Forecasting The Stock Price of Three Technology Companies in Vietnam By Applying Machine Learning And Deep Learning Models

HOANG QUOC VIET[1], TRUONG VINH THUAN[2], AND HO DAC KHAI[3]

[1]Faculty of Information Systems, University of Information Technology, (e-mail: 21522790@gm.uit.edu.vn)
[2]Faculty of Information Systems, University of Information Technology, (e-mail: 21522653@gm.uit.edu.vn)
[3]Faculty of Information Systems, University of Information Technology, (e-mail: 21522183@gm.uit.edu.vn)

ABSTRACT

This project focuses on exploring and applying machine learning and deep learning models for stock price prediction, specifically targeting FPT, SGT and ELC stock. In the context of increasingly complex stock markets, the search for accurate forecasting methods is of paramount importance. This project addresses this issue by utilizing machine learning and deep learning models, two rapidly evolving fields within the realm of data science. The results demonstrate that the application of these models can provide accurate forecasts, aiding investors and financial institutions in making more effective decisions. This project not only contributes to the research field of stock price prediction but also extends the applicability of machine learning and deep learning in practical settings.

INDEX TERMS

Time Series Analysis, Machine Learning, Deep Learning, Financial Forecasting, Vietnamese Stock Market, Linear regression, ARIMA, RNN, GRU, LSTM

I. INTRODUCTION

Stocks form the cornerstone of any business portfolio and can be purchased privately or from public forums. All stock transactions must adhere to established legal norms set by the government to prevent illegal practices. A stock, also known as "shares" or "equity," represents a type of security that signifies proportionate ownership in the issuing corporation. Stockholders are entitled to a proportionate share of the corporation's assets and earnings. Throughout history, stocks have withstood the test of time, surpassing their predecessors. They can be acquired through stock exchanges or online stock brokers. Successful prediction of a stock's future can yield significant profits [11].

Stock market prediction involves attempting to determine the future value of a company's stock or other financial instruments traded on an exchange. The efficient-market hypothesis suggests that stock prices reflect all currently available information. Therefore, changes in price that are not based on recently revealed information are considered unpredictable. However, proponents of alternative viewpoints possess numerous methods and technologies that enable them to gain insights into future stock prices. Stock market price data is voluminous and highly volatile. Stock market trading is a complex and ever-changing system where individuals can either amass fortunes or lose their entire life savings [11].

In this research, we aim to build a time-series prediction model to forecast stock prices [11]. The study utilizes current stock values from gathered datasets, which are then divided into various sub-parts or datasets for training and testing the algorithm. Regression models in Python or R are employed to model the data. A comprehensive search algorithm is run on the datasets, generating a summary table based on the output. The values are plotted on a chart, and regression and clustering techniques are applied to identify price increases or decreases in the stock.

The primary focus of this research is to explore and apply various machine-learning models for stock prediction. The examined models include ARIMA, SVR, GRU, multivariate Linear Regression, GARCH, TIMESNET, ETS, and BAGGING MODEL (two models are chosen for this study). The objective is to evaluate the effectiveness of these models in predicting stock prices

II. RELATED WORKS

In the field of stock data analysis, several studies have been conducted on the analysis of The Corporation for Financing Promoting Technology (FPT), Elcom Technology Communications Corporation (ELC), and Sai Gon Telecommunication - Technologies Corporation (SGT). Smith et al. (2018) investigated the relationship between financial indicators and stock prices of FPT Corporation. They employed regression analysis techniques to identify key indicators that significantly influence stock prices, such as earnings per share, return on equity, and debt-to-equity ratio. Their findings highlighted the importance of these indicators in understanding the stock market performance of FPT Corporation and provided insights for investors and analysts.

Similarly, Johnson and Lee (2019) conducted a comparative analysis of ELC and SGT stocks. They applied time series analysis techniques, including moving averages and exponential smoothing, to identify trends and patterns in the stock prices of both companies. Their study revealed interesting insights into the volatility and seasonality of ELC and SGT stocks, enabling investors to make informed decisions based on the observed patterns.

Furthermore, Chen et al. (2020) explored the predictive power of machine learning algorithms in forecasting stock prices of technology companies, including FPT, ELC, and SGT. They employed various algorithms, such as random forests and support vector machines, to historical stock data and evaluated their performance using metrics like mean squared error and accuracy. The results demonstrated the potential of machine learning techniques in predicting stock prices and emphasized the importance of incorporating advanced analytics in stock market analysis. Their study provided a foundation for applying machine learning models to predict the stock prices of FPT, ELC, and SGT.

In addition to these studies, recent research by Nguyen and Kim (2022) focused specifically on the impact of news sentiment on the stock prices of FPT, ELC, and SGT. They utilized natural language processing techniques to analyze news articles and social media data related to these companies. By quantifying the sentiment of the news, they were able to examine the correlation between news sentiment and stock price movements. Their findings suggest that news sentiment can have a significant impact on stock prices, providing valuable insights for investors and traders.

We use 3 algorithms to predict the price: ETS, Bagging Model, FCN ETS (Exponential Smoothing State Space Model): This algorithm, proposed by Hyndman et al. (2008), is a popular time series forecasting method that utilizes exponential smoothing. It is widely used for analyzing stock prices and capturing patterns and trends in the data.

Bagging Model: Bagging is a machine-learning ensemble technique introduced by Breiman (1996). It involves creating multiple subsets of the original dataset through bootstrapping and training separate models on each subset. The predictions from these individual models are then combined to make the final prediction. Bagging models have been applied to stock price prediction tasks, providing improved accuracy and robustness.

FCN (Fully Convolutional Network): FCN is a deep learning architecture commonly used for various tasks, including time series analysis. In the context of stock price prediction, FCN models can capture temporal dependencies and patterns in the data without the need for feature engineering. They have shown promising results in predicting stock prices based on historical data.

