Optimizing Stock Market Prediction Decisions using Artificial Neural Networks and Dynamic Programming

Rizqika Mulia Pratama - 13522126[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522126@mahasiswa.itb.ac.id

***Abstract*—This paper presents an innovative approach that combines artificial neural networks specifically Long Short-Term Memory (LSTM) models, with dynamic programming to optimize stock market prediction decisions. The primary goal is to predict stock prices accurately and determine the best investment strategies to maximize returns. This method involves training an LSTM model on historical stock data to forecast future prices and then using dynamic programming to optimize investment decisions based on these predictions. The results demonstrate significant potential for this approach in enhancing decision-making processes in the stock market.**

***Keywords*—Stock Market Prediction, LSTM, Dynamic Programming. Investment Strategy**

I. INTRODUCTION

In the fast-paced and volatile environment of the stock market, accurate predictions and effective investment strategies are crucial. The advent of artificial intelligence and machine learning, particularly deep learning models like Long Short-Term Memory (LSTM) networks, has revolutionized predictive analytics in finance. This paper explores the application of LSTM networks to predict stock prices and leverages dynamic programming to optimize investment decisions based on these predictions.

The stock market is a complex system influenced by a multitude of factors, including economic indicators, company performance, investor sentiment, and global events. Traditional methods of stock market analysis, such as technical analysis, often fall short in capturing the dynamic and nonlinear relationships present in the data. Machine learning models, particularly LSTM networks, offer a powerful alternative by leveraging their ability to model temporal dependencies and capture complex patterns in time series data.

This paper aims to develop a comprehensive framework that integrates LSTM-based stock price predictions with dynamic programming for optimal decision-making. By utilizing historical stock data, the LSTM model forecasts future prices, which are then used to determine the best investment strategies using dynamic programming. This approach not only enhances prediction accuracy but also provides a systematic methodology for maximizing returns in stock trading.

II. LITERATURE REVIEW

A. Graph Theory

A graph is a collection of points or nodes connected by edges. A graph G = (V, E) is defined, where V is a non-empty set of nodes, denoted as n $\{v_1, v_2, \dots, v_n\}$, and E is a set of edges connecting pairs of nodes $\{e_1, e_2, \dots, e_n\}$.

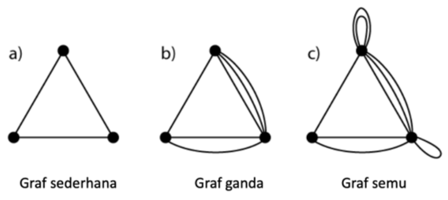


Figure 1. Types of Graphs based on bracelets and double sides.
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf

Based on the presence of loops or multiple edges, a graph can be classified into three types:

1. *Graf sederhana* (simple graph), which does not contain loops or multiple edges.
2. *Graf ganda* (multigraph), which contains multiple edges.
3. *Graf semu* (pseudograph), which contains both loops and multiple edges.

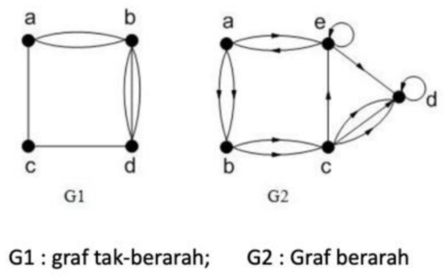


Figure 2. Graph types are based on the direction orientation of the graph.
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf

Based on the orientation of edges, a graph can be categorized into two types:

1. *Graf tak-berarah* (Undirected graph), which does not have a specified direction for its edges.

2. *Graf berarah* (Directed graph), which has a specified direction for its edges.

Here are some terminologies related to graphs:

1. Adjacency, two nodes are considered adjacent if they are connected by an edge.
2. Incident, an edge is said to be an incident to a node if the edge connects the node.
3. Isolated node, an isolated node is a node that has no incident edges.
4. Empty graph, an empty graph is a graph with an empty set of edges.
5. Degree, the degree of a node is the number of edges incident to that node.
6. Path, a path of length n from the initial node v0 to the destination node v_n is a sequence pf alternating node and edges (v_0, e_1, v_1, e_1, … , e_1, v_n) such each $e_1 = (v_0, v_1)$, … , $e_n = (v_{n-1}, v_n)$ is an edge in the graph.
7. Connected, two nodes are considered connected if there is a path from the initial node.

