Formosa Journal of Science and Technology (FJST)
Vol. 3, No. 11, 2024: 2445-2464

Design of a Stock Forecasting Dashboard using Python-Streamlit and FB Prophet with AI

Edra Arkananta Gultom[1*], Kartika Dewi Sri Susilowati[2], Anik Kusmintarti[3]
State Polytechnic of Malang
Corresponding Author: Edra Arkananta Gultom,
edraarkanantagultom@gmail.com

ARTICLEINFO

Keywords: Stock Forecasting, Dashboard Design, Artificial Intelligence, Machine Learning

Received : 23, October
Revised : 7, November
Accepted: 20, November

ABSTRACT

This research aims to develop a stock price forecasting application using time series analysis with the Prophet model. The application retrieves historical stock data from Yahoo Finance (2015–present) for Indonesian stocks, which is then processed and analyzed to predict future prices. The study integrates yfinance for data collection, Prophet for forecasting, and Plotly for visualizing the results. The application allows users to select stocks and customize prediction periods (1–4 years). The findings indicate that while the model provides useful short-term predictions, its accuracy is limited by market volatility and external factors. This tool can support decision-making but should be used in conjunction with other forecasting methods.

ISSN-E: 2964-6804
DOI: https://doi.org/10.55927/fjst.v3i11.12216
https://journal.formosapublisher.org/index.php/fjst

INTRODUCTION

The stock market serves as a primary mechanism through which public companies raise capital, making it a critical component of the capital market. Often regarded as an 'economic barometer,' the stock market's fluctuations are closely aligned with broader economic conditions, reflecting both periods of growth and recession. By issuing stocks and bonds, public companies can address temporary capital shortages by attracting investment from the stock market. Stocks are widely available and accessible financial instruments for investors. Through strategic stock selection and effective portfolio management, personal assets have the potential to appreciate in value over time.

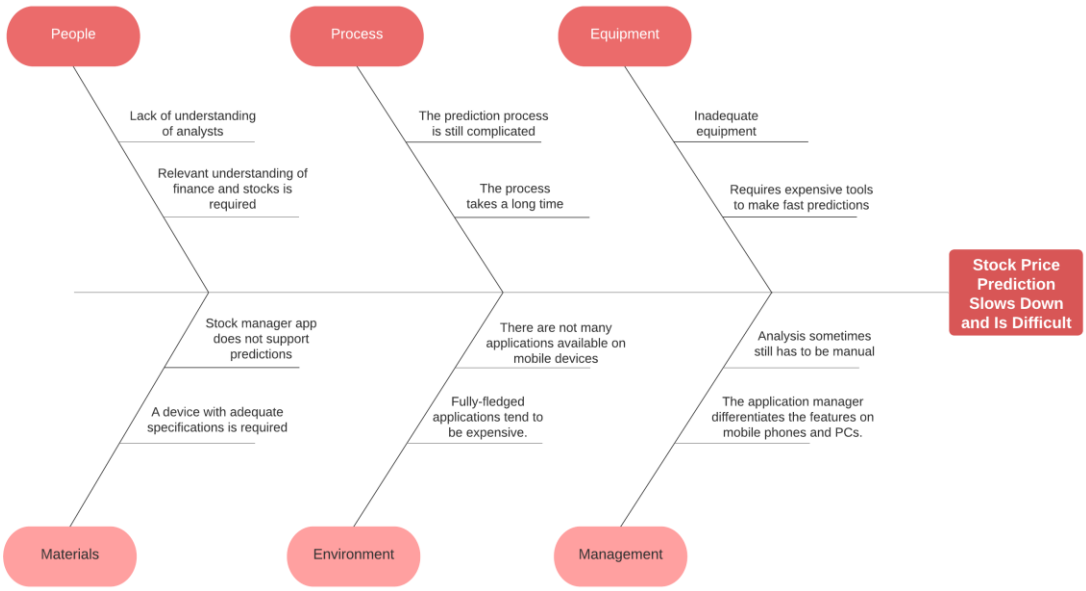
Stock trading has become a core business for securities firms and investment banks. With advancements in computer science, an increasing number of companies are incorporating big data technology and Artificial Intelligence into financial investment activities. According to a 2000 news report, Goldman Sachs once employed over 600 U.S. cash equity traders in New York, yet by 2017, only two equity traders remained, as computers had taken over the majority of the work.

Stock price trend prediction involves assessing the future value of a company's shares, a practice that has existed since the establishment of stock markets. Accurate predictions of future stock prices can yield substantial profits. However, the stock market is influenced by a variety of macro and micro factors, including interconnected political and economic indicators, stock supply-demand dynamics, company operational conditions, and other related factors.

Fama (1965), introduced the Efficient Market Hypothesis (EMH), proposing that the stock market is efficient, with stock prices reflecting all publicly known or newly disclosed information as well as rational expectations. This theory established a foundational basis for subsequent stock prediction studies. Based on the extent of market information and the speed at which the market reacts to information, Fama categorized the market hypothesis into three forms: weak, semi-strong, and strong. The U.S. market is classified as semi-strong, meaning it is robust and stock prices adjust instantaneously to absorb new information.

Burton (1973), a prominent advocate of the EMH, argued that stock prices are unpredictable and follow a random pattern. While the Efficient Market Hypothesis is widely accepted within academic finance circles, real-world market experiences provide contrasting perspectives that challenge the hypothesis of unpredictability. Today, analysts strive to decipher the complexities of the stock market, and several major companies invest substantial resources by hiring experts to develop statistical models for prediction. Several factors explain this trend. Firstly, with the advent of big data technology and cloud computing, it has become much easier to manage and analyze massive stock market data. In many hedge funds and investment firms, quantitative analysis has shifted from manual work to reliance on Artificial Intelligence. Secondly, significant advancements in deep learning have enhanced natural language processing, thus increasing computers' ability to gather and analyze publicly available stock market information. Thirdly, as computers incorporate

more machine learning techniques, they become increasingly intelligent. This discussion leads to the development of a Fishbone analysis model to help identify issues and find optimal solutions as :



Source : Data Processed (2024)
Figure 1. Fishbone Diagram Stock Price Prediction Problem

Artificial Intelligence presents a promising solution for addressing challenges in stock market prediction. In May 2017, the latest version of Google's AlphaGo defeated the world's top Go player, Jie Ke, utilizing advanced deep learning techniques. This achievement suggests that, one day, computers may possess the capability to analyze and predict the stock market with an accuracy that surpasses human securities analysts. Given the critical role of stocks in the economy, accurately predicting future trends and prices holds significant value.

