Multi-Attribute Stock Risk Screening Using Dynamic Programming on Historical Market Indicators

Daniel Anindito Nugroho - 13524002
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13524002@std.stei.itb.ac.id
danielnugroho900@gmail.com

***Abstract*—This paper examines a stock risk screening method for prioritizing candidate stocks under limited analysis capacity. Each candidate stock is represented by historical market indicators derived from price and volume data, including volatility, maximum drawdown, negative momentum, abnormal volume, and illiquidity. These indicators are normalized and combined into a composite risk score. The screening task is formulated as a 0/1 knapsack problem, where each stock is treated as an item, the risk score is the value, the analysis cost is the weight, and the analyst capacity is the knapsack capacity. Dynamic Programming is used to obtain an optimal subset of candidate stocks under the defined constraints. The method is evaluated under several analysis capacity scenarios to observe how the selected watchlist changes. The output is a prioritized watchlist for further risk analysis, not an investment recommendation or price prediction.**

***Index Terms*—Dynamic Programming, 0/1 Knapsack, stock risk screening, historical market indicators, risk score, watchlist prioritization**

I. Introduction

Analysts may need to review many candidate stocks while having limited time for manual analysis. In this situation, reviewing every available candidate with the same level of detail may be impractical. A screening mechanism can help prioritize which candidate stocks should be inspected further, especially when the objective is to identify stocks that show stronger historical risk signals.

This paper focuses on stock risk screening, not stock recommendation. The proposed method does not recommend stocks to buy or sell, does not predict future prices, and does not optimize a portfolio. Its output is a prioritized watchlist of candidate stocks for further risk analysis. This distinction is important because the method is designed as an algorithmic screening tool, not as an investment decision system.

A simple ranking by risk score may ignore the analysis cost of each stock. Under limited capacity, the highest individual scores do not always produce the highest total score when selected as a subset. Therefore, this paper models the screening task as a constrained subset selection problem. Each stock has a risk score, an analysis cost, and a selection decision. The analyst has a limited analysis capacity, and the goal is to select a subset of stocks with the maximum total risk score without exceeding this capacity.

The contributions of this paper are as follows. First, the stock risk screening task is formulated as a 0/1 knapsack problem. Second, a composite risk score is constructed from historical market indicators. Third, Dynamic Programming is used to select an optimal watchlist under a limited analysis capacity. Fourth, the method is evaluated using several capacity scenarios to observe how the selected subset changes under different constraints.

II. Theoretical Foundation

A. Dynamic Programming

Dynamic Programming is an algorithmic technique used to solve problems that have optimal substructure and overlapping subproblems [1]. A problem has optimal substructure when an optimal solution can be constructed from optimal solutions of smaller subproblems. It has overlapping subproblems when the same subproblems appear repeatedly during computation.

Instead of recomputing the same subproblems many times, Dynamic Programming stores intermediate results in a table. This property makes it suitable for constrained selection problems where each decision affects the remaining capacity. In this paper, Dynamic Programming is used to solve the stock screening problem after it is formulated as a 0/1 knapsack problem.

B. 0/1 Knapsack Problem

The 0/1 knapsack problem is a classical optimization problem in which each item can either be selected or not selected [2]. Each item has a value and a weight, and the objective is to maximize total value without exceeding a given capacity. This formulation fits the screening task because the analyst cannot inspect every candidate stock, and the selected subset must balance risk score and analysis effort under a limited capacity.

The mapping used in this paper is shown in Table I.

TABLE I
MAPPING OF 0/1 KNAPSACK COMPONENTS

Knapsack Component	Stock Screening Interpretation
Item	Candidate stock
Value	Risk score
Weight	Analysis cost
Capacity	Analysis capacity
Selected items	Prioritized watchlist

Let $dp[i][w]$ denote the maximum total risk score obtainable from the first i stocks with analysis capacity w. The base cases are

$$dp[0][w] = 0 \tag{1}$$

for all w, and

$$dp[i][0] = 0 \tag{2}$$

for all i.