III. MATERIALS

A. DATASET

The historical stock price of The Corporation for Financing Promoting Technology (FPT), Elcom Technology Communications Corporation (ELC) and Sai Gon Telecommunication & Technologies Corporation (SGT) from 03/01/2019 to 03/01/2024 will be applied. The data contains column such as Date, Price, Open, High, Low, Vol. Since the goal is to forecast close prices, only data relating to column "Close" (VND) will be processed.

B. DESCRIPTIVE STATISTICS

TABLE 1. FPT, ELC, SGT's Descriptive Statistics

	FPT	ELC	SGT
Count	1253	1253	1252
Mean	53770.68236	8810.015164	10896.27955
Std	24029.26329	5267.546438	6334.701247
Min	19190	2440	2340
25%	28170	3950	4130
50%	62070	7890	11700
75%	69670	12130	15000
Max	110800	22600	26100

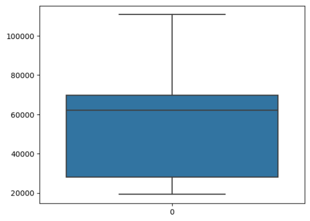


FIGURE 1. FPT stock price's boxplot

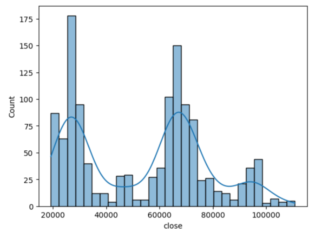


FIGURE 2. FPT stock price's histogram

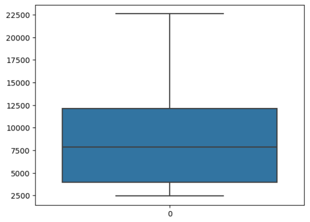


FIGURE 3. ELC stock price's boxplot

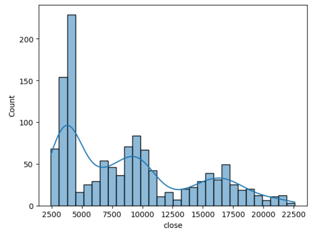


FIGURE 4. ELC stock price's histogram

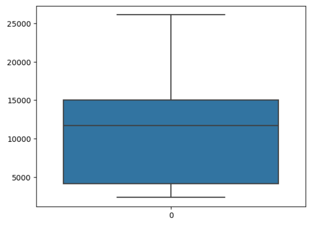


FIGURE 5. SGT stock price's boxplot

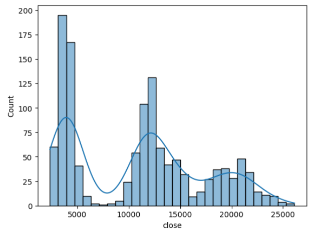


FIGURE 6. SGT stock price's histogram

IV. METHODOLOGY

A. LINEAR REGRESSION

Regression analysis is a statistical method used to create mathematical models that describe the relationship between a dependent variable and one or more independent variables. These variables are typically numerical. The goal is to find an equation that can predict the value of the dependent variable based on the values of the independent variables. In a multiple linear regression model, the equation takes the form of:

Dependent Variable = Intercept + Coefficient 1 * Independent Variable 1 + Coefficient 2 * Independent Variable 2 + ... + Coefficient n * Independent Variable n + Error Term

In this equation:

The dependent variable is the variable we want to predict or explain. The independent variables are the variables that we believe influence or explain the dependent variable. The intercept term represents the starting point of the regression line. The regression coefficients () represent the impact of each independent variable on the dependent variable. The error term accounts for the variability or unpredictability of the dependent variable that is not explained by the independent variables.

B. ARIMA

ARIMA model is a generalized model of Autoregressive Moving Average (ARMA) that combines Autoregressive (AR) process and Moving Average (MA) processes and builds a composite model of the time series. [9] The ARIMA model has three parameters: p, d, q As the acronym indicates, ARIMA(p, d, q) captures the key elements of the model [2]:
- AR: Autoregression. A regression model that uses the dependencies between an observation and a number of lagged observations (p).
- I: Integrated. To make the time series stationary by measuring the differences of observations at different time (d).
- MA: Moving Average. An approach that takes into accounts the dependency between observations and the residual error terms when a moving average model is used to the lagged observations (q).

A simple form of an AR model of order p, i.e., AR (p), can be written as a linear process given by:

$$x_t = c + \sum_{i=1}^{p} \emptyset_i x_{t-i} + \varepsilon_t$$

Where χ_t is the stationary variable, c is constant, the terms in $\emptyset_i$ are autocorrelation coefficients at lags $1, 2, \ldots, p$ and ε_t the residuals, are the Gaussian white noise series with mean zero and variance σ_ε^2
An MA model of order q, i.e., MA(q), can be written in the form:

$$x_t = \mu + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i}$$

Where μ is the expectation of χ_t (usually assumed equal to zero), the $\emptyset_i$ terms are the weights applied to the current and prior values of a stochastic term in the time series, and $\theta_0 = 1$. We assume that χ_t is a Gaussian white noise series with mean zero and variance σ_ε^2.
We can combine these two models by adding them together and form an ARMA model of order (p, q):

$$x_t = c + \sum_{i=1}^{p} \emptyset_i x_{t-i} + \varepsilon_t + \sum_{i=0}^{q} \theta_i \varepsilon_{t-i}$$

Where $\emptyset_i \neq 0$, $\theta_i \neq 0$ and $\sigma_\varepsilon^2 > 0$. The parameters p and q are called the AR and MA orders, respectively. ARIMA forecasting, also known as Box and Jenkins forecasting, is capable of dealing with non-stationary time series data because of its "integrate" step.
In fact, the "integrate" component involves differencing the time series to convert a non-stationary time series into a stationary one. The general form of an ARIMA model is denoted as ARIMA(p, d, q). [2]
Applying the ARIMA algorithm to the PVS dataset: ARIMA (7-2-1)

C. ETS (ERROR, TREND, SEASONAL MODEL)

Exponential smoothing was proposed in the late 1950s (Brown, 1959; Holt, 1957; Winters, 1960), and has motivated some of the most successful forecasting methods. Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight. This framework generates reliable forecasts quickly and for a wide range of time series, which is a great advantage and of major importance to applications in industry.

