



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

NGUYEN MINH DUY¹, NGUYEN CHI KHA², AND BUI DINH TRIEU³

¹Faculty of Information Systems, University of Information Technology, (e-mail: 21522005@gm.uit.edu.vn)

²Faculty of Information Systems, University of Information Technology, (e-mail: 21522179@gm.uit.edu.vn)

³Faculty 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 an 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 pharmaceutical industry in Vietnam is experiencing rapid growth, with companies like Hataphar, Central Pharmaceutical No3, and American Vietnamese Biotech emerging as key players. Accurately forecasting stock prices in this sector is challenging due to complex factors such as long product development cycles, regulatory uncertainties, and scientific breakthroughs.

Traditional forecasting models often struggle to capture the intricate patterns in pharmaceutical stock price movements. This study aims to address these limitations by leveraging an ensemble of statistical methods (Fast Fourier Transform, Linear Regression with Calendar/Fourier terms, ARIMA) and machine learning models (LightGBM, RNNs, GRUs, LSTMs) to forecast stock prices of three major Vietnamese pharmaceutical firms.

By combining diverse approaches, this research seeks to uncover insights into factors influencing price dynamics in the pharmaceutical sector. The findings can contribute to

financial forecasting knowledge, guide investment strategies, and support industry growth in Vietnam's pharmaceutical market. This comprehensive analysis aims to bridge the gap between traditional financial modeling and cutting-edge machine learning approaches, with potential applications in other emerging markets and sectors.

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.

Ke et al. (2017) [10] 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 |

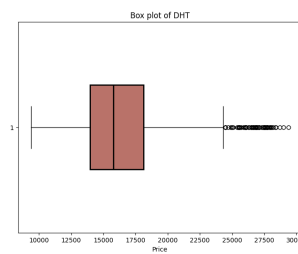


FIGURE 1. DHT stock price's boxplot

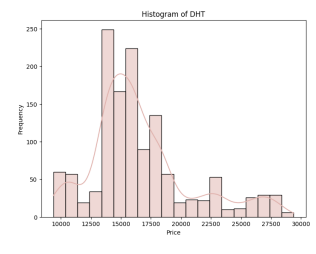


FIGURE 2. DHT stock price's histogram

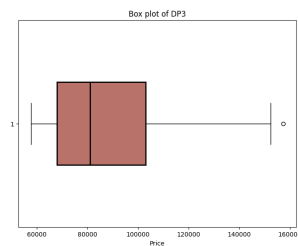


FIGURE 3. DP3 stock price's boxplot

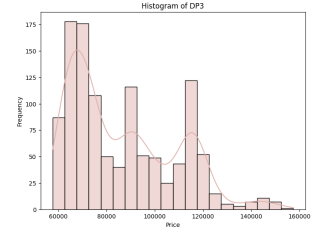


FIGURE 4. DP3 stock price's histogram

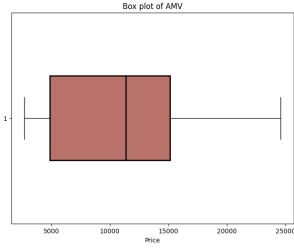


FIGURE 5. AMV stock price's boxplot

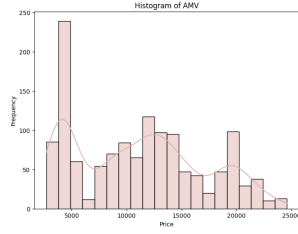


FIGURE 6. AMV stock price's histogram

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:

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

Where:

- Y is the dependent variable (Target Variable).
- X_1, X_2, \dots, X_k are the independent (explanatory) variables.
- β_0 is the intercept term.
- β_1, \dots, β_k are the regression coefficients for the independent variables.
- ε is the error term.

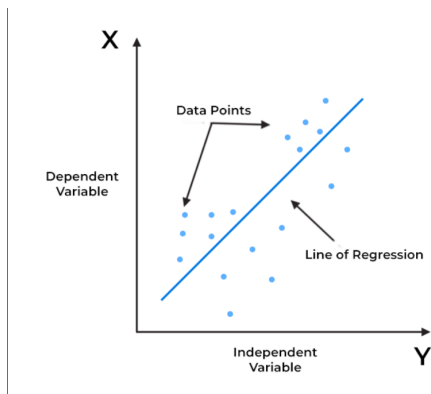


FIGURE 7. LN model's

B. FAST FOURIER TRANSFORM

The Fast Fourier Transform (FFT) is a powerful method for analyzing and forecasting time series data, particularly in financial markets. It allows for the decomposition of a time series into its constituent frequency components in $O(n \log n)$ time, significantly faster than traditional methods.

