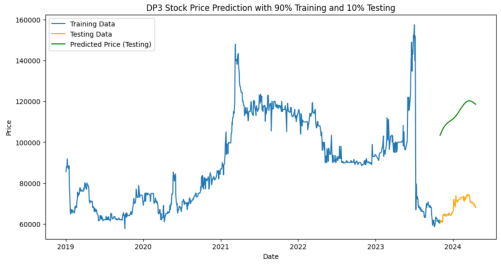


**FIGURE 21.** LN-CF-DP model's result with 9:1 splitting proportion

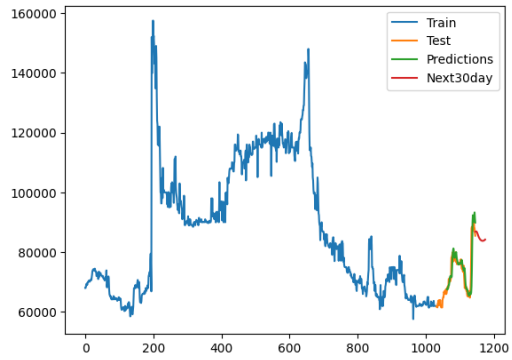


**FIGURE 22.** LSTM model's result with 9:1 splitting proportion

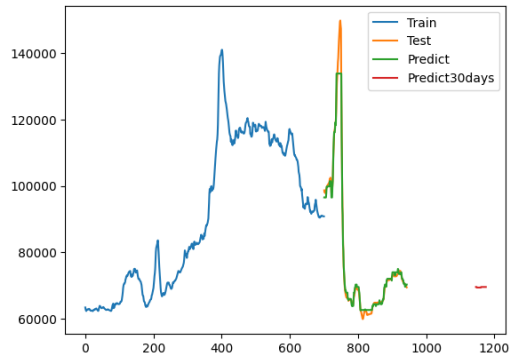


**FIGURE 23.** LightGBM model's result with 7:3 splitting proportion

### *D. DHT DATASET*

| DHT Dataset's Evaluation | | | | |
|---|---|---|---|---|
| Model | Training:Testing | RMSE | MAPE (%) | MSLE |
| LN | 7:3 | | | |
| | 8:2 | | | |
| | **9:1** | | | |
| SVR | 7:3 | | | |
| | 8:2 | | | |
| | **9:1** | | | |
| GRU | **7:3** | | | |
| | 8:2 | | | |
| | 9:1 | | | |
| ARIMA | 7:3 | | | |
| | 8:2 | | | |
| | **9:1** | | | |
| FFT | **7:3** | 6287.60 | 36.38% | 0.11 |
| | 8:2 | 9919.21 | 72.76% | 0.32 |
| | 9:1 | 12859.83 | 117.69% | 0.61 |
| LN-CF-DP | **7:3** | 5726.49 | 24.00 | 0.08 |
| | 8:2 | 8043.72 | 29.24 | 0.16 |
| | 9:1 | 6725.16 | 24.92 | 0.10 |
| LSTM | 7:3 | 497.09 | 2.65% | 0.0014 |
| | 8:2 | 524.89 | 3.44% | 0.00019 |
| | **9:1** | 387.75 | 3.10% | 0.0013 |
| LightGBM | 7:3 | 0.25 | 45.29 | 0.02 |
| | 8:2 | 0.27 | 32.08 | 0.03 |
| | **9:1** | 0.05 | 5.28 | 0.0006 |