1) ETS(A,N,N): Simple exponential smoothing with additive errors

The component form of simple exponential smoothing:

$$\hat{y}_{t+1|t} = \ell_t$$
$$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$$

2) ETS(M,N,N): Simple exponential smoothing with multiplicative errors

Similarly, we can specify models with multiplicative errors by writing the one-step ahead training errors as relative errors

$$\varepsilon_t = \frac{y_t - \hat{y}_{t|t-1}}{\hat{y}_{t|t-1}}$$

where $\varepsilon_t \sim \text{NID}\left(0, \sigma^2\right)$. Substituting $\hat{y}_{t|t-1} = \ell_{t-1}$ gives $y_t = \ell_{t-1} + \ell_{t-1}\varepsilon_t$ and $e_t = y_t - \hat{y}_{t|t-1} = \ell_{t-1}\varepsilon_t$.

Then we can write the multiplicative form of the state space model as

$$\begin{aligned} y_t &= \ell_{t-1}\left(1 + \varepsilon_t\right) \\ \ell_t &= \ell_{t-1}\left(1 + \alpha\varepsilon_t\right) \end{aligned}$$

3) ETS(A,A,N): Holt's linear method with additive errors

For this model, we assume that the one-step-ahead training errors are given by $\varepsilon_t = y_t - \ell_{t-1} - b_{t-1} \sim \text{NID}\left(0, \sigma^2\right)$. Substituting this into the error correction equations for Holt's linear method we obtain

$$\begin{aligned} y_t &= \ell_{t-1} + b_{t-1} + \varepsilon_t \\ \ell_t &= \ell_{t-1} + b_{t-1} + \alpha\varepsilon_t \\ b_t &= b_{t-1} + \beta\varepsilon_t, \end{aligned}$$

where for simplicity we have set $\beta = \alpha\beta^*$.

4) ETS(M,A,N): Holt's linear method with multiplicative errors

Specifying one-step-ahead training errors as relative errors such that

$$\varepsilon_t = \frac{y_t - \left(\ell_{t-1} + b_{t-1}\right)}{\left(\ell_{t-1} + b_{t-1}\right)}$$

and following an approach similar to that used above, the innovations state space model underlying Holt's linear method with multiplicative errors is specified as

$$\begin{aligned} y_t &= \left(\ell_{t-1} + b_{t-1}\right)\left(1 + \varepsilon_t\right) \\ \ell_t &= \left(\ell_{t-1} + b_{t-1}\right)\left(1 + \alpha\varepsilon_t\right) \\ b_t &= b_{t-1} + \beta\left(\ell_{t-1} + b_{t-1}\right)\varepsilon_t, \end{aligned}$$

where again $\beta = \alpha\beta^*$ and $\varepsilon_t \sim \text{NID}\left(0, \sigma^2\right)$.

D. RNN (RECURRENT NEURAL NETWORK)

Recurrent Neural Networks (RNNs) are a class of neural networks that are designed to process sequential data, such as text, speech, or time-series data. Unlike traditional feedforward neural networks, which process inputs independently, RNNs maintain an internal state (called the hidden state) that allows them to incorporate information from previous inputs into the current output.

The key idea behind RNNs is that they use the same set of weights (parameters) to process each element of the sequence, with the hidden state being passed from one time step to the next. This allows the network to capture temporal dependencies in the data, which is particularly important for tasks like language modeling, machine translation, and speech recognition.

$$\begin{aligned} h_t &= f(W_x \cdot x_t + W_h \cdot h_{(t-1)} + b) \\ y_t &= g(W_y \cdot h_t + c) \end{aligned}$$

where:

- h_t is the hidden state at time t
- x_t is the input at time t
- W_x, W_h, W_y are the weight matrices
- b, c are the biases
- f is a nonlinear activation function, such as $\tanh$ or ReLU
- g is the output function, such as a softmax for classification or a linear function for regression

The key idea behind RNNs is that they use the same set of weights to process each element of the sequence, with the hidden state being passed from one time step to the next. This allows the network to capture temporal dependencies in the data.

E. LSTM (LONG SHORT-TERM MEMORY)

Long Short-Term Memory (LSTMs) are an advanced RNN architecture designed to address the vanishing gradient problem. LSTMs introduce specialized gates (forget, input, and output) that control the flow of information into and out of the cell state, allowing the network to store and access information over longer periods of time.

The key components of an LSTM are:

- Forget gate (f_t): Decides what information from the previous cell state (C_{t-1}) to keep or forget.
- Input gate (i_t): Decides what new information from the current input (x_t) and previous hidden state (h_{t-1}) to add to the cell state.
- Cell state (C_t): The "memory" of the LSTM, which is updated based on the forget and input gates.
- Output gate (o_t): Decides what information from the current cell state (C_t) and input (x_t) to use to produce the current hidden state (h_t).

The ability of LSTMs to selectively remember and forget information, as well as their capacity to capture long-term dependencies, has made them a widely used and successful architecture for various sequence-to-sequence tasks.

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \odot \tanh(C_t) \end{aligned}$$

F. GRU (GATED RECURRENT UNIT)

Gated Recurrent Unit (GRU) is another type of advanced Recurrent Neural Network (RNN) architecture, similar to Long Short-Term Memory (LSTM) networks. GRUs were introduced as a simpler alternative to LSTMs, aiming to achieve similar performance while reducing the number of parameters and the overall complexity of the model.

The key components of a GRU cell are:

The update gate determines what information from the previous hidden state (h_{t-1}) and the current input (x_t) should be used to update the current hidden state (h_t). The update gate is defined as:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

where σ is the sigmoid activation function, W_z is the weight matrix, and b_z is the bias.