In the context of stock price prediction, let $S(t)$ be a time series of stock prices:

$$S(t) = s_0 + s_1 t + s_2 t^2 + \dots + s_{n-1} t^{n-1} \quad (1)$$

Where:

- $S(t)$ represents the stock price at time t .
- s_0, s_1, \dots, s_{n-1} are the coefficients representing different frequency components.
- t is the time variable.
- n is the number of data points, typically chosen as a power of 2 for computational efficiency.

The FFT decomposes this time series into its frequency components:

$$F(\omega) = \text{FFT}(S(t)) \quad (2)$$

Where $F(\omega)$ represents the frequency domain representation of the time series.

This decomposition allows us to:

- 1) Identify underlying periodic patterns in the stock price movements.
- 2) Filter out noise and focus on significant frequency components.
- 3) Extrapolate future values based on the identified frequency patterns.

The inverse FFT can then be used to transform the frequency domain representation back into the time domain, providing a forecast:

$$S'(t) = \text{IFFT}(F(\omega)) \quad (3)$$

Where $S'(t)$ is the forecasted time series.

By applying FFT to stock price data, we can uncover cyclical patterns that may not be immediately apparent in the raw time series. This information can be crucial for predicting future price movements, especially when combined with other forecasting techniques.

The FFT's ability to efficiently process large datasets makes it particularly suitable for analyzing high-frequency trading data or long historical price series. Moreover, it can be integrated into more complex forecasting models, providing frequency-domain features that complement traditional time-domain analysis.

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 Y_t depends only on its own lags. That is, Y_t is a function of the 'lags of Y_t '.

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

Where:

- Y_t : The value of the time series at time t .
- a : A constant term.
- B_1, B_2, \dots, 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.
- ε_t : A white noise error term at time t .

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

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

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 (LSTM) network is a type of recurrent neural network (RNN) capable of learning long-term dependencies. The LSTM model consists of several components, including gates that control the flow of information through the network. The LSTM model can be described as follows:

- Let \mathbf{x}_t be the input vector at time step t .
- Let \mathbf{h}_t be the hidden state vector at time step t .
- Let \mathbf{c}_t be the cell state vector at time step t .
- Let \mathbf{W} , \mathbf{U} , and \mathbf{b} be the weight matrices and bias vectors.

The LSTM model is defined by the following equations:

1. Forget Gate:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f)$$

2. Input Gate:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i)$$

3. Candidate Memory Cell:

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c)$$

4. Cell State:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

5. Output Gate:

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o)$$

6. Hidden State:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

Where:

- σ denotes the sigmoid activation function.
- \tanh denotes the hyperbolic tangent activation function.
- \odot denotes element-wise multiplication.
- $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_c, \mathbf{W}_o$ are the input weight matrices.
- $\mathbf{U}_f, \mathbf{U}_i, \mathbf{U}_c, \mathbf{U}_o$ are the recurrent weight matrices.
- $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_c, \mathbf{b}_o$ are the bias vectors.
- \mathbf{f}_t is the forget gate vector.
- \mathbf{i}_t is the input gate vector.
- $\tilde{\mathbf{c}}_t$ is the candidate memory cell vector.
- \mathbf{c}_t is the cell state vector.
- \mathbf{o}_t is the output gate vector.
- \mathbf{h}_t is the hidden state vector.