B. Dynamic Programming

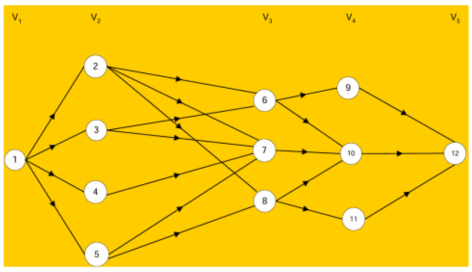


Figure 3. Dynamic Programming Illustration.
Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf

Dynamic programming is a mathematical optimization approach created using graphs that solves complex problems by breaking them down into simpler subproblems. It is particularly useful in decision-making processes where the goal is to find the optimal strategy over time, such as in stock trading. Dynamic programming involves defining a set of states and actions, and recursively computing the value of each state based on the possible actions and their outcomes.

In the context of stock market prediction, dynamic programming can be used to determine the optimal investment strategy based on predicted stock prices. By considering different states (e.g., holding cash, holding stock) and actions (e.g., buying, selling), dynamic programming can identify the sequence of actions that maximizes the final investment value

C. Stock Market Prediction

Predicting stock prices is inherently challenging due to market volatility and the influence of numerous factors. Traditional methods, such as technical analysis and fundamental analysis, rely on historical price patterns and financial indicators to forecast future prices. However, these methods often fail to capture the complex and nonlinear relationship present in the data.

Machine learning models, including LSTM networks, offer a more robust approach to stock price prediction. By training on large datasets of historical prices and other relevant features, these models can learn intricate patterns and make accurate predictions. Previous studies have demonstrated the effectiveness of LSTM networks in predicting stock prices, showing that they can outperform traditional methods and other machine learning models.

D. Long Short-Term Memory (LSTM)

LSTM networks, a type of recurrent neural network (RNN), are designed to model temporal sequences and long-term dependencies more effectively than traditional RNNs. Introduced by Hochreiter and Schmidhuber in 1997, LSTM networks address the vanishing gradient problem inherent in standard RNNs by incorporating memory cells that can maintain information over extended periods. This makes LSTM networks particularly suitable for time series forecasting, including stock price predictions.

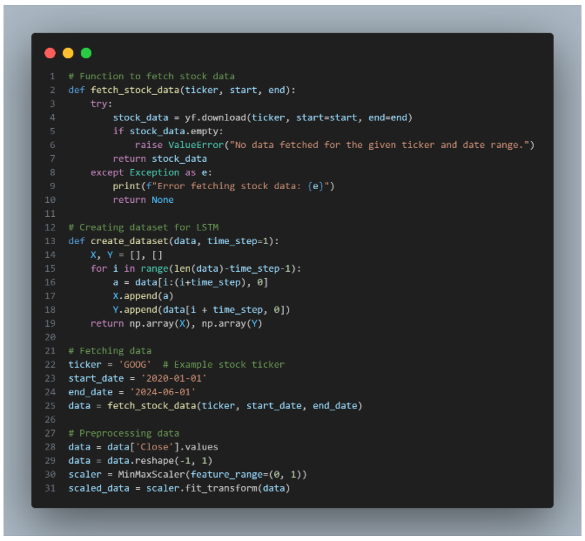
LSTM networks have been widely used in various applications, such as natural language processing, speech recognition, and financial forecasting. Their ability to capture intricate temporal patterns and dependencies in sequential data has made them a popular choice for predicting stock prices, where historical prices and trends play a crucial role in future price movements.

III. Methodology

A. Data Scraping

Historical stock data for a specific ticker (e.g., Google Inc., ticker: GOOG) is collected using Yahoo Finance. The data spans from January 1, 2020, to June 1, 2024. This data includes daily closing prices, which are essential for training the LSTM model. The data is downloaded and stored in a structured format for further processing

Here's the code to scrape the Yahoo Finance data:



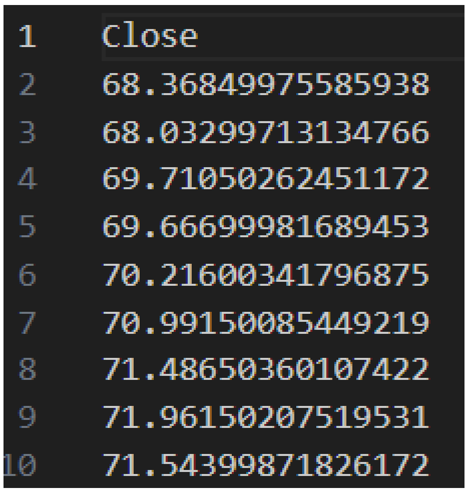
```python
# Function to fetch stock data
def fetch_stock_data(ticker, start, end):
    try:
        stock_data = yf.download(ticker, start=start, end=end)
        if stock_data.empty:
            raise ValueError("No data fetched for the given ticker and date range.")
        return stock_data
    except Exception as e:
        print(f"Error fetching stock data: {e}")
        return None

# Creating dataset for LSTM
def create_dataset(data, time_step=1):
    X, Y = [], []
    for i in range(len(data)-time_step-1):
        a = data[i:(i+time_step), 0]
        X.append(a)
        Y.append(data[i + time_step, 0])
    return np.array(X), np.array(Y)

# Fetching data
ticker = 'GOOG'  # Example stock ticker
start_date = '2020-01-01'
end_date = '2024-06-01'
data = fetch_stock_data(ticker, start_date, end_date)

# Preprocessing data
data = data['Close'].values
data = data.reshape(-1, 1)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)
```

Figure 4. Data Scraping Using Yahoo Finance.

Source: personal documentation

In this step, data was scraped and stored in CSV format for further processing process.

Here's the result in CSV format:



```
1   Close
2   68.36849975585938
3   68.03299713134766
4   69.71050262451172
5   69.66699981689453
6   70.21600341796875
7   70.99150085449219
8   71.48650360107422
9   71.96150207519531
10  71.54399871826172
```

Figure 5. Scrapped Google Stock Data in CSV Format.
Source: personal documentation

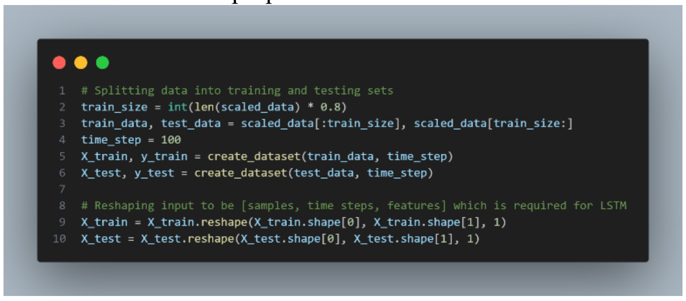
B. Data Processing

The collected data is preprocessed to scale the values between 0 and 1 using MinMaxScaler. This step is crucial for ensuring that the LSTM model performs well, as neural networks generally work better with normalized data. The data is then split into training and testing sets, with 80% used for training and 20% for testing. This split ensures that the model can learn from historical data while being evaluated on unseen data to assess its predictive performance.

The data preprocessing steps include:

1. Scaling: The closing prices are scaled to a range of 0 and 1 using MinMaxScaler.
2. Splitting: The scaled data is divided into training and testing sets.
3. Creating Sequences: The data is transformed into sequences of fixed length (time steps) to be used as input for the LSTM model.