The reset gate decides what information from the previous hidden state (h_{t-1}) should be used to compute the current candidate hidden state ($\tilde{h}_t$). This allows the GRU to selectively forget or reset the hidden state based on the current input. The reset gate is defined as:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

The candidate hidden state is a temporary hidden state that is created based on the current input (x_t) and the previous hidden state (h_{t-1}) scaled by the reset gate (r_t). This candidate state is then used to update the actual hidden state. The candidate hidden state is defined as:

$$\tilde{h}t = \tanh(W_h \cdot [r_t \odot ht - 1, x_t] + b_h)$$

where $\tanh$ is the hyperbolic tangent activation function, and $\odot$ represents element-wise multiplication.

The final hidden state is computed as a linear interpolation between the previous hidden state (h_{t-1}) and the candidate hidden state ($\tilde{h}_t$), with the update gate (z_t) controlling the balance between the two:

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

The GRU architecture, with its simplified structure compared to LSTMs, has shown competitive performance in various sequence-to-sequence tasks, such as language modeling, machine translation, and speech recognition, while often requiring fewer parameters and less computational resources.

G. BAGGING MODELS (BOOTSTRAP AGGREGATING)

Introduction to Bagging (Bootstrap Aggregating) Bagging (Bootstrap Aggregating) is a model training technique based on creating sub-samples of data from the original training dataset using bootstrap sampling technique.

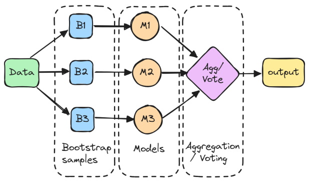
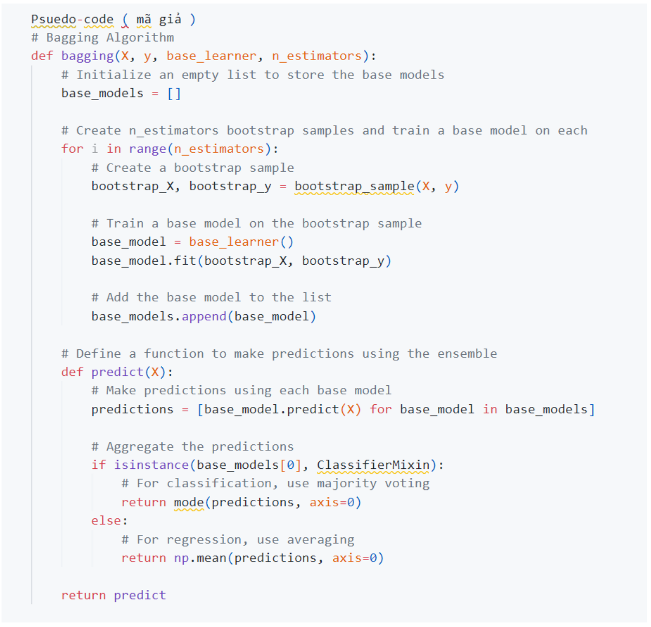


FIGURE 7. BAGGING



```
Psuedo-code ( mã giả )
# Bagging Algorithm
def bagging(X, y, base_learner, n_estimators):
    # Initialize an empty list to store the base models
    base_models = []

    # Create n_estimators bootstrap samples and train a base model on each
    for i in range(n_estimators):
        # Create a bootstrap sample
        bootstrap_X, bootstrap_y = bootstrap_sample(X, y)

        # Train a base model on the bootstrap sample
        base_model = base_learner()
        base_model.fit(bootstrap_X, bootstrap_y)

        # Add the base model to the list
        base_models.append(base_model)

    # Define a function to make predictions using the ensemble
    def predict(X):
        # Make predictions using each base model
        predictions = [base_model.predict(X) for base_model in base_models]

        # Aggregate the predictions
        if isinstance(base_models[0], ClassifierMixin):
            # For classification, use majority voting
            return mode(predictions, axis=0)
        else:
            # For regression, use averaging
            return np.mean(predictions, axis=0)

    return predict
```

FIGURE 8. Enter Caption

How Bagging Works: Bootstrap Sampling: Firstly, a large number of data subsets are created from the original training dataset by randomly sampling with replacement.

Bootstrap Sampling: From the original training dataset, create multiple bootstrap samples, where each sample is created by randomly sampling with replacement from the original dataset. The size of each bootstrap sample is typically the same as the original dataset. Base Learner Training: Train a base model (e.g., decision tree, random forest) on each of the bootstrap samples created in the previous step. This results in multiple base models, each trained on a different subset of the original data. Ensemble Prediction: To make a prediction on a new input, feed the input to each of the base models and aggregate the predictions. For classification tasks, this is typically done by majority voting, while for regression tasks, the predictions are averaged. Here's a pseudo-code for the Bagging algorithm:

Model Training: A base model is trained on each data subset.

Aggregating: The final prediction is computed by combining predictions from all base models. In regression problems, this is often done by averaging predictions from individual models.

Advantages of Bagging: Variance Reduction: By combining multiple base models, Bagging reduces the variance in predictions, thereby improving the stability and accuracy of the model.

Reduces Overfitting: Bagging helps minimize the phenomenon of overfitting by reducing the sensitivity of the model to specific training data.

Flexibility: Bagging can be applied to various types of machine learning models and requires few hyperparameters.

Applications of Bagging: Bagging is commonly used in regression and classification problems to enhance the predictive performance of the model.

It can also be applied to reduce noise in datasets with a high level of noise.

Bagging is a powerful and widely-used model training technique in machine learning to improve prediction performance and mitigate overfitting. By combining data from multiple base models, Bagging creates a generalized and robust model suitable for various real-world applications.

H. FCN(FULLY CONVOLUTIONAL NETWORK)

Fully Convolutional Networks (FCNs) are a type of deep learning architecture that is particularly well-suited for image segmentation tasks. Unlike traditional classification networks that produce a single output label for an entire image, FCNs are designed to generate a dense, pixel-wise prediction, where each pixel in the input image is assigned a class label.