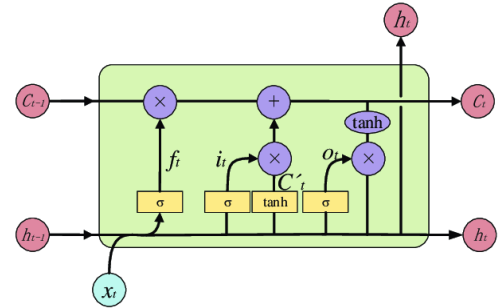


FIGURE 8. LSTM model's

E. LINEAR REGRESSION COMBINED WITH CALENDARFOURIER AND DETERMINISTICPROCESS

The model can be expressed as:

$$P(t) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n + f(t) + g(t) + \varepsilon \quad (4)$$

Where:

- $P(t)$ is the stock price at time t .
- β_0 is the intercept term.
- $\beta_1, \beta_2, \dots, \beta_n$ are coefficients for external factors X_1, X_2, \dots, X_n (e.g., market indices, economic indicators).
- $f(t)$ is the CalendarFourier component, capturing cyclical patterns.
- $g(t)$ is the DeterministicProcess component, representing the long-term trend.

- ε is the error term.

The CalendarFourier component $f(t)$ models recurring patterns in stock prices:

$$f(t) = \alpha_1 \sin(\omega t) + \beta_1 \cos(\omega t) + \alpha_2 \sin(2\omega t) + \beta_2 \cos(2\omega t) + \dots + \alpha_k \sin(k\omega t) + \beta_k \cos(k\omega t) \quad (5)$$

Where:

- ω is the frequency (e.g., $2\pi/252$ for annual cycles in trading days).
- k is the number of Fourier terms.
- $\alpha_1, \beta_1, \dots, \alpha_k, \beta_k$ are coefficients to be estimated.

This component can capture cyclical patterns such as quarterly earnings reports, annual industry events, or seasonal demand fluctuations in the pharmaceutical sector.

The DeterministicProcess component $g(t)$ models the long-term trend in stock prices:

$$g(t) = \gamma_1 t + \gamma_2 t^2 + \gamma_3 t^3 + \dots \quad (\text{polynomial trend}) \quad (6)$$

$$\text{or } g(t) = \gamma_1 + \gamma_2 \log(t) \quad (\text{logarithmic trend}) \quad (7)$$

$$\text{or } g(t) = \gamma_1 \exp(\gamma_2 t) \quad (\text{exponential trend}) \quad (8)$$

Where $\gamma_1, \gamma_2, \gamma_3, \dots$ are coefficients to be estimated.

This component can capture long-term growth trends in pharmaceutical stocks, which may be influenced by factors such as R&D pipelines, market expansion, or industry-wide growth.

The model parameters are estimated using Ordinary Least Squares (OLS) or other optimization techniques, minimizing the sum of squared residuals between the predicted and actual stock prices.

By combining these components, the model can effectively capture both the cyclical patterns and long-term trends in pharmaceutical stock prices, potentially leading to more accurate forecasts compared to simpler models. This approach allows for the incorporation of industry-specific knowledge, such as known cycle periods or expected growth patterns, into the forecasting process.

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, $\text{Base_tree}(X)$ is the prediction of the base model, $\text{Tree}_1(X)$, $\text{Tree}_2(X)$, and $\text{Tree}_3(X)$ 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.

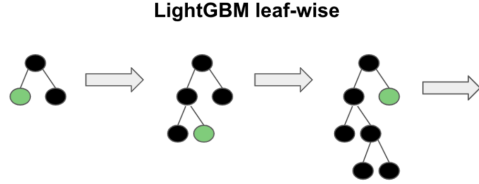


FIGURE 9. LightGBM model's

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 :

$$|X| = x_1, x_2, \dots, x_n \quad (9)$$

Where:

- $|X|$ is the length of the input sequence X .
- x_1, x_2, \dots, 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:

$$h_1 = f(x_1, h_0) \quad h_2 = f(x_2, h_1) \quad \dots \quad h_n = f(x_n, h_{n-1}) \quad (10)$$

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:

$$y_t = g(h_t, x_t) \quad (11)$$

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 | 7781.26 | 49.62 | 0.19 |
| | 8:2 | 9119.01 | 67.34 | 0.27 |
| | 9:1 | 6907.31 | 51.82 | 0.17 |
| RNN | 7:3 | 1276.19 | 0.3207 | 0.0801 |
| | 8:2 | 135.78 | 0.0264 | 0.0011 |
| | 9:1 | 264.547 | 0.0665 | 0.0054 |
| GRU | 7:3 | | | |
| | 8:2 | | | |
| | 9:1 | | | |
| ARIMA | 7:3 | 558.44 | 11.50% | 0.02 |
| | 8:2 | 801.64 | 18.26% | 0.04 |
| | 9:1 | 290.25 | 5.51% | 0.00 |
| FFT | 7:3 | 0.0829 | 4151 | 0.0054 |
| | 8:2 | 0.05950 | 0.9657 | 0.003 |
| | 9:1 | 0.0259 | 0.5279 | 0.0006 |
| 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 | 1861.67 | 1.78% | 0.00068 |
| | 8:2 | 522.86 | 2.41% | 0.00135 |
| | 9:1 | 298.001 | 1.74% | 0.00046 |
| LightGBM | 7:3 | 0.11 | 215.22 | 0.01 |
| | 8:2 | 0.05 | 128.70 | 0.002 |
| | 9:1 | 0.05 | 188.43 | 0.002 |