Here's the code to preprocess the data:



```python
# Splitting data into training and testing sets
train_size = int(len(scaled_data) * 0.8)
train_data, test_data = scaled_data[:train_size], scaled_data[train_size:]
time_step = 100
X_train, y_train = create_dataset(train_data, time_step)
X_test, y_test = create_dataset(test_data, time_step)

# Reshaping input to be [samples, time steps, features] which is required for LSTM
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

Figure 6. Google Stock Data Preprocessing.
Source: personal documentation

C. Model Training

An LSTM model is constructed and trained on the historical stock data. The model is designed with multiple layes, including LSTM layers and dense layer, and is compiled using the mean squared error (MSE) loss function and the Adam optimizer. The model architecture consists of:

1. Input Layer: Accepts sequences of stock prices.
2. LSTM Layers: Two LSTM layers with 50 units each to capture temporal dependencies.
3. Dropout Layer: A dropout layer to prevent overfitting by randomly setting a fraction of input units to 0.
4. Dense Layer: Dense layers to map the LSTM outputs to the final prediction.

The model is trained for a specified number of epochs, with batch size set to 1 for fine-grained updates. The training process involves iteratively updating the model parameters to minimize the loss function on the training data.

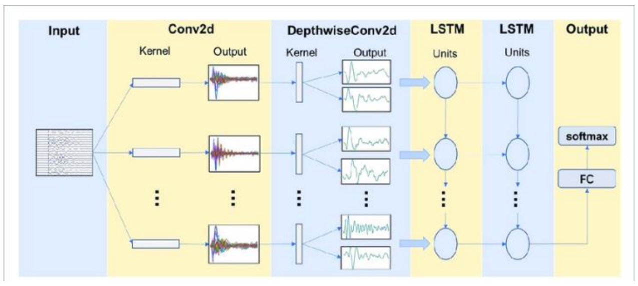
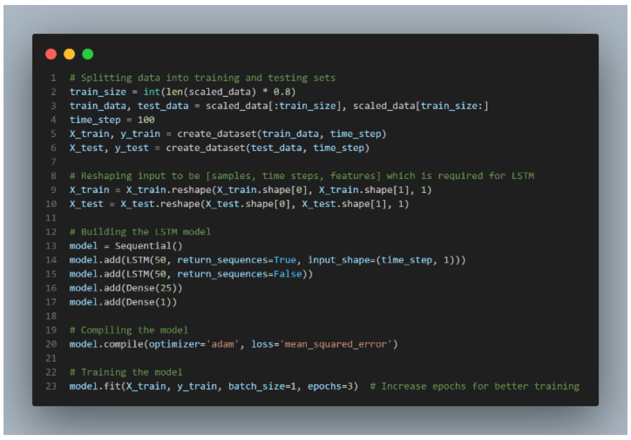


Figure 7. LSTM Layer.
Source: https://www.researchgate.net/figure/Overall-visualization-of-the-CNN-LSTM-model-architecture-Lines-represent-the-connections_fig1_361829455

Here's the code to train the model:



```python
# Splitting data into training and testing sets
train_size = int(len(scaled_data) * 0.8)
train_data, test_data = scaled_data[:train_size], scaled_data[train_size:]
time_step = 100
X_train, y_train = create_dataset(train_data, time_step)
X_test, y_test = create_dataset(test_data, time_step)

# Reshaping input to be [samples, time steps, features] which is required for LSTM
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

# Building the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(time_step, 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