IDXChannel recommends five free technical analysis software options for investors. First, TradingView, a platform that enables free technical analysis of stocks and provides real-time stock prices with various analytical indicators. Second, StockChart, a website offering graphical displays of instruments such as stocks and cryptocurrencies. Third, Investing.com, a website that provides real-time stock price chart analysis and is also accessible via a mobile app. Fourth, MetaTrader 5, a technical analysis software for stocks developed by Russia-based MetaQuotes, available in mobile app form. Fifth, ChartNexus, recommended by IDXChannel, which has over 800,000 users and is used for analyzing stocks, including company fundamental data analysis and technical market indicators.

Several previous studies have explored the use of machine learning for predicting stock market prices. Tay et al. (2001) compared Support Vector Machine (SVM) and Multi-Layer Back Propagation Neural Network (MLP) for financial time series forecasting. Their experiment demonstrated that SVM outperformed MLP in terms of prediction accuracy.

Jain et al. (2015) investigated the performance of Extreme Gradient Boosting (XGBoost) in sales forecasting and found that XGBoost outperformed traditional regression algorithms, providing higher accuracy. Similarly, Sawon et al. (2016) conducted a prediction study on large-scale data to compare the performance of XGBoost with other traditional models, such as linear regression and Random Forest regression. Their results confirmed that XGBoost delivered superior forecasting accuracy compared to traditional methods.

Based on the explanations above, assuming that stock prices can be predicted, this study aims to develop a method for using Machine Learning to predict stock price trends and present the results through a dashboard for visualization. In light of the above discussion, the title of this research is "Design of a Stock Forecasting Dashboard using Python-Streamlit and FB Prophet with AI."

THEORETICAL REVIEW

Previous Stock Predictions

Financial time series forecasting, such as stock market prediction, is often considered a challenging task due to its volatile nature (Wang et al., 2012). Accurately predicting stock market trends and price movements remains an open question today. Over the past few decades, various methodologies have been proposed to refine these predictions. In 1952, American economist Markowitz introduced the concept of portfolio theory in his paper "Portfolio Selection." In this theory, he presented two quantitative investment indices: the mean and the variance of a portfolio's assets.

Markowitz (1952) formalized investor preferences mathematically, using formulas to explain investment diversification and systematically discussing portfolio selection and management. This achievement marked the beginning of using mathematical approaches to understand and master economic market trends. In addition to predicting overall stock market trends, some stock analysts also explore the price fluctuations of individual stocks using statistical methods. For instance, Fung et al. (2002) employed hypothesis testing methods along with a piecewise segmentation algorithm to predict trends in time series. Upward or downward trends were categorized based on the slope of each piecewise segment and the coefficient of determination.

Neural Network

Given the complexity of financial time series data, the introduction of deep learning into financial market prediction has become a popular topic (Cavalcante et al., 2016). Among technological methods, one of the most promising techniques is Artificial Neural Networks (ANN). ANN is a nonlinear computational model capable of making predictions based on market data without learning the relationship between input and output variables beforehand (Maciel et al., 2008). For stock prediction, the most popular type of ANN is the Feed-Forward Neural Network with Back-Propagation for weight training. Another form of ANN that is suitable for stock prediction is the Recurrent Neural Network (RNN).

Many stock analysts use Neural Networks to predict financial markets. Wilson et al. (1994) compared the predictive capabilities of Neural Networks with Classical Multivariate Discriminant Analysis for corporate bankruptcy prediction. The results showed that Neural Networks outperformed other models in predicting bankruptcy. Hsieh et al. (2011) applied an integrated wavelet transform model and Recurrent Neural Network to forecast the stock market based on four stock indices: the Dow Jones Industrial Average, FTSE 100 Index, Nikkei 225 Index, and Taiwan Stock Exchange Capitalization Weighted Stock Index. They demonstrated that this model performed exceptionally well on the four indices and could be applied in real-life scenarios for profit generation. Tanigawa et al. (1990) used Recurrent Neural Networks for stock pattern recognition, achieving accurate results in 15 out of 16 recognition trials.

Unlike conventional Recurrent Neural Networks, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) are better suited for learning from past data to classify, process, and predict time series with varying time step sizes. Both models are specifically designed to address the vanishing gradient problem. Akita et al. (2016) emphasized that LSTM is more effective in capturing the time series effects on stock prices compared to other models. Persio et al. (2016) applied the LSTM model to stock price prediction and concluded that the algorithm significantly improved prediction reliability. Similarly, Chen et al. (2017) used the GRU model for stock market prediction, achieving good performance with minimal errors. Therefore, this study will utilize LSTM and GRU models to predict stock trends.

Definition of Stock Trend

This study will focus on predicting the fluctuations in stock prices. The stock market leaves behind several important trading data after each trading day, such as the opening price, closing price, adjusted closing price, highest price, lowest price, volume, and others. Overall, the adjusted closing price will represent the stock price for that trading day. Over a trading period, there will be a series of adjusted closing prices, which are sequenced as follows:

$$\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\ldots\mathbf{p}_T$$

In this case, $\mathbf{p}_T$ represents the closing price on the **t** trading day, and **T** is the total number of trading days in the period. The stock price on a specific trading day will either increase or decrease compared to the previous trading day. Therefore, the researcher uses the change in the closing price between two consecutive trading days as the evaluation metric. The trading situation is expressed as follows:

$$y_t = \begin{cases} 1 \; if \; p_t > p_t - 1, \\ 0 \; if \; p_t \leq p_t - 1, \end{cases}$$

Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) is a category of Neural Networks that resembles the concept of loops or feedback. After the data is processed by the

hidden layer, the original data is fed back into the hidden layer along with new input data. RNNs are powerful and well-suited for time series prediction. According to Hinton (2013), RNNs possess two key properties: (1) the distributed hidden states enable them to improve the efficiency of storing past information, and (2) the non-linear dynamics help update the hidden states in a more sophisticated manner.