TABLE 3. AMV Dataset's Evaluation

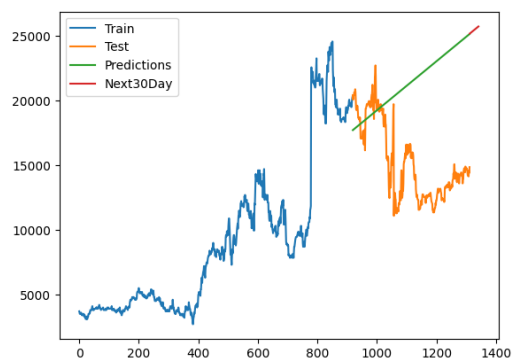


FIGURE 10. Linear model's result with 7:3 splitting proportion



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

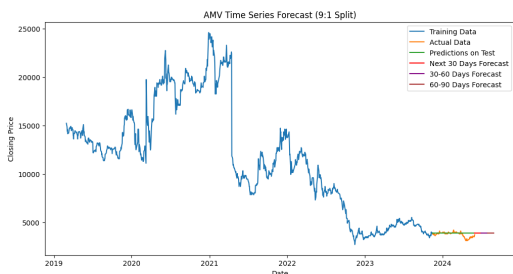


FIGURE 12. Arima model's result with 8:2 splitting proportion

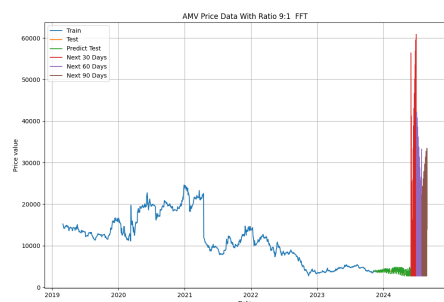


FIGURE 13. FFT model's result with 9:1 splitting proportion

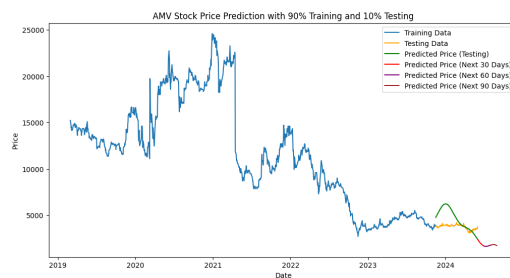


FIGURE 14. LN-CF-DP model's result with 9:1 splitting proportion

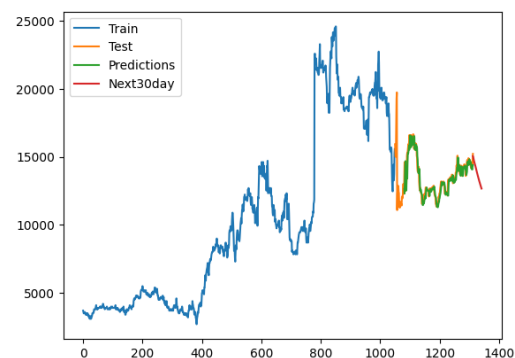


FIGURE 15. LSTM model's result with 8:2 splitting proportion

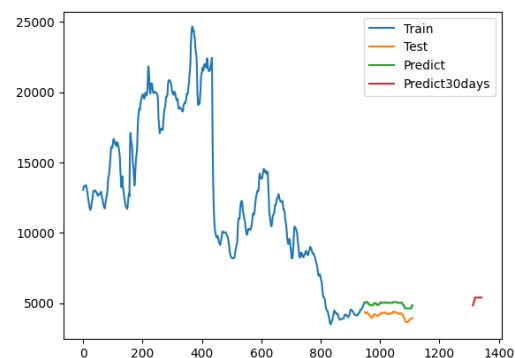


FIGURE 16. LightGBM model's result with 8:2 splitting proportion

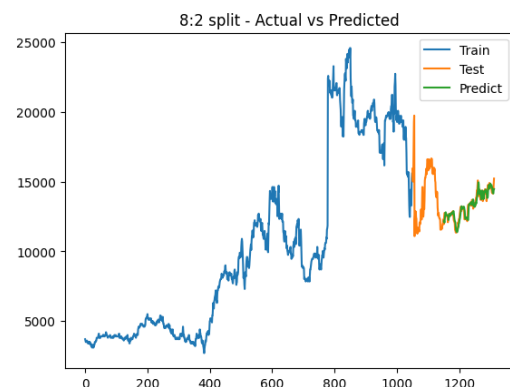


FIGURE 17. GRU model's result with 8:2 splitting proportion

C. DP3 DATASET

| DP3 Dataset's Evaluation | | | | |
|--------------------------|------------------|----------|----------|---------|
| Model | Training:Testing | RMSE | MAPE (%) | MSLE |
| LN | 7:3 | 7781.26 | 49.62 | 0.19 |
| | 8:2 | 9119.01 | 67.34 | 0.27 |
| | 9:1 | 6907.31 | 51.82 | 0.17 |
| RNN | 7:3 | 1675.09 | 0.0412 | 0.0037 |
| | 8:2 | 2503.94 | 0.0700 | 0.0081 |
| | 9:1 | 1499.11 | 0.0275 | 0.0021 |
| GRU | 7:3 | | | |
| | 8:2 | | | |
| | 9:1 | | | |
| ARIMA | 7:3 | 27750.14 | 39.01% | 0.36 |
| | 8:2 | 7936.36 | 10.027% | 0.01 |
| | 9:1 | 6394.82 | 7.79% | 0.01 |
| FFT | 7:3 | 0.2076 | 61.27% | 0.0232 |
| | 8:2 | 0.1987 | 59.18% | 0.0224 |
| | 9:1 | 0.1352 | 37.86% | 0.0103 |
| 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 | 1861.67 | 1.78% | 0.00068 |
| | 8:2 | 522.86 | 2.41% | 0.00135 |
| | 9:1 | 298.001 | 1.74% | 0.00046 |
| LightGBM | 7:3 | 0.11 | 215.22 | 0.01 |
| | 8:2 | 0.05 | 128.70 | 0.002 |
| | 9:1 | 0.05 | 188.43 | 0.002 |