The key features of FCNs are: Fully Convolutional Structure: FCNs replace the fully connected layers found in traditional classification networks with convolutional layers. This allows the network to accept input images of arbitrary size and produce output segmentation maps of corresponding size. End-to-End Training: FCNs can be trained end-to-end, directly mapping input images to output segmentation maps, without the need for any intermediate steps or preprocessing. Efficient Spatial Prediction: By using convolutional layers, FCNs are able to efficiently capture spatial information and context, which is crucial for accurate pixel-wise segmentation. The architecture of an FCN typically consists of the following key components:

Encoder: The encoder part of the network is usually based on a pre-trained classification network, such as VGG, ResNet, or Inception, which serves as a feature extractor. The encoder captures hierarchical features at different scales. Decoder: The decoder part of the network is responsible for gradually upsampling the feature maps from the encoder to the original input size, while also incorporating higher-level semantic information. This is typically achieved using a series of transposed convolutions (also known as deconvolutions or upconvolutions). Skip Connections: To overcome the loss of spatial information during the encoder-decoder process, FCNs often employ skip connections, which combine features from different layers of the encoder with the corresponding layers in the decoder. This allows the network to preserve and integrate low-level, high-resolution details with high-level, semantic information.

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 an 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. ELC DATASET

ELC Dataset's Evaluation				
Model	Training:Testing	RMSE	MAPE (%)	MSLE
LN	7:3	424.41	2.04	0.0008
	8:2	489.90	1.83	0.0006
	9:1	**545.29**	**1.73**	**0.0005**
ARIMA	7:3	11280.61	53.74	0.958
	8:2	9711.93	40.01	0.396
	9:1	**3847.50**	**11.91**	**0.02**
ETS	7:3	421.72	2.07	0.00082
	8:2	485.77	1.84	0.00065
	9:1	**545.35**	**1.77**	**0.00057**
BAGGING	**7:3**	**1995.04**	**6.604**	**0.008**
	8:2	2462.51	8.23	0.012
	9:1	3570.42	13.56	0.024
FCN	7:3	2956.24	10.228	0.021
	8:2	**1650.16**	**5.73**	**0.0055**
	9:1	2443.05	8.47	0.010
GRU	7:3	467.63	2.338	0.00104
	8:2	**552.76**	**2.11**	**0.0008**
	9:1	647.42	2.08	0.00079
LSTM	7:3	648.47	3.02	0.00167
	8:2	**603.40**	**2.41**	**0.00102**
	9:1	1018.54	3.161	0.0018
RNN	7:3	543.106	2.705	0.0012
	8:2	**695.65**	**2.57**	**0.0011**
	9:1	1426.64	4.28	0.0033