Long Short-Term Memory

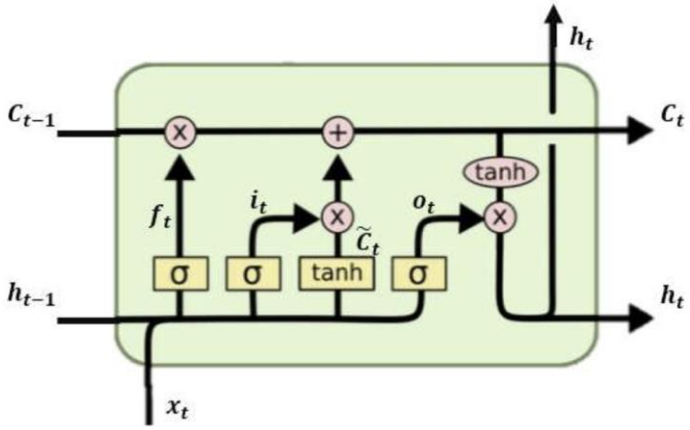


Figure 2. Memory Cell LSTM

Long Short-Term Memory (LSTM) is a variation of Recurrent Neural Networks. Palangi et al. (2012) noted that although RNNs are effective for time series data, they struggle with long-term dependencies due to the vanishing gradient problem. LSTM was introduced as an effective solution to address the vanishing gradient issue by incorporating memory cells that store time-related information. These memory cells include the input gate $\mathbf{i}_t$, output gate $\mathbf{o}_t$, and forget gate $\mathbf{f}_t$. The gates are used to control the interactions between the memory cell itself and the cells within its scope..

At time **t**, the input is $\mathbf{x}_t$, the output is $\mathbf{h}_t$, and the historical information memorized by the memory cell is $\mathbf{C}_t$. The input gate controls the portion of information at time t, the forget gate selects which information to retain or discard from the historical data at time t, and the output gate determines how much information will be passed to the next layer. Since the memory cell has a sequential feature, $\mathbf{C}_t$ contains all the input information from time 0 to time t, representing the state at that particular time t.

The forget gate $\mathbf{f}_t$ is determined by the input $\mathbf{x}_t$ and the previous output $\mathbf{f}_{t-1}$, and is then passed through a sigmoid function. The value of the forget gate oscillates between 0 and 1. When $\mathbf{f}_t$ is 0, it means the previous value is forgotten in the computation. Conversely, if $\mathbf{f}_t$ is 1, the forget gate retains the previous information.

In the input $\mathbf{x}_t$ and previous output $\mathbf{h}_{t-1}$, through a tanh transformation, $\Delta\mathbf{C}_t$ is obtained, with values ranging from -1 to 1. This represents the amount of information that will be added to $\mathbf{C}_t$ after introducing the input $\mathbf{x}_t$. Similar to the forget gate, the input gate $\mathbf{i}_t$ regulates the input information $\mathbf{x}_t$ and the previous

output $\mathbf{h}_{t-1}$. If $\mathbf{i}_t$ is 0, then $\Delta\mathbf{C}_t$ can be ignored. On the other hand, if $\mathbf{i}_t$ is 1, then $\Delta\mathbf{C}_t$ will be fully incorporated into $\mathbf{C}_t$.

The state vector $\mathbf{C}_t$ consists of two parts. The first part is the state vector from the previous time step, $\mathbf{C}_{t-1}$, which is controlled by the forget gate. The second part is $\Delta\mathbf{C}_t$, which determines how much input is required by the gate. The output gate $\mathbf{o}_t$ is similar to both the input and forget gates. The size of the output $\mathbf{o}_t$ determines the likelihood that the state vector $\mathbf{C}_t$ will be output and passed to other layers. The output vector $\mathbf{h}_t$ is a combination of the output gate $\mathbf{o}_t$ and the cell state $\mathbf{C}_t$. $\mathbf{C}_t$. The cell state $\mathbf{C}_t$. $\mathbf{C}_t$ needs to be transformed using the tanh function, and then the output gate determines the proportion of the output.

Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is very similar to Long Short-Term Memory (LSTM), with the main difference being how the gates are applied. In GRU, there are only two gates, instead of three: the reset gate and the update gate. The reset gate decides how much of the new input and previous memory should be combined, and it controls the degree to which the previous memory is discarded. The smaller the value of the reset gate, the more previous memory will be ignored. The update gate determines how much of the previous memory will be retained. The larger the value of the update gate, the more of the previous information will be carried forward. When all reset gate values are set to 1 and update gate values are set to 0, the model effectively becomes a standard Recurrent Neural Network (RNN).

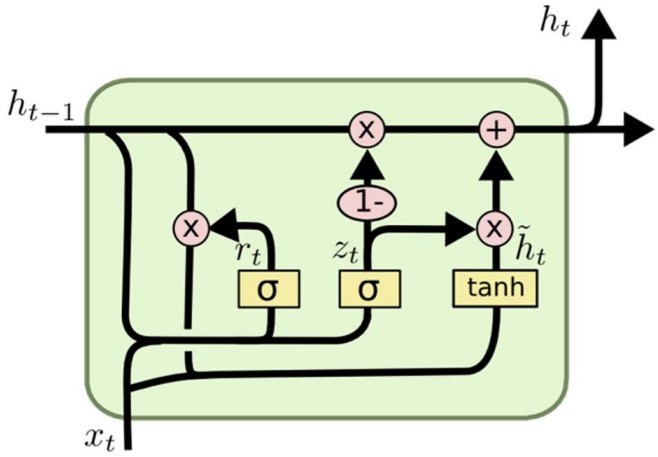


Figure 3. Memory Cell of GRU

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

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

$$\widetilde{h_t} = tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h_t}$$

There are several differences between LSTM and GRU. First, LSTM can control memory exposure, whereas GRU exposes memory without such control. Additionally, GRU does not have an output gate like LSTM. Moreover, the input and forget gates in LSTM are replaced with the update gate in GRU, and the reset gate is directly applied to the previous hidden state. Since GRU has fewer parameters compared to LSTM, it can be initially assumed that GRU will be faster and require less data. However, when the data scale is large, LSTM may produce better results.

Support Vector Machine (VMM)

According to Vapnik (1995), Support Vector Machine (SVM) supports vector machine algorithms. It is based on statistical learning theory for classification and regression. In the case of binary classification, the goal is to find the optimal separating hyperplane (line). This hyperplane divides the data into two categories, with each class located on either side of the line. Boser et al. (1992) described SVM as a maximum margin classifier, where the margin is the distance between the closest training data points from each category.