TABLE 4. DP3 Dataset's Evaluation

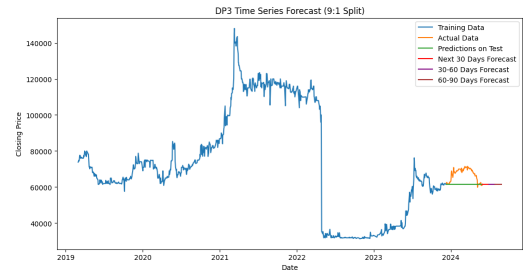


FIGURE 20. ARIMA model's result with 8:2 splitting proportion

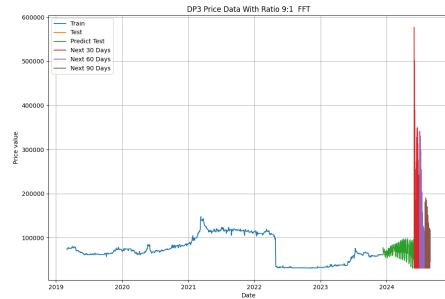


FIGURE 21. FFT model's result with 9:1 splitting proportion

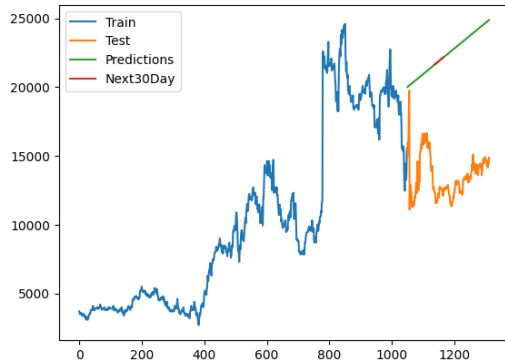


FIGURE 18. Linear model's result with 8:2 splitting proportion

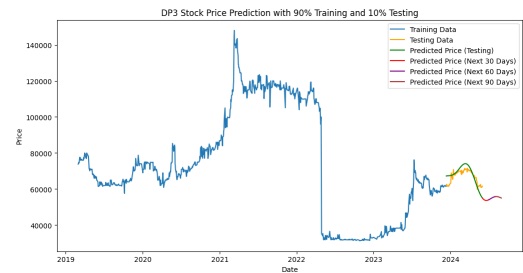


FIGURE 22. LN-CF-DP model's result with 9:1 splitting proportion

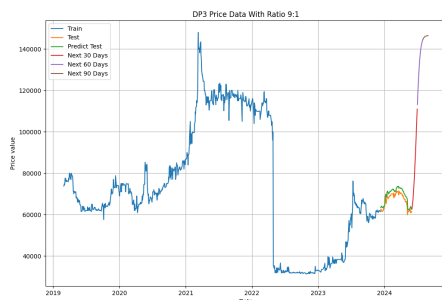


FIGURE 19. RNN model's result with 9:1 splitting proportion

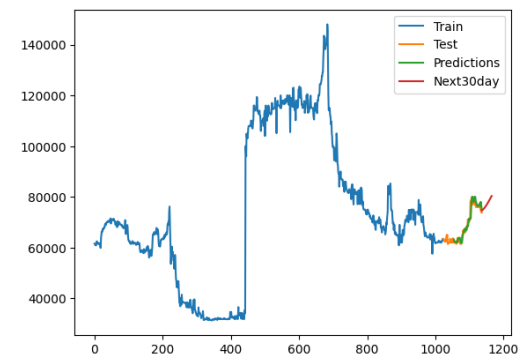


FIGURE 23. LSTM model's result with 9:1 splitting proportion

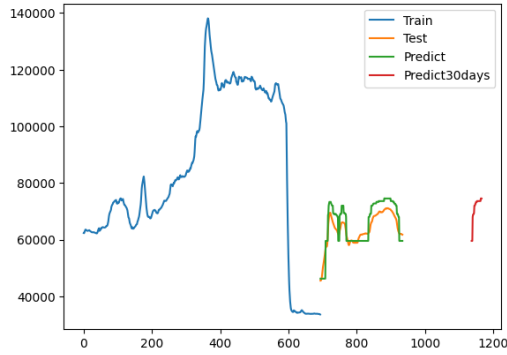


FIGURE 24. LightGBM model's result with 7:3 splitting proportion

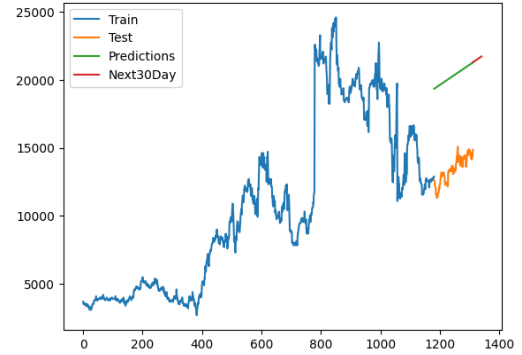


FIGURE 26. Linear model's result with 9:1 splitting proportion

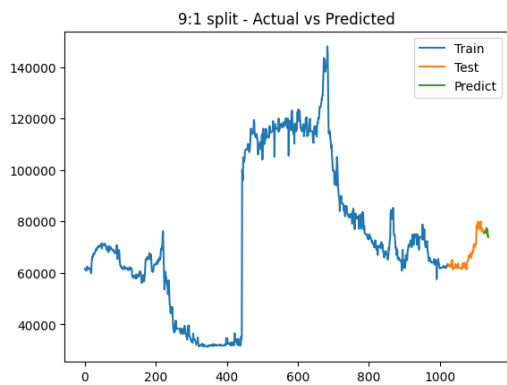


FIGURE 25. GRU model's result with 9:1 splitting proportion

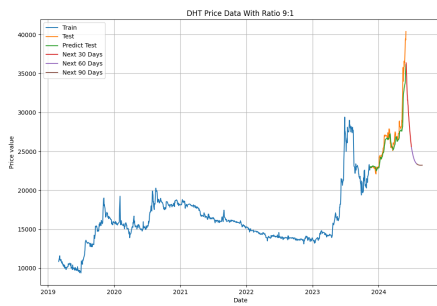