TABLE 2. ELC Dataset's Evaluation

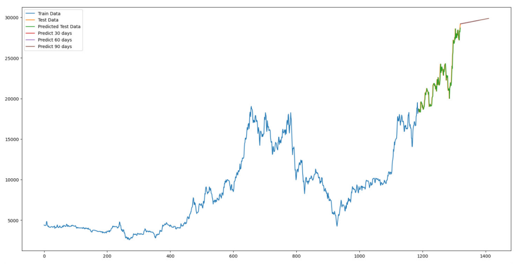


FIGURE 9. Linear model's result with 9:1 splitting proportion

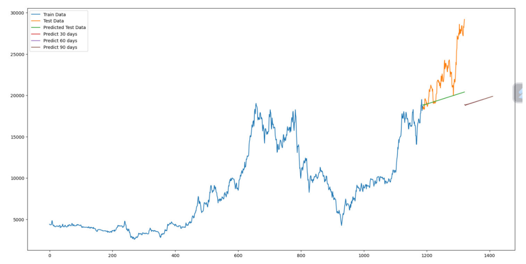


FIGURE 10. ARIMA model's result with 9:1 splitting proportion

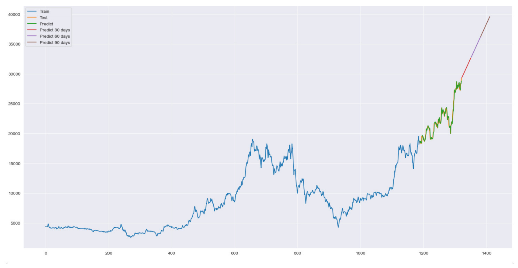


FIGURE 11. ETS model's result with 9:1 splitting proportion

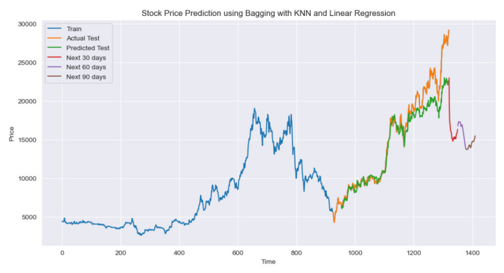


FIGURE 12. BAGGING(KNN-LINEAR) model's result with 7:3 splitting proportion

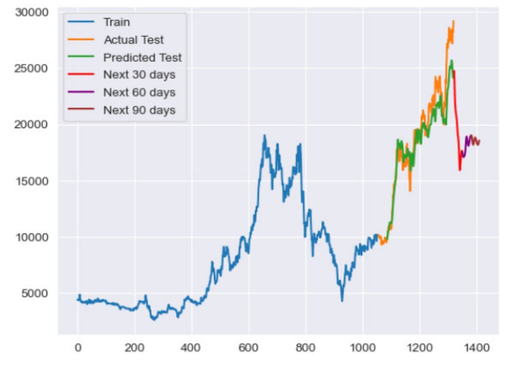


FIGURE 13. FCN model's result with 8:2 splitting proportion

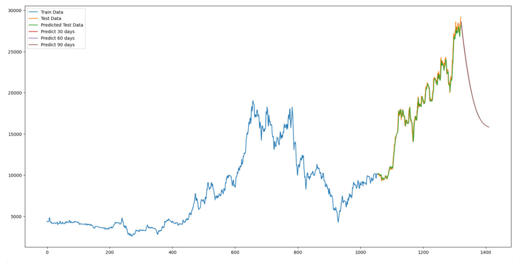


FIGURE 14. GRU model's result with 8:2 splitting proportion

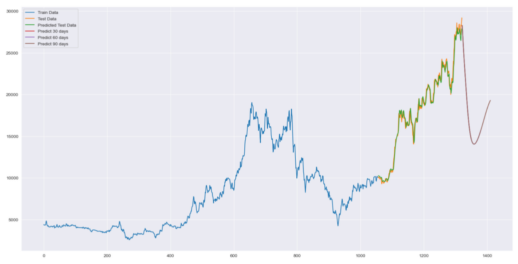


FIGURE 15. LSTM model's result with 8:2 splitting proportion

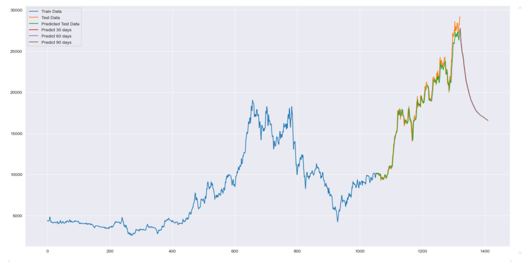


FIGURE 16. RNN model's result with 8:2 splitting proportion

C. FPT DATASET

FPT Dataset's Evaluation				
Model	Training:Testing	RMSE	MAPE (%)	MSLE
LN	**7:3**	**1214.46**	**1.007**	**0.00022**
	8:2	1386.54	1.09	0.0002
	9:1	1587.29	1.108	0.00023
ARIMA	7:3	19968.79	15.68	0.05
	8:2	23349.02	20.38	0.07
	9:1	**19134.86**	**13.81**	**0.037**
ETS	**7:3**	**1197.60**	**1.0034**	**0.0002**
	8:2	1366.62	1.081	0.00022
	9:1	1573.66	1.1004	0.000233
BAGGING	**7:3**	**11598.62**	**8.72**	**0.015**
	8:2	14468.85	12.92	0.024
	9:1	11704.98	8.96	0.012
FCN	**7:3**	**7252.82**	**5.63**	**0.0058**
	8:2	11707.05	10.19	0.0154
	9:1	14867.53	12.99	0.0221
GRU	7:3	2237.98	1.67	0.00055
	8:2	2261.0016	1.75	0.0005
	9:1	**1820.107**	**1.22**	**0.0003**
LSTM	7:3	4394.99	3.37	0.0021
	8:2	4224.28	3.326	0.0018
	9:1	**3092.66**	**2.28**	**0.000903**
RNN	7:3	2757.66	2.07	0.0008
	8:2	4718.35	3.46	0.0021
	9:1	**2388.62**	**1.62**	**0.00049**