If the analysis cost of stock i does not exceed the remaining capacity, the recurrence is

$$dp[i][w] = \max(dp[i-1][w], r_i + dp[i-1][w-c_i]), \tag{3}$$

where r_i is the risk score and c_i is the analysis cost of stock i. This transition is applied when $c_i \leq w$. Otherwise,

$$dp[i][w] = dp[i-1][w]. \tag{4}$$

The time complexity is $O(nW)$, where n is the number of candidate stocks and W is the analysis capacity. In this formulation, analysis cost and analysis capacity are represented as integer units of review effort so that capacity can be used as the Dynamic Programming table index. The risk score may be a decimal value because it is the value being maximized, not the capacity index.

C. Historical Market Indicators

The screening score uses five historical market indicators: volatility, maximum drawdown, negative momentum, abnormal volume, and illiquidity. These indicators are selected because they describe different historical risk signals from price and volume data. Volatility captures price fluctuation, maximum drawdown captures downside movement, negative momentum captures recent weakness, abnormal volume captures unusual trading activity, and illiquidity captures liquidity-related risk [3].

Historical return is not used as a separate indicator because direction of price movement is already represented by negative momentum and maximum drawdown.

III. METHODOLOGY

A. Overall Screening Framework

The proposed method follows a pipeline from historical market data to a prioritized watchlist. The input consists of daily OHLCV records for each candidate stock. The system computes historical indicators, normalizes the indicators, constructs a risk score, estimates analysis cost, and then applies Dynamic Programming to select the watchlist.

In sequence, the workflow consists of historical OHLCV data collection, indicator calculation, cross-sectional normalization, risk score calculation, analysis cost estimation, Dynamic Programming selection, and prioritized watchlist generation. The Dynamic Programming stage is the main algorithmic component because it determines which stocks are selected under the analysis capacity constraint.

B. Historical Indicator Calculation

The input data uses daily OHLCV records, consisting of date, open price, high price, low price, close price, and volume. The experiment uses data retrieved from Yahoo Finance through yfinance for candidate stocks from the combined manual LQ45 and IDX30 ticker lists.

The daily return is computed as

$$daily\_return_t = \frac{close_t - close_{t-1}}{close_{t-1}}. \tag{5}$$

Volatility is computed as the standard deviation of daily returns:

$$volatility = std(daily\_return). \tag{6}$$

Maximum drawdown is computed using the running maximum of close prices:

$$running\_max_t = \max(close_1, close_2, \ldots, close_t), \tag{7}$$

$$drawdown_t = \frac{close_t - running\_max_t}{running\_max_t}, \tag{8}$$

$$maximum\_drawdown = |\min(drawdown_t)|. \tag{9}$$

The 30-day momentum is computed as

$$momentum\_30d = \frac{latest\_close}{close\_30\_days\_ago} - 1. \tag{10}$$

The negative momentum component, denoted as s^-_{mom}, is then

$$s^-_{mom} = \max(0, -momentum\_30d). \tag{11}$$

Abnormal volume is computed as

$$abnormal\_volume = \frac{latest\_volume}{average\_volume\_20d}. \tag{12}$$

Liquidity is represented using average traded value:

$$liquidity = mean(close\_price \times volume). \tag{13}$$

The five indicators are interpreted as complementary historical risk signals. Volatility represents the degree of price fluctuation through the standard deviation of daily returns. Maximum drawdown captures the largest historical decline from a previous price peak, so a larger value indicates stronger downside movement. Negative momentum represents recent downward price movement over the 30-day lookback window. Abnormal volume compares the latest trading volume with the 20-day average volume to identify unusual trading activity. Illiquidity is derived by inverting the normalized liquidity value, so stocks with lower average traded value receive a higher liquidity-related risk component.