FIGURE 27. RNN model's result with 9:1 splitting proportion

D. DHT DATASET

| DHT Dataset's Evaluation | | | | |
|--------------------------|------------------|---------|----------|---------|
| Model | Training:Testing | RMSE | MAPE (%) | MSLE |
| LN | 7:3 | 7781.26 | 49.62 | 0.19 |
| | 8:2 | 9119.01 | 67.34 | 0.27 |
| | 9:1 | 6907.31 | 51.82 | 0.17 |
| RNN | 7:3 | 3065.26 | 0.0401 | 0.0026 |
| | 8:2 | 2855.37 | 0.0276 | 0.0018 |
| | 9:1 | 2327.70 | 0.0319 | 0.0011 |
| GRU | 7:3 | | | |
| | 8:2 | | | |
| | 9:1 | | | |
| ARIMA | 7:3 | 9640.50 | 29.78% | 0.24 |
| | 8:2 | 9305.45 | 31.50% | 0.18 |
| | 9:1 | 5570.55 | 12.58% | 0.04 |
| FFT | 7:3 | 0.5714 | 63.88% | 0.0871 |
| | 8:2 | 0.5188 | 57.93% | 0.0862 |
| | 9:1 | 0.3351 | 34.96% | 0.3354 |
| 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 | 1861.67 | 1.78% | 0.00068 |
| | 8:2 | 522.86 | 2.41% | 0.00135 |
| | 9:1 | 298.001 | 1.74% | 0.00046 |
| LightGBM | 7:3 | 0.11 | 215.22 | 0.01 |
| | 8:2 | 0.05 | 128.70 | 0.002 |
| | 9:1 | 0.05 | 188.43 | 0.002 |