**TABLE 5.** DHT Dataset's Evaluation

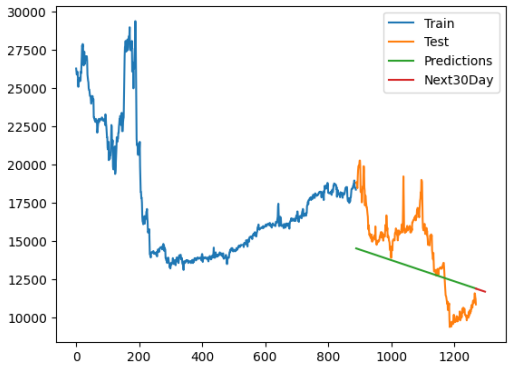


**FIGURE 24.** Linear model's result with 7:3 splitting proportion

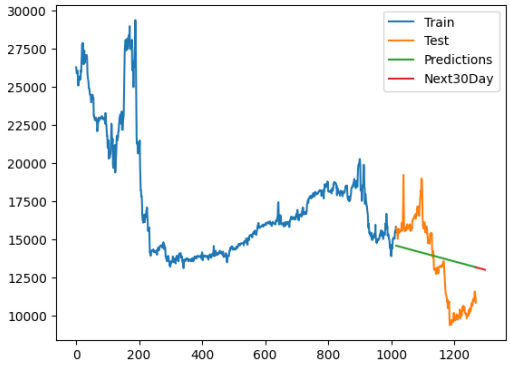


**FIGURE 25.** Linear model's result with 8:2 splitting proportion

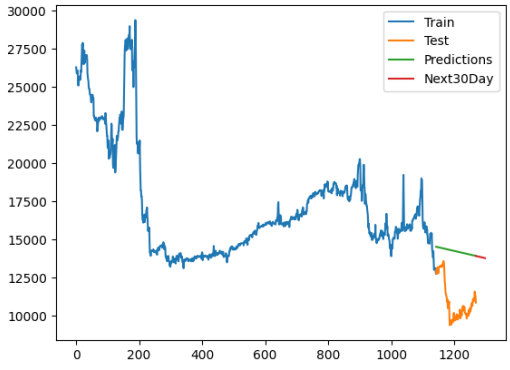


**FIGURE 26.** Linear model's result with 9:1 splitting proportion

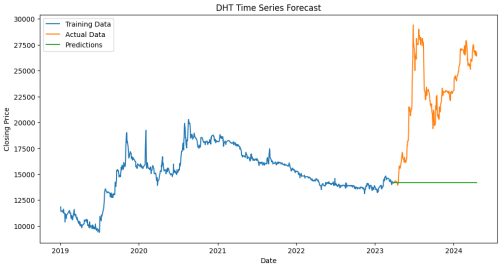


**FIGURE 27.** Arima model's result with 8:2 splitting proportion

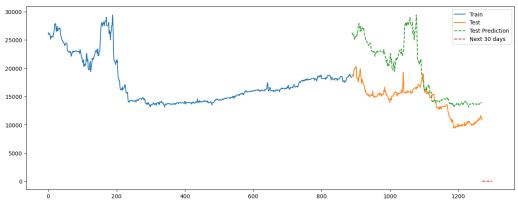


**FIGURE 28.** FFT model's result with 7:3 splitting proportion

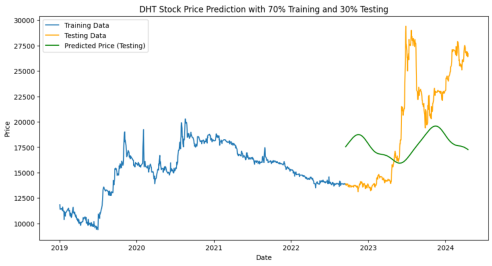


**FIGURE 29.** LN-CF-DP model's result with 7:3 splitting proportion

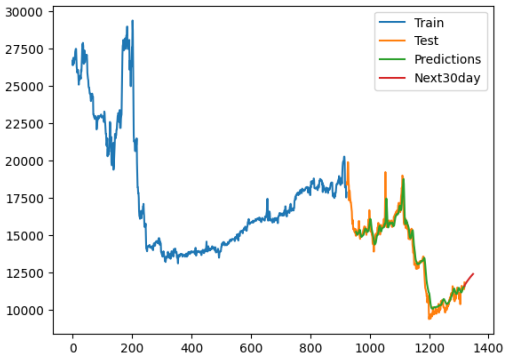


**FIGURE 30.** LSTM model's result with 8:2 splitting proportion

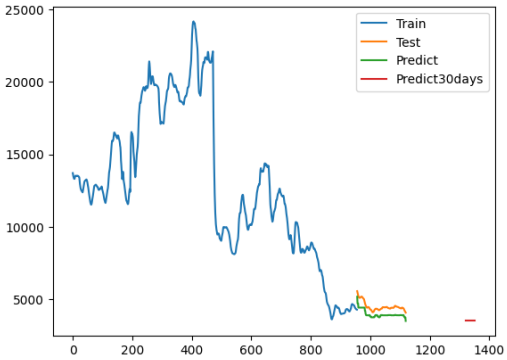


**FIGURE 31.** LightGBM model's result with 8:2 splitting proportion

## ACKNOWLEDGMENT

First and foremost, we would like to express our sincere gratitude to **Assoc. Prof. Dr. Nguyen Dinh Thuan** and **Mr. Nguyen Minh Nhut** for their exceptional guidance, expertise, and invaluable feedback throughout the research process. Their mentorship and unwavering support have been instrumental in shaping the direction and quality of this study. Their profound knowledge, critical insights, and attention to detail have significantly contributed to the success of this research.

This research would not have been possible without the support and contributions of our mentors. We would like to extend our heartfelt thanks to everyone involved for their invaluable assistance, encouragement, and belief in our research. Thank you all for your invaluable assistance and encouragement.

## REFERENCES

[1] Xiao, Y., & Wang, J. (2017). A hybrid machine learning approach to stock price prediction. arXiv preprint arXiv:1708.05419.

[2] Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. In International joint conference on neural networks (pp. 1419-1426).

[3] Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one, 12(7), e0180944.

[4] Minh, D. L., Sadeghi-Niaraki, A., Huy, H. D., Min, K., & Moon, H. (2018). Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network. IEEE Access, 6, 55392-55404.

[5] Chen, J., Li, K., Tang, Z., Bilal, K., Yu, S., Weng, C., & Li, K. (2020). A parallel random forest algorithm for big data in a spark cloud computing environment. IEEE Transactions on Parallel and Distributed Systems, 32(1), 1-17.

[6] Mamon, R. S., Arya, V., Kamalakannan, R., & Gadde, S. (2021). Forecasting Stock Index Returns During COVID-19: Machine Learning Fusion Approach. Mathematics, 9(14), 1641.