C. Normalization

The indicators have different scales, so they must be normalized before being combined into a single score. This paper uses min-max normalization:

$$normalized\_x = \frac{x - min_x}{max_x - min_x}. \tag{14}$$

Min-max normalization is applied cross-sectionally across all candidate stocks for the same indicator, not across time within a single stock. If all values of an indicator are equal, the normalized score can be set to 0 to avoid division by zero.

For liquidity, the normalized value is inverted to obtain an illiquidity score:

$$illiquidity\_score = 1 - normalized\_liquidity. \tag{15}$$

This conversion is used because lower liquidity corresponds to higher liquidity-related risk.

D. Risk Score Formulation

The risk score is a heuristic composite score. It is not an absolute financial risk measurement, but a comparable screening score used as the value in the knapsack formulation. The risk score may be represented as a decimal value because it is the value being maximized, not the capacity index.

The composite score is defined as

$$\begin{aligned} risk\_score = {} & 0.25 \cdot volatility\_score \\ & + 0.25 \cdot drawdown\_score \\ & + 0.20 \cdot negative\_momentum\_score \\ & + 0.15 \cdot abnormal\_volume\_score \\ & + 0.15 \cdot illiquidity\_score. \end{aligned} \tag{16}$$

The score is then scaled to 0–100:

$$risk\_score\_100 = risk\_score \times 100. \tag{17}$$

The weights give stronger emphasis to volatility and drawdown because both indicators directly describe price instability and downside movement. Momentum, abnormal volume, and illiquidity are included as supporting risk signals. The weight assignment is heuristic and is used to construct a comparable screening score, not an absolute financial risk measure.

E. Analysis Cost Formulation

The analysis cost represents the estimated effort needed to manually inspect a stock. It is not a monetary cost. It is used as the weight in the 0/1 knapsack formulation. In this paper, analysis cost is represented as an integer unit of review effort.

The analysis cost is defined as

$$analysis\_cost = 1 + number\_of\_triggered\_risk\_flags. \tag{18}$$

A risk flag is triggered when the normalized value of volatility, drawdown, abnormal volume, or illiquidity is above the 75th percentile among candidate stocks. For negative momentum, a flag is triggered when the raw momentum is negative and its normalized negative momentum score is above the 75th percentile.

This formulation reflects the assumption that a stock with more triggered risk flags may require more manual analysis because the analyst needs to inspect multiple risk aspects. Both analysis cost and analysis capacity are represented as integer units of review effort, allowing capacity to be used as the Dynamic Programming table index.

F. Dynamic Programming Formulation

The Dynamic Programming formulation uses $dp[i][w]$ as the maximum total risk score obtainable from the first i stocks with analysis capacity w:

$$dp[i][w]. \tag{19}$$

The decision for each stock is binary: either the stock is not selected, or it is selected if its analysis cost does not exceed the remaining capacity. The recurrence follows the 0/1 knapsack formulation:

$$dp[i][w] = \max(dp[i-1][w], r_i + dp[i-1][w - c_i]), \tag{20}$$

when $c_i \leq w$. Otherwise,

$$dp[i][w] = dp[i-1][w]. \tag{21}$$

After the DP table is filled, the selected stocks are obtained by tracing the table backward. If $dp[i][w]$ differs from $dp[i-1][w]$, stock i is included in the watchlist and the remaining capacity is reduced by its analysis cost. If two subsets produce the same total risk score, the implementation may choose the subset with lower total analysis cost. This tie-handling rule does not change the main Dynamic Programming formulation.

Stock/Cap.	0	1	2	3	4
0	0	0	0	0	0
1	0	0	40	40	40
2	0	30	40	70	70
3	0	30	60	90	100

Fig. 1. Illustration of the Dynamic Programming table for the 0/1 knapsack formulation.