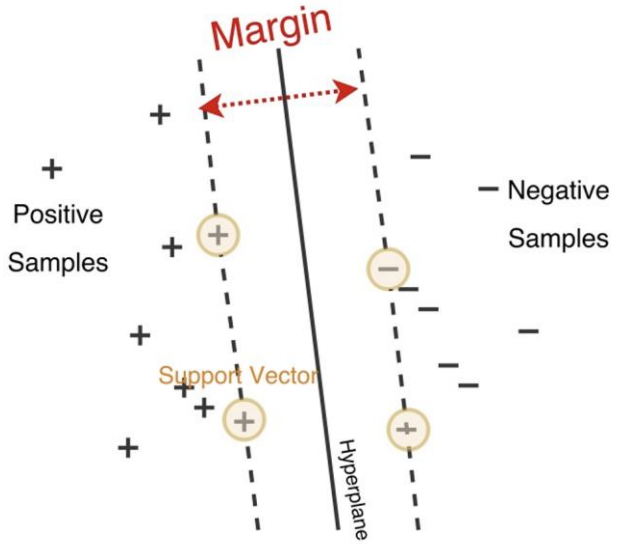


Figure 4. Maximum Margin Illustration for SVM

Let $\{x_1,x_2,x_3,\ldots,x_n\}$ be a set of data, and let $y_i \in \{0,1\}$ be the class label of x_i. The weight vector $w = \{w_1,w_2,w_3,\ldots,w_n\}$ determines the contribution of each input vector to the prediction. Binary linear classification separates all points correctly by using:

$$y_i(w_i * x_i + b) \geq 1, \forall i.$$

Since we want the decision boundary to be as far as possible from the support vectors of both categories, we need to maximize the margin, $\mathbf{m} = \frac{2}{||w||}$. In other words, the larger the margin, the smaller the generalization error of the classifier. Therefore, we need to minimize $\|w\|$, which can be done by solving the following optimization problem:

$$minimize\ \frac{1}{2}\|w\|^2,$$

$$\text{subject to } y_i(w_i * x_i + b) \geq 1, \forall i.$$

The solution can be solved in Lagrungian form. The process will not be shown in this study.

***eXtreme Gradient Boosting* (*XGBoost*)**

Extreme Gradient Boosting (XGBoost) was developed by Tianqi Chen, utilizing a gradient descent algorithm to combine classification and regression trees (CART). Compared to using a single tree in the CART model, the ensemble tree model demonstrates stronger predictive capabilities. These trees vote for the most promising outcome. In other words, gradient descent minimizes the loss by adding new models and correcting the errors of the existing models.

A decision tree is a model that starts with a single leaf node and branches into different possible outcomes. These outcomes then lead to more additional nodes. Each non-leaf node represents a test on a specific feature, each branch represents the result of that feature, and each leaf node stores the classification. After the split for each feature is made, the feature with the minimum loss is considered the best splitting criterion and is set as the rule for that node. The splitting process continues until a stopping condition is met.

Freund et al. (1997) revealed that the Boosting technique is based on the principle that a set of weak classifiers can create a strong classifier. A weak classifier is one that performs only slightly better than a random classifier, while a strong classifier can be corrected with high accuracy. For boosting methods, an additive training approach is applied at each step, where a new classifier is added to the model each week. In XGBoost, the weak classifier is a new decision tree. The equation below illustrates this characteristic:

$$F_0 = 0$$

$$F_t(x) = F_{t-1}(x) + h(x)$$

Here, **h**(x) is the new decision tree added after $F_{t-1}(x)$ dan $F_t(x)$, and $F_{t-1}(x)$ dan $F_t(x)$ is the complete model after step t. The goal of the XGBoost model is to find the tree $F_t(x)$ that minimizes the equation at step t.

$$Obj(F_t) = L(F_{t-1} + F_t) + \Omega(F_t)$$

Here, L is the loss function that determines the predictive strength, and Ω is the regularization function that controls overfitting.

METHODOLOGY

The type of research conducted is Research and Development (R&D). R&D is a research method used to produce a product and test its effectiveness (Sidik, 2019). The development model used in this research is the waterfall model.

According to Solichin (2021), the waterfall development model consists of several stages as follows:

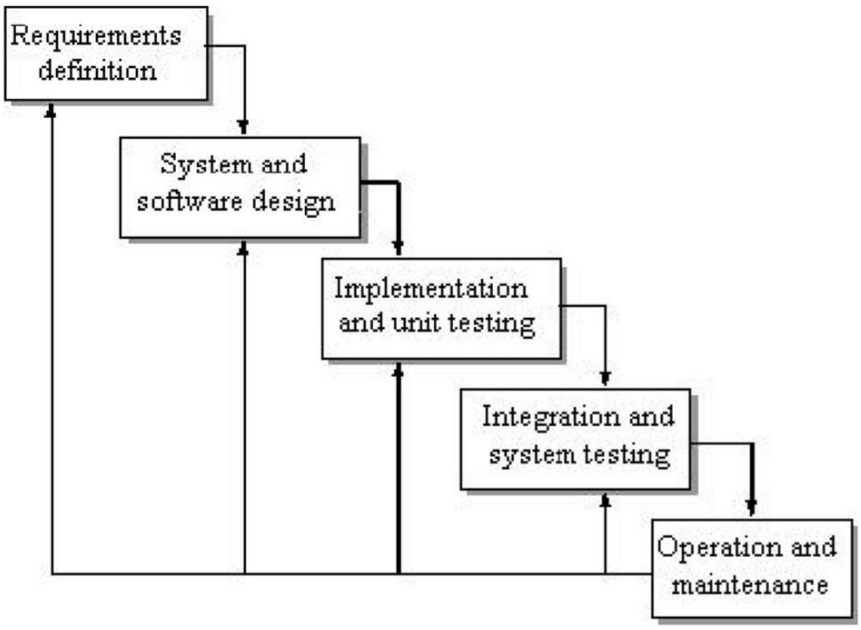


Figure 5. Research Development Model (Solichin, 2021)

The stages implemented in developing this research are as follows in detail:

1. Requirements definition
   This stage is part of the development process, where literature review and data collection are essential to design the flow and determine the algorithms to be applied in the system. Additionally, analyzing the hardware and software requirements will support the progress of the research.
2. System and software design
   The system design phase is used to identify the requirements for the next stages that need to be prepared. This system design illustrates the appearance of the system to be developed and helps define the overall system architecture.
3. Implementation and unit testing
   The implementation phase describes the initial development of the system, which is integrated into the next stage for testing, such as unit testing.
4. Integration and system testing
   In this stage, the system undergoes verification and integration of all models integrated into the system after the testing phase. After the integration stage, the entire system is tested to check for any system failures.
5. Operation and maintenance
   This stage is the final phase in the waterfall development method. Here, the operator is responsible for operating the system and performing maintenance to address any issues that were not identified in the previous stages.