TABLE 5. DHT Dataset's Evaluation

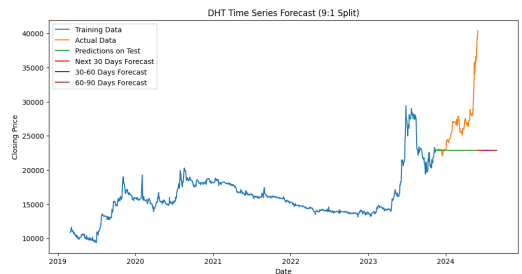


FIGURE 28. Arima model's result with 8:2 splitting proportion

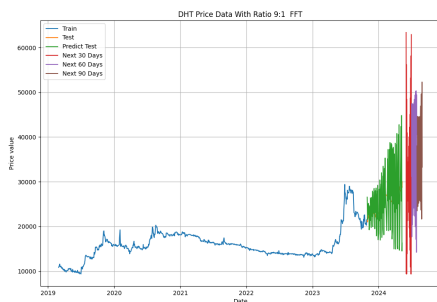


FIGURE 29. FFT model's result with 9:1 splitting proportion

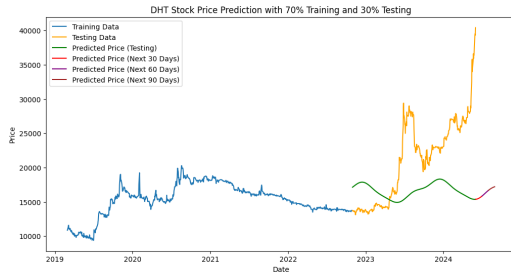


FIGURE 30. LN-CF-DP model's result with 7:3 splitting proportion

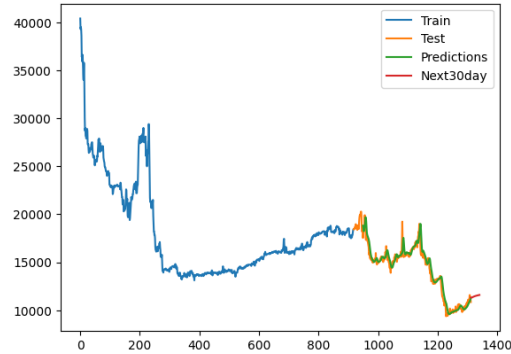


FIGURE 31. LSTM model's result with 7:3 splitting proportion

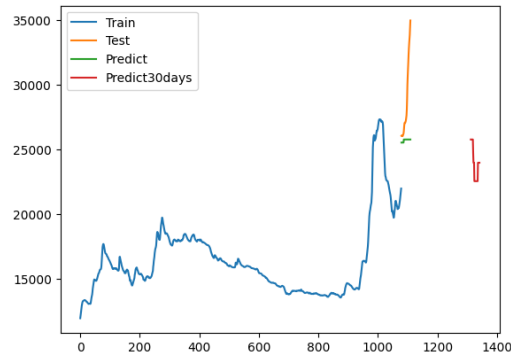


FIGURE 32. LightGBM model's result with 8:2 splitting proportion

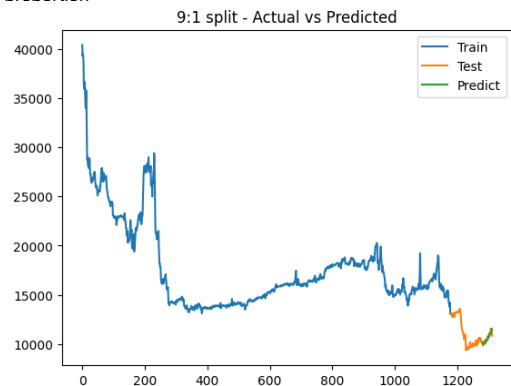


FIGURE 33. GRU model's result with 8:2 splitting proportion

ACKNOWLEDGMENT

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 meticulous

mentoring and corrections, especially from Dr. Thuan, significantly shaped the direction and quality of this study. Mr. Nhut's dedicated guidance has been instrumental in our research journey.

This research would not have been possible without their support and contributions. We extend our heartfelt thanks to everyone involved for their invaluable assistance, encouragement, and belief in our research. We deeply appreciate their continuous support and guidance.

VI. CONCLUSION

A. SUMMARY

This research explored various methods for predicting stock prices, ranging from traditional statistical models to advanced machine learning algorithms. The models tested include Linear Regression (LR), ARIMA, GRU, RNN, LSTM, FFT, LightGBM, and LN-CF-DP. Among these, LSTM and GRU stand out as the most effective models for stock price prediction.

Each model provides unique insights: deep learning models effectively capture non-linear patterns, traditional methods offer interpretable results, while LightGBM efficiently handles multiple data features.

Although no method can perfectly predict stock prices due to market complexities, this study provides a comprehensive toolkit for investors in the pharmaceutical sector.

B. FUTURE CONSIDERATIONS

To advance this research in the future, focus should be placed on:

- 1) Optimizing existing models: Improve the accuracy of LightGBM, LSTM, and GRU through hyperparameter tuning and advanced data processing techniques.
- 2) Incorporating sentiment analysis: Integrate data from pharmaceutical industry news and social media to capture the impact of market sentiment.
- 3) Expanding data scope: Add macroeconomic indicators and pharmaceutical industry-specific data to enhance predictive capabilities.
- 4) Applying ensemble learning techniques: Explore methods such as Bagging or Boosting to combine the strengths of multiple models.
- 5) Researching new models: Monitor and experiment with the latest forecasting algorithms in machine learning and quantitative finance.

By continuously exploring and incorporating new features, data sources, and modeling techniques, we can strive to continuously optimize forecasting models and enhance the ability to predict stock prices with greater accuracy and reliability.

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.
- [7] Kostadinov, S. (2017, December 16). Understanding GRU networks. Towards Data Science. <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>.
- [8] Fuqua School of Business. (n.d.). Introduction to ARIMA models. Duke.edu. Retrieved June 20, 2024, from <https://people.duke.edu/rnau/411arim.htm>
- [9] Nau, R. (2014). The mathematical structure of arima models. *Duke University Online Article*, 1(1), 1-8.
- [10] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... and Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.
- [11] CP-Algorithms. (n.d.). Fast Fourier Transform (FFT). Retrieved June 21, 2024, from <https://cp-algorithms.com/algebra/fft.html>