# Compiling the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Training the model
model.fit(X_train, y_train, batch_size=1, epochs=3)  # Increase epochs for better training
```

Figure 8. Training Model Mechanism.
Source: personal documentation

D. Predictions and Optimization

The trained LSTM model is used to predict future stock prices, these predictions are then used as inputs for the dynamic programming algorithm to optimize investment decisions. The dynamic programming approach considers different states, including holding cash, holding stock, and deciding when to buy or sell stock to maximize returns.

The dynamic programming algorithm involves:

1. Defining States and Actions: States include holding cash,

holding stock, and holding value. Actions include buying stock, selling stock, and holding cash.

2. Computing Values: For each state and action, the algorithm computes the value based on the predicted stock prices and the possible outcomes of the actions.
3. Backtracking: The optimal sequence of actions is determined by backtracking from the final state to the initial state.

***Table 1**. Actions represented in number.*

Number	Action
0	Hold
1	Sell
2	Buy

As shown in the table above, each action is represented by a number: 0 for hold, 1 for sell, and 2 for buy. This numerical representation is used to simplify the creation of the path. By converting actions into numerical values, the process of tracking and optimizing the sequence of decisions becomes more efficient and manageable, facilitating the identification of the most profitable strategy over the given period.

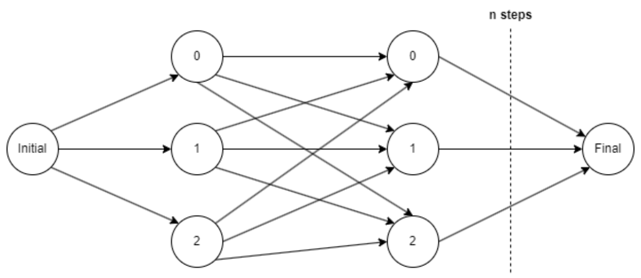


Figure 9. Dynamic Programming Prediction Illustration.
Source: personal documentation

The algorithm aims to maximize the final investment value by identifying the optimal strategy based on the predicted stock prices.

IV. Analysis and Result

A. Model Performance

The LSTM model's performance is evaluated using metrics such as Mean Squared Error (MSE) and visual comparisons of predicted and actual stock prices. The model shows promising results with relatively low error rates, indicating its effectiveness in predicting stock prices.

The performance evaluation includes:

1. Training Performance: Assessment of the model's accuracy on the training data.
2. Testing Performance: Evaluation of the model's accuracy on the training data.
3. Visualization: plots comparing predicted and actual stock prices to visually assess the model's performance.

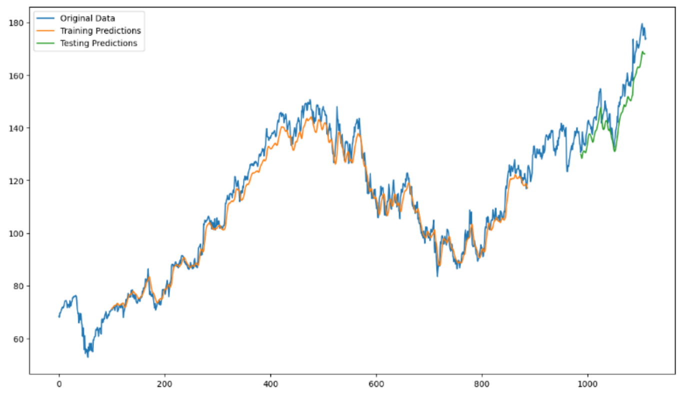


Figure 10. Real-Time Google Stock Data vs Prediction Result.
Source: personal documentation

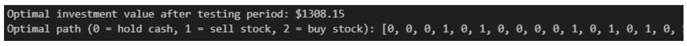
Based on the visualization, the model performs well, utilizing a 2-layer LSTM with 50 units per layer and 3 epochs. The model can predict prices with high accuracy, closely matching real-world data. This strong predictive performance contributes to the accuracy and reliability of the action determination process using the dynamic programming algorithm, ultimately leading to maximum profit.

B. Investment Strategy Optimization

The dynamic programming algorithm successfully identifies the optimal investment strategy based on the predicted stock prices. This strategy involves a series of buy, sell, and hold decisions aimed at maximizing the final investment value. The effectiveness of the optimized strategy is demonstrated through simulations and comparisons with baseline strategies.

The optimization results include:

1. Optimal Value: The maximum investment value achieved using optimal strategy.
2. Optimal Path: The sequence of actions (buy, sell, hold) that lead to the optimal value.
3. Comparison with Baselines: Comparison of the optimal strategy with simple buy-and-hold or random strategies.



```
Optimal investment value after testing period: $1308.15
Optimal path (0 = hold cash, 1 = sell stock, 2 = buy stock): [0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
```

Figure 11. Optimal Value and its Optimum Path.
Source: personal documentation

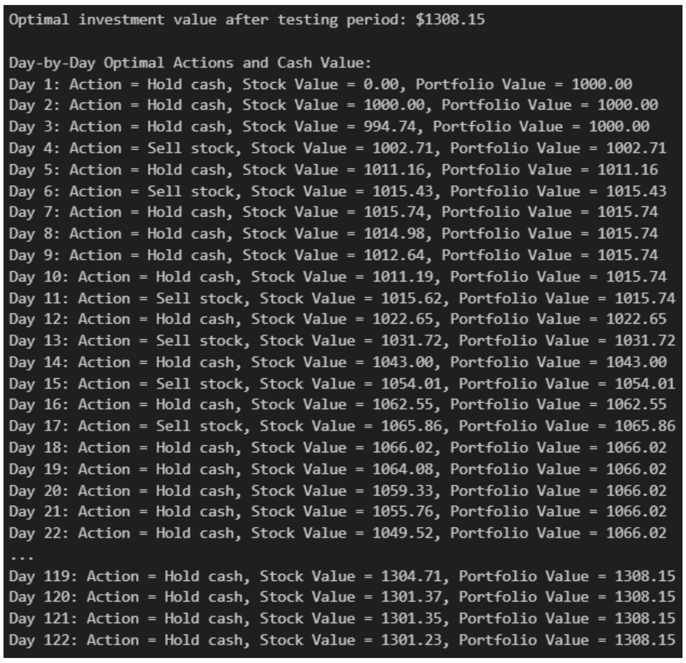
In this program, dynamic programming successfully determined the most optimal path to maximize profits. By treating each day as a layer/phase within the dynamic programming algorithm, the program selects nodes representing (sell, buy, and hold) actions to achieve the best possible profit. These nodes are aggregated until the most optimal path is identified.

C. Day-by-Day Optimal Actions and Cash Value

The optimal actions for each day are recorded, showing when to buy, hold, or sell stocks. These actions are then translated into portfolio values, illustrating the effectiveness of the optimized strategy. The day-by-day analysis provides detailed insights into the decision-making process and the resulting portfolio performance.

The day-by-day analysis includes:

1. Action Log: A detailed log of the actions taken each day based on the optimal strategy.
2. Cash value Tracking: Daily tracking of cash and stock values to monitor portfolio performance.
3. Portfolio Growth: Visualization of portfolio growth over time, highlighting the impact of the optimized strategy.