The waterfall development model, according to Solichin, has several advantages and disadvantages. The advantages of this model include its linear

workflow, which helps minimize errors. Additionally, the structured approach with clear direction makes this method a suitable choice for system design. However, the weakness of this model is its lack of flexibility in project execution. Due to its linear nature, making changes from the initial concept is challenging. If there is undocumented information, it becomes impossible to implement changes.

RESULTS

Dataset

In this research, the dataset for stock forecasting was obtained using Python through the integration of several libraries, including **Streamlit** for user interface development, **yfinance** for retrieving historical stock data, and **Prophet** for forecasting analysis. Specifically, **yfinance** was employed to import daily stock data, such as opening and closing prices, high and low prices, and volume, which served as the basis for forecasting. The **Prophet** model, a robust time-series forecasting tool, was used for trend prediction and analysis. Additionally, **Plotly** was used for visualization, enabling interactive charting of forecasting results. This combination of tools provided a reliable and efficient way to collect, process, and visualize stock data for analysis.

```
import streamlit as st
from datetime import date

import yfinance as yf
from prophet import Prophet
from prophet.plot import plot_plotly
from plotly import graph_objs as go
```

In conducting stock forecasting analysis, defining the data timeframe is essential for capturing long-term trends and seasonality. For this purpose, the period from January 1, 2015, to the present date was selected to provide a robust dataset covering recent market fluctuations and historical trends.

```
START = "2015-01-01"
TODAY = date.today().strftime("%Y-%m-%d")
```

To enhance user interaction and flexibility in forecasting, this application offers a user-friendly interface for stock selection and prediction timeframe customization. Users can choose from a range of popular Indonesian stocks, presented as options in a dropdown menu, and select the specific stock dataset they wish to analyze. Additionally, a slider feature allows users to set the desired forecast period, from one to four years, providing tailored predictions aligned with individual forecasting needs.

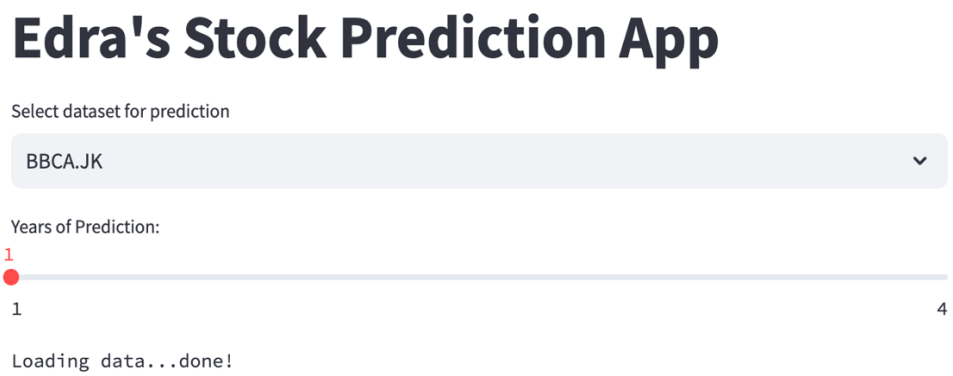
```
st.title("Edra's Stock Prediction App")
```

```
stocks = ("BBCA.JK", "BBRI.JK", "TLKM.JK", "PGAS.JK", "PTBA.JK", "SCMA.JK", "SMGR.JK", "TINS.JK",
"SIDO.JK", "BMRI.JK", "ASII.JK", "BRPT.JK", "CPIN.JK", "JPFA.JK", "EMTK.JK", "ERAA.JK", "EXCL.JK",
"GGRM.JK", "INKP.JK", "HMSP.JK", "INTP.JK", "BSDE.JK", "ICBP.JK", "INCO.JK", "INDF.JK",
"BDMN.JK", "KLBF.JK", "MYOR.JK", "UNTR.JK")
selected_stocks = st.selectbox("Select dataset for prediction", stocks)

n_years = st.slider("Years of Prediction:", 1 , 4)
period = n_years * 365
```

Load Data

The app employs a streamlined data loading function to fetch historical stock data efficiently. By using @st.cache_data, it ensures that data for each selected stock is downloaded and cached, reducing the need to re-download data during the session. This approach helps improve the app's performance, especially when working with large datasets. Once the data is loaded, the app updates the user with a "Loading data...done!" message, confirming that the historical stock data is ready for analysis.

```
@st.cache_data
def load_data(ticker):
    data = yf.download(ticker, START, TODAY)
    data.reset_index(inplace=True)
    return data

data_load_state = st.text("Load data...")
data = load_data(selected_stocks)
data_load_state.text("Loading data...done!")
```



Source : Data Processed (2024)

Figure 6. Interface of Edra's Stock Prediction App for Selecting Dataset and Prediction Period

Pre Processing

To begin the analysis, it's important to inspect the raw stock data to understand its structure and identify key variables. The code below displays the last few rows of the dataset using the st.write(data.tail()) function. This allows us to quickly review the most recent data points and confirm that the dataset is

loaded correctly. By examining the raw data, we can identify any anomalies or trends that may inform the subsequent analysis and modeling stages.

```
st.subheader('Raw data')
st.write(data.tail())
```