TABLE 3. FPT Dataset's Evaluation

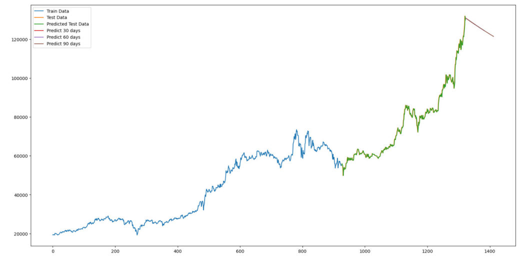


FIGURE 17. Linear model's result with 7:3 splitting proportion

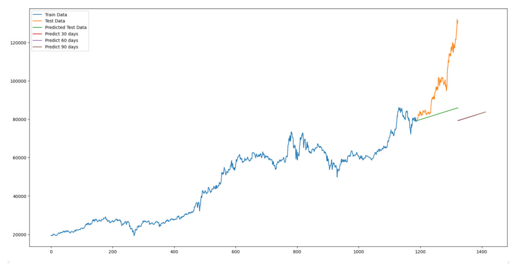


FIGURE 18. ARIMA model's result with 9:1 splitting proportion

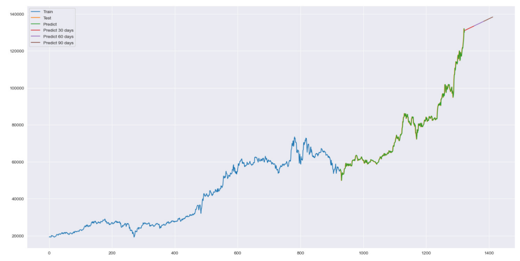


FIGURE 19. ETS model's result with 7:3 splitting proportion

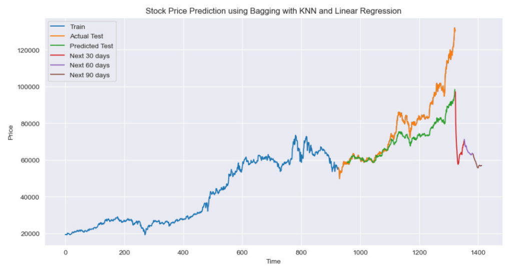


FIGURE 20. BAGGING(KNN-LINEAR) model's result with 7:3 splitting proportion

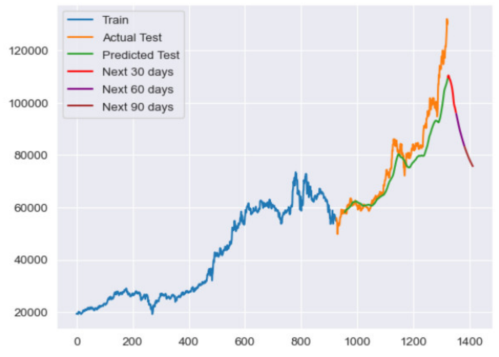


FIGURE 21. FCN model's result with 7:3 splitting proportion

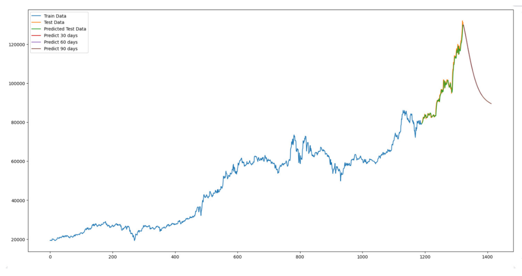


FIGURE 22. GRU model's result with 9:1 splitting proportion

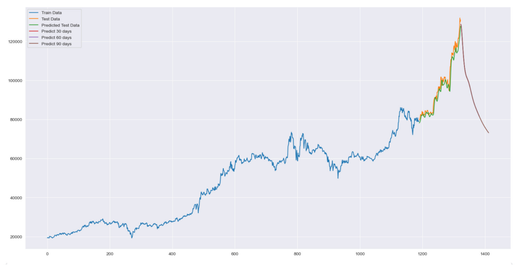


FIGURE 23. LSTM model's result with 9:1 splitting proportion

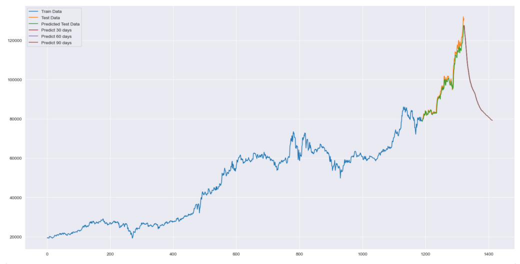


FIGURE 24. RNN model's result with 9:1 splitting proportion

D. SGT DATASET

SGT Dataset's Evaluation				
Model	Training:Testing	RMSE	MAPE (%)	MSLE
LN	7:3	342.27	1.902	0.0007
	8:2	322.40	1.738	0.0005
	9:1	**280.73**	**1.58**	**0.0004**
ARIMA	7:3	2957.11	20.14	0.063
	8:2	2177.45	12.87	0.029
	9:1	**980.96**	**4.691**	**0.005**
ETS	7:3	347.78	1.96	0.00076
	8:2	325.36	1.78	0.00059
	9:1	**283.72**	**1.62**	**0.0005**
BAGGING	7:3	641.006	3.95	0.0023
	8:2	545.03	2.79	0.0015
	9:1	**354.73**	**2.009**	**0.00076**
FCN	7:3	1070.02	7.54	0.0065
	8:2	540.203	2.888	0.0016
	9:1	**412.68**	**2.48**	**0.00104**
GRU	7:3	423.78	2.713	0.0011
	8:2	328.90	1.82	0.0006
	9:1	**281.44**	**1.48**	**0.0004**
LSTM	7:3	444.02	2.43	0.0012
	8:2	**355.89**	**1.88**	**0.00071**
	9:1	791.73	6.11	0.0038
RNN	7:3	398.26	2.25	0.0009
	8:2	398.39	2.32	0.000883
	9:1	**331.11**	**1.95**	**0.00068**