IV. Implementation and Experiment

A. Dataset

The experiment uses the combined manual ticker lists of LQ45 and IDX30 as the candidate stock universe. The script setting requests 45 unique Yahoo Finance tickers with the Indonesian `.JK` suffix. The historical period is set from 2025-01-01 to 2026-01-01, where the end date is treated as exclusive by `yfinance`. In the latest run, the retained OHLCV records cover trading dates from 2025-01-02 to 2025-12-30.

Daily OHLCV data are obtained from Yahoo Finance through `yfinance` [4]–[6]. The data columns include date, open price, high price, low price, close price, and volume. From the 45 requested tickers, 38 tickers are retained after excluding tickers with unavailable, empty, or incomplete OHLCV records in the latest data-fetching run. Each retained ticker contains 236 daily records.

The filtering step is needed because every candidate stock must have enough historical records for daily return, 30-day momentum, 20-day average volume, and liquidity calculation. The dataset is used only for historical risk screening. It is not used to train a forecasting model or to generate buy or sell recommendations.

B. Program Structure

The experiment is implemented in Python using `yfinance` for data retrieval, `pandas` for tabular data processing, and `numpy` for numerical computation. The script is organized into separate functions for ticker selection, OHLCV retrieval, indicator calculation, normalization, risk score calculation, analysis cost calculation, Dynamic Programming selection, and simple ranking comparison. This separation keeps the data processing steps distinct from the algorithmic selection step.

The program input consists of the index setting, the historical date range, and the analysis capacity scenarios. The main configuration uses `INDEX_NAME = "BOTH"`, `START_DATE = "2025-01-01"`, `END_DATE = "2026-01-01"`, and capacities 5, 10, and 15. The program output consists of the selected watchlist, total risk score, total analysis cost, number of selected stocks, and detailed indicator values for the selected stocks.

The implementation script stores raw OHLCV files under `data/raw` and writes experiment outputs under `outputs`. The generated files include `indicator_summary.csv`, `dp_selected_stocks.csv`, `ranking_comparison.csv`, and one selected-detail CSV file for each analysis capacity scenario. These outputs are used to fill the experimental tables in this paper.

C. Data Retrieval and Preprocessing

The data retrieval function downloads daily OHLCV records for each ticker independently. This design allows the program to continue running even if one ticker fails to return valid data. After a ticker is downloaded, the program extracts the required OHLCV columns, removes rows with missing values in those columns, and stores the cleaned raw data as a CSV file. If a ticker returns empty data or incomplete OHLCV columns, the program prints a warning and excludes the ticker from the experiment.

Figure 2 should show the part of the implementation that calls `yfinance.download`, validates the OHLCV columns, and saves the raw data. This screenshot is included as implementation evidence, while the mathematical definitions of the indicators remain in the Methodology section.

D. Indicator, Score, and Cost Computation

After valid OHLCV data are collected, the program calculates volatility, maximum drawdown, 30-day momentum, abnormal volume, and liquidity for each ticker. The resulting indicator table is normalized cross-sectionally so that every score is comparable across the candidate stock universe. The composite risk score is then computed using the predefined indicator weights, and the analysis cost is computed from triggered risk flags.

```python
def fetch_ohlcv_data(
    tickers: list[str], start_date: str, end_date: str
) -> dict[str, pd.DataFrame]:
    """Fetch daily OHLCV data and save raw data per ticker."""
    os.makedirs(RAW_DATA_DIR, exist_ok=True)

    price_data: dict[str, pd.DataFrame] = {}

    for ticker in tickers:
        try:
            downloaded = yf.download(
                ticker,
                start=start_date,
                end=end_date,
                interval="1d",
                auto_adjust=False,
                progress=False,
                threads=False,
            )
        except Exception as exc:
            print(f"WARNING: failed to fetch {ticker}: {exc}")
            continue

        ohlcv = _extract_ohlcv_frame(downloaded, ticker)
        if ohlcv.empty:
            print(f"WARNING: {ticker} returned empty or incomplete OHLCV data.")
            continue

        ohlcv = ohlcv.dropna(subset=REQUIRED_OHLCV_COLUMNS)
        if ohlcv.empty:
            print(f"WARNING: {ticker} has no complete OHLCV rows after cleaning.")
            continue

        output_path = os.path.join(RAW_DATA_DIR, f"{ticker}.csv")
        ohlcv.to_csv(output_path, index=True)
        price_data[ticker] = ohlcv

    return price_data
```

Fig. 2. Data retrieval and OHLCV validation in the Python implementation.