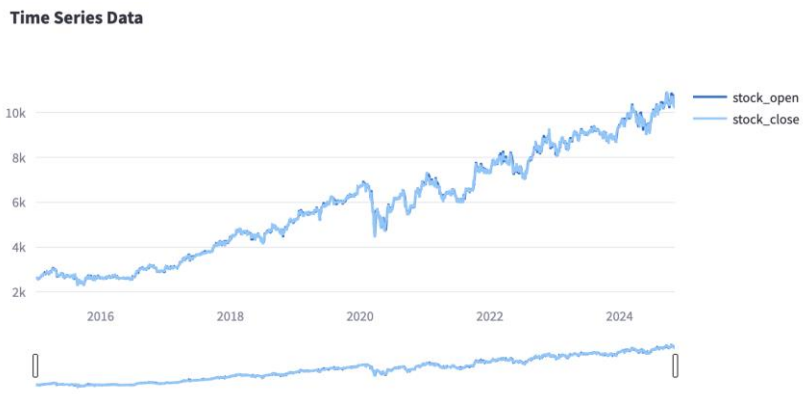
Raw data

	Date	Open	High	Low	Close	Adj Close	Volume
2,436	2024-11-01 00:00:00	10,275	10,425	10,275	10,425	10,425	47,710,000
2,437	2024-11-04 00:00:00	10,425	10,500	10,300	10,375	10,375	49,275,700
2,438	2024-11-05 00:00:00	10,300	10,575	10,300	10,500	10,500	50,296,500
2,439	2024-11-06 00:00:00	10,475	10,500	10,375	10,450	10,450	50,324,900
2,440	2024-11-07 00:00:00	10,325	10,450	10,175	10,175	10,175	131,579,100

Source : Data Processed (2024)
Figure 7. Interface of Edra's Stock Prediction App Raw Data

To visually explore the stock data, we generate a time series plot showing both the opening and closing prices. The code below uses Plotly to create an interactive chart that displays these two key price points over time. This visualization helps in identifying trends and patterns in the stock's performance.

```
def plot_raw_data():
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=data['Date'], y=data['Open'], name='stock_open'))
    fig.add_trace(go.Scatter(x=data['Date'], y=data['Close'], name='stock_close'))
    fig.layout.update(title_text="Time Series Data", xaxis_rangeslider_visible=True)
    st.plotly_chart(fig)

plot_raw_data()
```

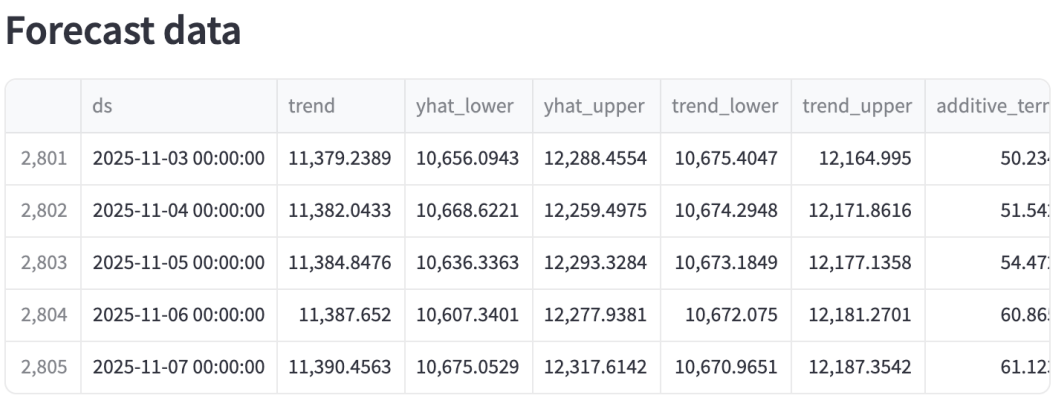


Source : Data Processed (2024)
Figure 8. Interface of Edra's Stock Prediction App Time Series Plot Data

Forecasting with FB Prophet

To forecast future stock prices, we use the FBProphet model, which is designed for time series forecasting. The code below prepares the training data by renaming the columns to the required format, fits the model to historical stock data, and generates future predictions for the specified period. The resulting forecast is then displayed for further analysis.

```
# Forecasting with FBProphet
df_train = data[['Date', 'Close']]
df_train = df_train.rename(columns={"Date": "ds", "Close": "y"})

m = Prophet()
m.fit(df_train)
future = m.make_future_dataframe(periods=period)
forecast = m.predict(future)

st.subheader('Forecast data')
st.write(forecast.tail())
```



Forecast data

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terr
2,801	2025-11-03 00:00:00	11,379.2389	10,656.0943	12,288.4554	10,675.4047	12,164.995	50.23
2,802	2025-11-04 00:00:00	11,382.0433	10,668.6221	12,259.4975	10,674.2948	12,171.8616	51.54
2,803	2025-11-05 00:00:00	11,384.8476	10,636.3363	12,293.3284	10,673.1849	12,177.1358	54.47
2,804	2025-11-06 00:00:00	11,387.652	10,607.3401	12,277.9381	10,672.075	12,181.2701	60.86
2,805	2025-11-07 00:00:00	11,390.4563	10,675.0529	12,317.6142	10,670.9651	12,187.3542	61.12

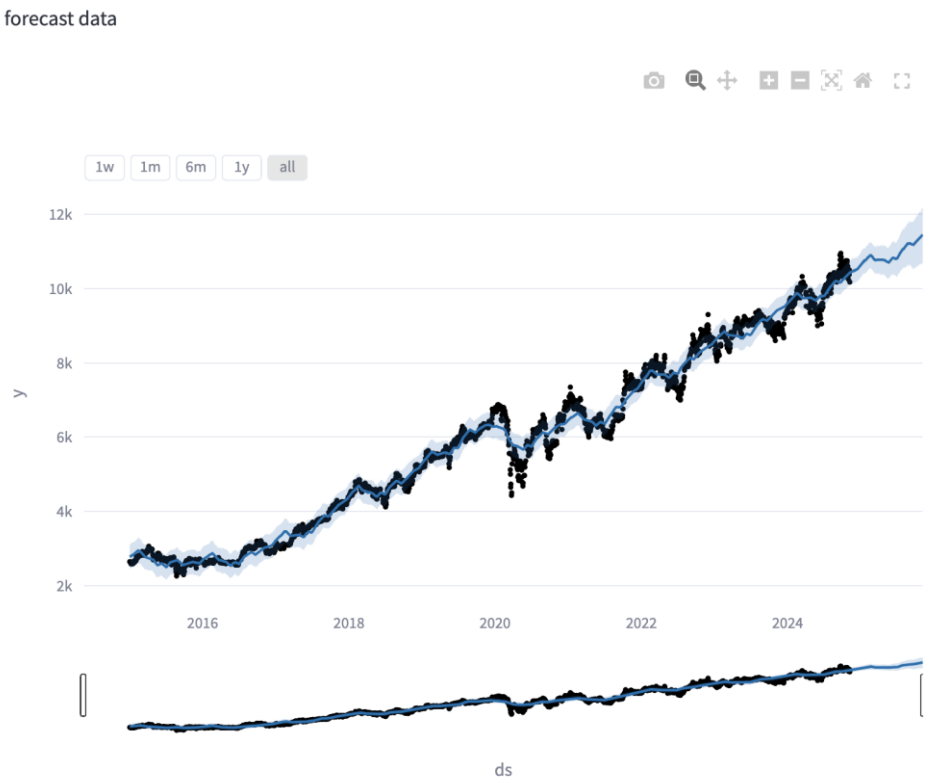
Source : Data Processed (2024)