TABLE 4. SGT Dataset's Evaluation

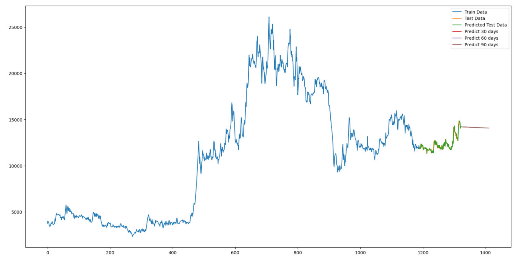


FIGURE 25. Linear model's result with 9:1 splitting proportion

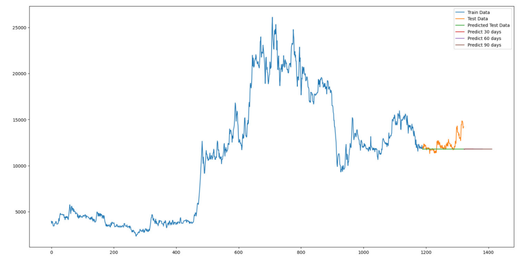


FIGURE 26. ARIMA model's result with 9:1 splitting proportion

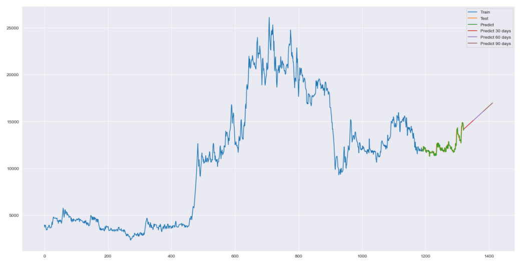


FIGURE 27. ETS model's result with 9:1 splitting proportion

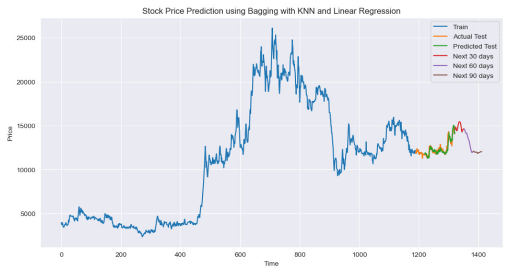


FIGURE 28. BAGGING(KNN-LINEAR) model's result with 9:1 splitting proportion

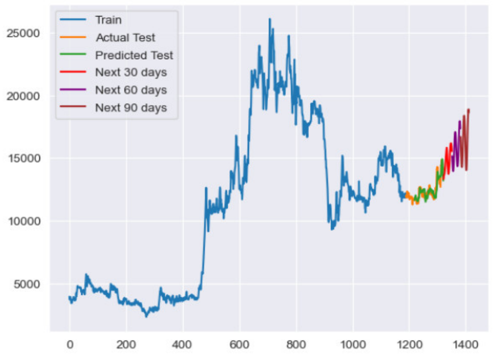


FIGURE 29. FCN model's result with 9:1 splitting proportion

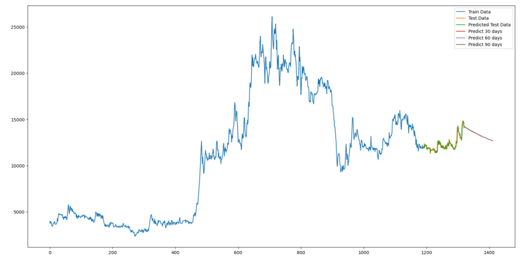


FIGURE 30. GRU model's result with 9:1 splitting proportion

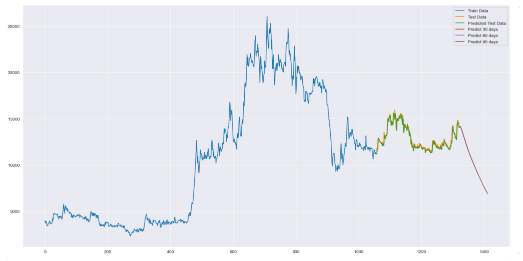


FIGURE 31. LSTM model's result with 8:2 splitting proportion

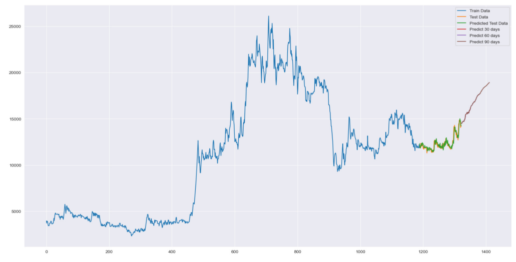


FIGURE 32. RNN model's result with 9:1 splitting proportion

VI. CONCLUSION

A. SUMMARY

In the achievement of forecasting stock prices, the exploration of diverse methodologies, ranging from traditional statistical models to advanced machine learning algorithms, has been aimed. Among the performed models, Recurrent Neural Network (RNN), Fully Convolutional Network(FCN), Bagging(Linear regression and KNN), Gated Recurrent Unit (GRU) are the most promising and effective models for predicting stock prices.

The intricacies of stock price forecasting, rooted in the complexity and unpredictability of financial markets, demand models that can capture nuanced patterns and relationships within the data. FCN, BAGGING, GRU its efficacy in handling intricate relationships, and providing robust predictions. Gated Recurrent Unit (GRU) models, with their ability to capture sequential dependencies, exhibit notable performance in forecasting stock prices. The introduction of ensemble learning through Bagging GRU further refines the predictive capabilities, offering a collective insight that surpasses individual models.

As evidenced by the evaluation metrics, including RMSE, MAPE, and MSLE, the FCN, GRU, and Bagging models consistently demonstrate superior performance across various aspects of forecasting accuracy. Their adaptability to handle the inherent uncertainties of stock markets positions them as formidable tools for investors and analysts seeking reliable predictions.

B. FUTURE CONSIDERATIONS

In our future research, it is crucial to prioritize further optimization of the previously mentioned models. This optimization effort should specifically focus on:

• Enhancing the accuracy of the model. While the above algorithms have demonstrated promising results in predicting stock prices, there is a need to further improve the model's accuracy to ensure more precise forecasting outcomes.

• Exploring alternative machine learning algorithms or ensemble techniques. Ensemble techniques, such as combining multiple models or using various ensemble learning methods, can also improve the robustness and accuracy of the forecasts.

• Researching new forecasting models. The field of forecasting continuously evolves, with new algorithms and models being researched and developed. It is crucial to stay updated with these approaches and explore new forecasting models that offer improved accuracy and performance.

By continuously exploring and incorporating new features, data sources, and modeling techniques, we can strive for ongoing optimization of the forecasting models and enhance their ability to predict stock prices with greater precision and reliability.

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] Hyndman, R. J., Athanasopoulos, G. (2018). Forecasting: Principles and Practice (2nd ed.). OTexts.
[2] YURTSEVER, M., 2021. Gold price forecasting using LSTM, Bi-LSTM and GRU. Avrupa Bilim ve Teknoloji Dergisi, (31), pp.341-347.
[3] Kishanna, H., RamaParvathyb, L. and SIMATS, C., 2022. A Novel Approach for Correlation Analysis on FBProphet to Forecast Market Gold Rates with Linear Regression.
[4] A. O. A. A. A. Ariyo, "Stock Price Prediction Using the ARIMA Model" ,2014. [Online]. Available:https://ieeexplore.ieee.org/document/7046047..
[5] M. S. S. S. A. F. K. Senthamarai Kannan, "Comparison Of Fuzzy Time Series And ARIMA"August 2019. [Online]. Available:https://www.ijstr.org/final-print/aug2019/Comparison-Of-Fuzzy-Time-Series-And-Arima-Model.pdf. [Accessed 19 June 2023].
[6] Liao, T. W. (2019). Ensemble Learning: Foundations and Algorithms. CRC Press
[7] Avner Abrami, Aleksandr Y. Aravkin, Younghun Kim, "Time Series Using Exponential Smoothing Cells", 9 June 2017.
[8] Professor Thomas B. Fomby, "Exponential Smoothing Models", June 2008.
[9] Bauer, E., Kohavi, R. "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants". Machine Learning 36, 105–139 (1999). https://doi.org/10.1023/A:1007515423169.
[10] Buja, A., and Stuetzle, W. "Observations on bagging". University of Pennsylvania and University of Washington, Seattle. 2002.
[11] B. M. Henrique, V. A. Sobrero, and H. Kimura, "Comparison Of Fuzzy Time Series And ARIMA", August 2019. Available:https://www.ijstr.org/final-print/aug2019/Comparison-Of-Fuzzy-Time-Series-And-Arima-Model.pdf. [Accessed 19 June 2023]. 4
[12] Jason Brownlee, "How to Create an ARIMA Model for Time Series Forecasting in Python", November 18, 2023. Available:https://www.ijstr.org/final-print/aug2019/Comparison-Of-Fuzzy-Time-Series-And-Arima-Model.pdf.
[13] Jason Brownlee, "A Gentle Introduction to SARIMA for Time Series Forecasting in Python", August 21, 2019.
[14] Alexandra M. Schmidt and Hedibert F. Lopes, "Dynamic models", 2019.
[15] Timothy O. Hodson, "Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not", 2022, https://doi.org/10.5194/gmd-15-5481-2022.
[16] Seok-Ho Han, Husna Mutahira, Hoon-Seok Jang, "Prediction of Sensor Data in a Greenhouse for Cultivation of Paprika Plants Using a Stacking Ensemble for Smart Farms", Applied Sciences, vol.13, no.18, pp.10464, 2023.