The analysis cost calculation uses the 75th percentile threshold for normalized volatility, drawdown, abnormal volume, and illiquidity. For negative momentum, the flag is triggered only when the raw momentum is negative and the normalized negative momentum score is high enough. This rule keeps analysis cost as an integer unit of review effort and makes it suitable as the weight in the 0/1 knapsack formulation.

```python
def calculate_risk_score(indicators: pd.DataFrame) -> pd.DataFrame:
    """Calculate composite risk score used as the knapsack value."""
    scored = indicators.copy()

    scored["risk_score"] = (
        0.25 * scored["volatility_score"]
        + 0.25 * scored["drawdown_score"]
        + 0.20 * scored["negative_momentum_score"]
        + 0.15 * scored["abnormal_volume_score"]
        + 0.15 * scored["illiquidity_score"]
    )
    scored["risk_score_100"] = scored["risk_score"] * 100

    return scored
```

Fig. 3. Risk score in the Python implementation.

Figure 3 should show the implementation of the weighted risk score and the risk-flag based analysis cost. This screenshot helps connect the formulas in the Methodology section with the actual experimental pipeline.

E. Dynamic Programming Implementation

The Dynamic Programming implementation receives the processed candidate stock table and an analysis capacity value.

The risk score scaled to 0–100 is used as the value, while the integer analysis cost is used as the weight. The program allocates a table of size $(n + 1) \times (W + 1)$, where n is the number of valid candidate stocks and W is the analysis capacity.

For each stock and each capacity value, the program compares two alternatives: excluding the current stock or including it when its analysis cost does not exceed the remaining capacity. After the table is filled, the selected watchlist is reconstructed through backtracking from the last cell of the DP table. The backtracking step identifies which stocks contribute to the highest total risk score under the capacity constraint.

```python
def knapsack_dp(items: pd.DataFrame, capacity: int) -> dict[str, Any]:
    candidate_stocks = items.reset_index(drop=True).copy()
    values = candidate_stocks["risk_score_100"].astype(float).to_numpy()
    weights = candidate_stocks["analysis_cost"].astype(int).to_numpy()
    n_items = len(candidate_stocks)
    dp = np.zeros((n_items + 1, capacity + 1), dtype=float)

    for i in range(1, n_items + 1):
        value_i = values[i - 1]
        cost_i = weights[i - 1]
        for w in range(capacity + 1):
            if cost_i <= w:
                include_value = value_i + dp[i - 1][w - cost_i]
                exclude_value = dp[i - 1][w]
                dp[i][w] = max(exclude_value, include_value)
            else:
                dp[i][w] = dp[i - 1][w]

    selected_indices: list[int] = []
    remaining_capacity = capacity
    for i in range(n_items, 0, -1):
        if dp[i][remaining_capacity] > dp[i - 1][remaining_capacity] + 1e-9:
            selected_indices.append(i - 1)
            remaining_capacity -= weights[i - 1]

    selected_indices.reverse()
    selected_detail = candidate_stocks.iloc[selected_indices][
        [
            "ticker",
            "risk_score_100",
            "analysis_cost",
            "volatility",
            "maximum_drawdown",
            "momentum_30d",
            "abnormal_volume",
            "liquidity",
        ]
    ].copy()

    return _result_dict(capacity, "Dynamic Programming", selected_detail)
```

Fig. 4. Dynamic Programming implementation.