Figure 9. Interface of Edra's Stock Prediction App Forecast Data

Predicted Result

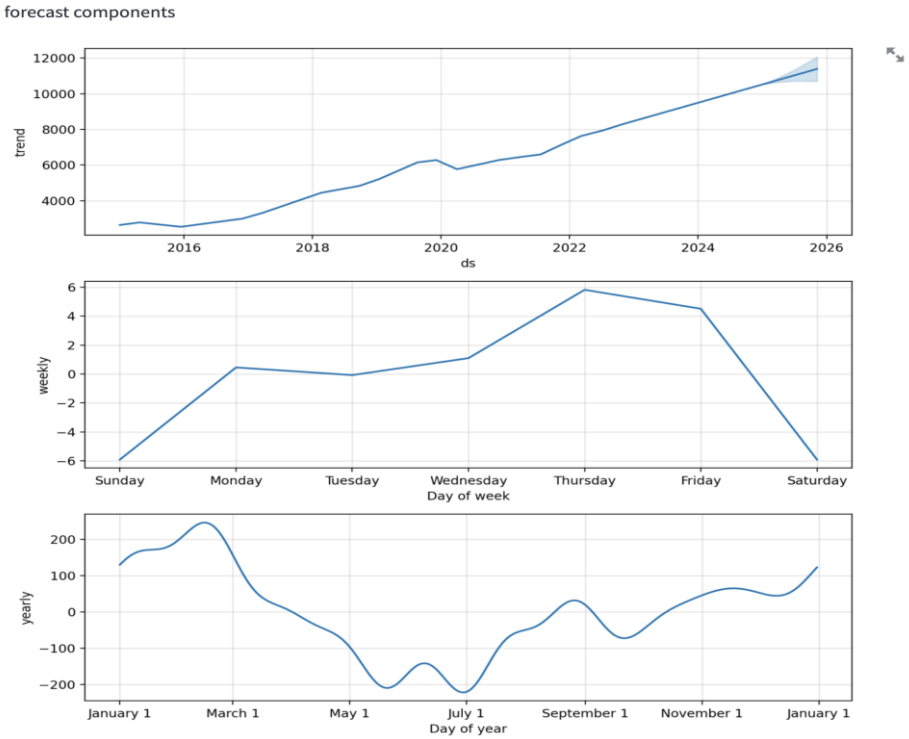
To assess the forecast's accuracy and underlying patterns, we visualize the predicted stock prices and their components. The following code generates two plots: one for the overall forecast and another to display the trend and seasonality factors.

```
st.write('forecast data')
fig1 = plot_plotly(m, forecast)
st.plotly_chart(fig1)

st.write('forecast components')
fig2 = m.plot_components(forecast)
st.write(fig2)
```



Source : Data Processed (2024)

Figure 10. Interface of Edra's Stock Prediction App Forecast Data (Plot)



Source : Data Processed (2024)

Figure 11. Interface of Edra's Stock Prediction App Forecast Components (Plot)

DISCUSSION

This research demonstrates the effective application of time series forecasting for stock price prediction using a combination of Python libraries. By leveraging yfinance, we were able to retrieve historical stock data, which served as the foundation for the forecasting process. The use of Prophet, a robust tool for handling time series data, allowed us to generate reliable forecasts by capturing underlying trends and seasonality in the stock market. The ability to visualize these forecasts through interactive charts using Plotly enhanced our understanding of the predictions and their components, including trends and seasonal fluctuations.

The results highlight the importance of selecting an appropriate timeframe, as we chose data from January 1, 2015, to the present, which covers a range of market conditions and fluctuations. This comprehensive dataset contributed to the model's ability to account for recent market behavior. Additionally, the user-friendly interface, which includes options for stock selection and forecast period customization, enables users to tailor the analysis to their specific needs.

While the model provides valuable insights, it's important to acknowledge the inherent challenges of stock forecasting, such as market volatility, external factors, and model limitations. The forecasts generated by Prophet are based on historical data, and while they offer useful predictions, they cannot fully account for unpredictable market events. Therefore, while the application offers a reliable method for short-term forecasting, users should consider it as one of many tools in the decision-making process rather than as a definitive predictor of future stock prices. Further improvements could include incorporating additional factors, such as financial indicators or sentiment analysis, to enhance the model's accuracy and robustness.

CONCLUSIONS AND RECOMMENDATIONS

This study successfully applied time series forecasting techniques, utilizing the FBProphet model to predict stock prices based on historical data. By integrating tools such as yfinance, Prophet, and Plotly, we were able to efficiently collect, process, and visualize stock data, providing insightful forecasts. The application allows for a user-friendly interface, enabling users to customize their analysis by selecting stocks and adjusting the forecast period. The results demonstrate that, despite the challenges posed by stock market volatility, the forecasting model offers valuable insights into potential future price movements. The model's ability to capture trends and seasonality highlights its usefulness for short-term forecasting, particularly for investors seeking a data-driven approach to decision-making.

While the forecasting model demonstrates promising results, there are several areas for improvement. Future research could focus on incorporating additional external factors, such as macroeconomic indicators, company-specific data, or social sentiment analysis, to improve the model's predictive accuracy. Moreover, expanding the dataset to include a broader range of stocks or global market indices could offer a more comprehensive view of stock behavior. Additionally, exploring more advanced machine learning models, such as LSTM

(Long Short-Term Memory) networks, could provide deeper insights into stock price forecasting. It is also recommended that users approach stock forecasting with caution, as predictions are based on historical patterns and may not fully account for sudden market changes. Therefore, the use of such models should be combined with expert judgment and other forecasting tools to make informed investment decisions.