```
Optimal investment value after testing period: $1308.15

Day-by-Day Optimal Actions and Cash Value:
Day 1: Action = Hold cash, Stock Value = 0.00, Portfolio Value = 1000.00
Day 2: Action = Hold cash, Stock Value = 1000.00, Portfolio Value = 1000.00
Day 3: Action = Hold cash, Stock Value = 994.74, Portfolio Value = 1000.00
Day 4: Action = Sell stock, Stock Value = 1002.71, Portfolio Value = 1002.71
Day 5: Action = Hold cash, Stock Value = 1011.16, Portfolio Value = 1011.16
Day 6: Action = Sell stock, Stock Value = 1015.43, Portfolio Value = 1015.43
Day 7: Action = Hold cash, Stock Value = 1015.74, Portfolio Value = 1015.74
Day 8: Action = Hold cash, Stock Value = 1014.98, Portfolio Value = 1015.74
Day 9: Action = Hold cash, Stock Value = 1012.64, Portfolio Value = 1015.74
Day 10: Action = Hold cash, Stock Value = 1011.19, Portfolio Value = 1015.74
Day 11: Action = Sell stock, Stock Value = 1015.62, Portfolio Value = 1015.74
Day 12: Action = Hold cash, Stock Value = 1022.65, Portfolio Value = 1022.65
Day 13: Action = Sell stock, Stock Value = 1031.72, Portfolio Value = 1031.72
Day 14: Action = Hold cash, Stock Value = 1043.00, Portfolio Value = 1043.00
Day 15: Action = Sell stock, Stock Value = 1054.01, Portfolio Value = 1054.01
Day 16: Action = Hold cash, Stock Value = 1062.55, Portfolio Value = 1062.55
Day 17: Action = Sell stock, Stock Value = 1065.86, Portfolio Value = 1065.86
Day 18: Action = Hold cash, Stock Value = 1066.02, Portfolio Value = 1066.02
Day 19: Action = Hold cash, Stock Value = 1064.08, Portfolio Value = 1066.02
Day 20: Action = Hold cash, Stock Value = 1059.33, Portfolio Value = 1066.02
Day 21: Action = Hold cash, Stock Value = 1055.76, Portfolio Value = 1066.02
Day 22: Action = Hold cash, Stock Value = 1049.52, Portfolio Value = 1066.02
...
Day 119: Action = Hold cash, Stock Value = 1304.71, Portfolio Value = 1308.15
Day 120: Action = Hold cash, Stock Value = 1301.37, Portfolio Value = 1308.15
Day 121: Action = Hold cash, Stock Value = 1301.35, Portfolio Value = 1308.15
Day 122: Action = Hold cash, Stock Value = 1301.23, Portfolio Value = 1308.15
```

Figure 12. Day-by-Day Optimal Actions and Cash Value.
Source: personal documentation

As an example, an initial investment of $1000 will be allocated to Google (GOOG) stock. Utilizing an LSTM model, the prediction of whether the stock price will rise of fall the next day will be made. Subsequently, nodes representing sell, buy, and hold actions will be established. The dynamic programming will iterate over a span of 100 steps (representing days) to determine the most appropriate action (sell, buy, or hold) to minimize losses and maximize gains over the 100-day period. Additionally, the program will allocate the amount of funds or shares to be used for these actions with the aid of machine learning. Ultimately, the optimal path will be identified. In this specific case, an optimal value of $1308.15 was achieved.

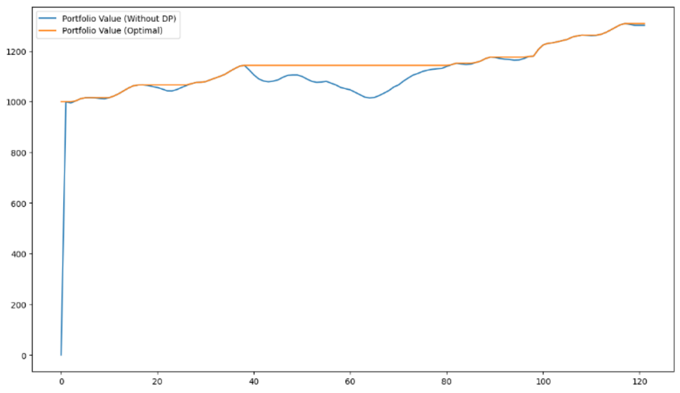


Figure 13. Portfolio Using DP vs Not Using DP.
Source: personal documentation

Based on the visualization above, it is evident that the invested capital remains optimal and does not decrease (the value tends to rise and does not incur losses). This is because the dynamic programming algorithm successfully makes precise decisions on when to sell, hold, and buy.

V. Conclusion and Suggestions