FURTHER STUDY

While this research provides a solid foundation for stock price forecasting using time series methods, there are numerous avenues for further study that could enhance the robustness and accuracy of the predictive model.

1. Incorporating Additional Data Sources
   One potential improvement is integrating external factors, such as macroeconomic indicators (e.g., interest rates, inflation, GDP growth) or geopolitical events, which could have a significant impact on stock prices. Incorporating news sentiment analysis or social media trends could also provide a more comprehensive view of market behavior, potentially enhancing the model's ability to capture unforeseen market movements.
2. Expanding the Dataset
   The study focused on a limited dataset of Indonesian stocks from 2015 to the present. Future research could extend the analysis to include a broader selection of stocks across different sectors and regions, which would allow for comparisons and potentially uncover more generalized trends that are applicable to a wider market. Additionally, using high-frequency trading data (e.g., minute or hourly data) could improve the granularity and accuracy of predictions.
3. Advanced Forecasting Techniques
   Exploring advanced machine learning and deep learning models, such as Long Short-Term Memory (LSTM) networks, Random Forests, or XGBoost, may further enhance the forecasting process. These techniques have been shown to capture complex non-linear patterns in data, potentially improving accuracy, especially for volatile or non-stationary stock price movements.
4. Model Evaluation and Validation
   Another important area for future research is improving model evaluation techniques. While Prophet is a reliable tool for time series forecasting, its accuracy could be evaluated using various performance metrics such as Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), or backtesting strategies, ensuring that the predictions align with real-world outcomes. Moreover, implementing cross-validation techniques could help in better generalizing the model's performance across unseen data.
5. Real Time Forecasting
   Implementing a real-time forecasting system that dynamically updates predictions as new data becomes available would be a valuable addition. This would allow users to receive up-to-date forecasts in real-time, making

it more relevant for active traders and investors who require immediate insights.

By addressing these areas, future studies can refine stock forecasting models, making them more adaptable, accurate, and capable of providing actionable insights for both short-term traders and long-term investors.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my esteemed research collaborators for their guidance, support, and valuable contributions throughout this study. Their expertise has been essential to the success of this research. I also appreciate the assistance of everyone who contributed to making this work possible.

REFERENCES

Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016). Deep learning for stock prediction using numerical and textual information. In *Computer and Information Science (ICIS)*, IEEE/ACIS 15th International Conference on (pp. 1-6). IEEE. https://doi.org/10.1109/ICIS.2016.19

Boser, B. E., Guyon, I., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *The Fifth Annual Workshop of Computational Learning Theory*, 5, 144-152. Pittsburgh, ACM.

Cavalcante, R.C., Brasileiro, R.C., Souza, V.L.F., Nobrega, J.P., & Oliveira, A.L.I. (2016). Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, 55, 194-211.

Chen, W., Zhang, Y., Yeo, C.K., Lau, C.T., & Lee, B.S. (2017). Stock market prediction using neural network through news on online social networks. *In Smart Cities Conference (ISC2), 2017 International* (pp. 11-12). IEEE.

Fama, E. F. (1965). The behavior of stock-market prices. *Journal of Business, 38*(1), 34-105. https://doi.org/10.1086/294743

Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 119-139.

Fung, G. P. C., Yu, J. X., & Lam, W. (2002). News sensitive stock trend prediction. In *Advances in Knowledge Discovery and Data Mining* (pp. 481-493). Springer Berlin Heidelberg.

Hinton, G. E., et al. (2013). "Learning representations by back-propagating errors." *Nature*, 323(6088), 533-536. https://doi.org/10.1038/323533a0

Hsieh, T., Hsiao, H., & Yeh, W. (2011). Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. *Applied Soft Computing*, 11(2), 2510-2525.

Jain, A., Menon, M.N., & Chandra, S. (2015). Sales Forecasting for Retail Chains. *Computing Science*, India.

Maciel, L.S., & Ballini, R. (2008). Design a neural network for time series financial forecasting: Accuracy and robustness analysis. *Instituto de Economia, Universidade Estadual de Campinas, Sao Paulo-Brasil*.

Malkiel, B. G. (1973). A random walk down Wall Street. W.W. Norton & Company.

Markowitz, H. (1952). Portfolio selection. *The Journal of Finance, 7*(1), 77-91.

Palangi, H., Ward, R., & Deng, L. (2016). Distributed compressive sensing: A deep learning approach. *IEEE Transactions on Signal Processing*, 64(17), 4504-4518. https://doi.org/10.1109/TSP.2016.2584962

Persio, L., & Honchar, O. (2016). Artificial neural networks architectures for stock price prediction: comparisons and applications. *International Journal of Circuits, Systems and Signal Processing, 10*, 403–413.

Sawon, M., & Hosen, M. (2016). Prediction on large scale data using extreme gradient boosting. *BSc Thesis Report*, Computer Science and Engineering, BRAC University, Dhaka, Bangladesh.

Sidik, M. (2019). Perancangan dan Pengembangan E-commerce dengan Metode Research and Development. *Jurnal Teknik Informatika UNIKA Santo Thomas*, 4(1), 99-107.

Solichin. (2021). Pengembangan dan Pengujian Aplikasi Pemesanan Makanan berbasis Website Menggunakan Metode Waterfall. *Journal of Computer Science an Engineering (JCSE)*, 2(1), 40–50.

Tanigawa, T., & Kamijo, K. (1990). Stock price pattern recognition: A recurrent neural network approach. *Proceedings of the 1990 IJCNN International Joint Conference on Neural Networks*, 215-221. https://doi.org/10.1109/IJCNN.1990.137572

Tay, F.E.H., & Cao, L. (2001). Application of Support Vector Machines in Financial Time Series Forecasting. *Omega*, 29(4), 309-317.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag.

Wang, B., Huang, H., & Wang, X. (2012). A novel text mining approach to financial time series forecasting. *Neurocomputing, 83*(6), 136-145.

Wilson, R.L., & Sharda, R. (1994). Bankruptcy prediction using neural networks. *Decision Support Systems*, 11(5), 545-557.