This paper demonstrates the potential of combining LSTM models with dynamic programming for stock market prediction and investment optimization. The integration of these two methodologies provides a robust framework for making informed and strategic investment decisions. The results indicate that the proposed approach can significantly enhance prediction accuracy and optimize investment strategies, leading to higher returns.

Future work can explore the application of this approach to other stocks and the incorporation of additional features to further enhance prediction accuracy and decision-making. Potential improvements include integrating sentiment analysis from financial news, incorporating macroeconomic indicators, and exploring advanced optimization techniques.

The key contributions of this paper include:

1. LSTM-Based Predictions: Demonstration of LSTM networks' effectiveness in predicting stock prices.
2. Dynamic Programming optimization: Novel application of dynamic programming for optimizing investment strategies based on predicted prices.
3. Comprehensive Framework: Development of a comprehensive framework that integrates prediction and optimization for stock market decision-making.

VI. Attachment

GitHub link to the source code and data used in the experiment: [Stock Analysis Using LSTM and Dynamic Programming]

VII. ACKNOWLEDGMENT

The author expresses gratitude to all parties who have assisted in the preparation of this paper, especially to:

1. The Almighty God, for His grace and guidance, allowing the smooth completion of this paper.
2. Both parents for providing both moral and material support to the author.
3. Extended family and friends who have encouraged and aided in the completion of this paper.
4. Ir. Rila Mandala, M.Eng., Ph.D., and Monterico Adrian, S.T., M.T., as the lecturers for the IF2211 Algorithm Strategy course, for kindly imparting additional knowledge and offering solutions to challenges encountered in writing this paper.
5. Friends in the BYTE cohort, especially those in the IF K3 class, who mutually supported each other during lectures and learned together throughout this semester.

May the Almighty God repay all the kindness that has been bestowed. May this paper serve as a contribution to the academic community.

REFERENCES

[1] geeksforgeeks, "Dynamic Programming - GeeksforGeeks," *GeeksforGeeks*, 2017. https://www.geeksforgeeks.org/dynamic-programming/

[2] R. Munir, "Program Dinamis (Dynamic Programming)." Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf

[3]
R. Munir, "Program Dinamis (Dynamic Programming)." Accessed: Jun. 12, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian2.pdf

[4] "Stock price using LSTM and its implementation," *Analytics Vidhya*, Dec. 06, 2021. https://www.analyticsvidhya.com/blog/2021/12/stock-price-prediction-using-lstm/

[5] "How to Scrape Yahoo Finance - 2024 Guide," *Bright Data*. https://brightdata.com/blog/how-tos/scrape-yahoo-finance-guide (accessed Jun. 12, 2024).

[6] R. Hamad, "What is LSTM? Introduction to Long Short-Term Memory," *Medium*, Dec. 11, 2023. https://medium.com/@rebeen.jaff/what-is-lstm-introduction-to-long-short-term-memory-66bd3855b9ce#:~:text=Long%20Short%2DTerm%20Memory%20(LSTM (accessed Jun. 12, 2024).

[7] R. Munir, "Graf (Bag.1) Bahan Kuliah IF2120 Matematika Diskrit," 2023. Accessed: Dec. 08, 2023. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/19-Graf-Bagian1-2023.pdf

STATEMENT

I hereby declare that the paper I have written is my own work, not a paraphrase or translation of someone else's paper, and it is not plagiarized.

Bandung, 12th June 2024

[illegible]

Rizqika Mulia Pratama
13522126