Figure 4 is the most important code screenshot because it shows the algorithmic core of the experiment. The screenshot should include the DP table initialization, the include-exclude transition, and the backtracking step used to recover the selected watchlist.

F. Experimental Scenarios

The experiment uses three analysis capacity scenarios: 5, 10, and 15. These scenarios represent low, medium, and high analysis capacity. They are used to observe how the selected watchlist changes when the available review effort increases. For each scenario, the program records the selected stocks, total risk score, total analysis cost, and number of selected stocks.

TABLE II
SELECTED STOCKS FOR DIFFERENT ANALYSIS CAPACITY VALUES

Cap.	Selected	Score	Cost	Count
5	BBTN.JK; GOTO.JK; ICBP.JK; INKP.JK; KLBF.JK	212.3005	5	5
10	BBTN.JK; BMRI.JK; GOTO.JK; ICBP.JK; INKP.JK; KLBF.JK; MAPI.JK; MEDC.JK; PTBA.JK	357.7607	10	9
15	BBTN.JK; BMRI.JK; BRIS.JK; EXCL.JK; GOTO.JK; ICBP.JK; INKP.JK; ITMG.JK; KLBF.JK; MAPI.JK; MEDC.JK; PTBA.JK; TLKM.JK	485.5968	15	13

G. Comparison Setup

A simple risk-score ranking is used only as a comparison reference, while the proposed selection method is solved using Dynamic Programming. The ranking reference sorts stocks by individual risk score and selects them in that order until the capacity is reached. This comparison is used to interpret the benefit of subset optimization, not to introduce another strategy algorithm.

The comparison is useful because ranking considers stocks one by one, while Dynamic Programming evaluates combinations of selected stocks under the same capacity constraint. Therefore, the comparison focuses on whether the selected subset has a higher total risk score under the defined score, cost, and capacity formulation. It does not compare investment quality or future market performance.

TABLE III
COMPARISON BETWEEN DYNAMIC PROGRAMMING AND SIMPLE RISK-SCORE RANKING

Cap.	Method	Selected	Score	Cost
5	Dynamic Programming	BBTN.JK; GOTO.JK; ICBP.JK; INKP.JK; KLBF.JK	212.3005	5
5	Simple ranking reference	MBMA.JK	78.3182	5
10	Dynamic Programming	BBTN.JK; BMRI.JK; GOTO.JK; ICBP.JK; INKP.JK; KLBF.JK; MAPI.JK; MEDC.JK; PTBA.JK	357.7607	10
10	Simple ranking reference	MBMA.JK; BRPT.JK; MAPI.JK	204.3073	10
15	Dynamic Programming	BBTN.JK; BMRI.JK; BRIS.JK; EXCL.JK; GOTO.JK; ICBP.JK; INKP.JK; ITMG.JK; KLBF.JK; MAPI.JK; MEDC.JK; PTBA.JK; TLKM.JK	485.5968	15
15	Simple ranking reference	MBMA.JK; BRPT.JK; EMTK.JK; MDKA.JK	280.1383	15

V. RESULT AND DISCUSSION

A. Dynamic Programming Selection Result

The selected stocks form the highest-scoring watchlist under the defined risk score, analysis cost, and capacity constraint.

Table II reports the Dynamic Programming result for each capacity scenario. With capacity 5, the selected watchlist contains 5 stocks with a total risk score of 212.3005. With capacity 10, the selected watchlist contains 9 stocks with a total risk score of 357.7607. With capacity 15, the selected watchlist contains 13 stocks with a total risk score of 485.5968.

The interpretation focuses on the relationship between score and cost. A stock with a high individual risk score may be excluded when its analysis cost is high and a combination of lower-cost stocks produces a higher total risk score under the same capacity. This behavior is consistent with the 0/1 knapsack formulation.

B. Effect of Analysis Capacity

The capacity scenarios show how the watchlist changes when the available analysis effort increases. The total risk score increases from 212.3005 at capacity 5 to 357.7607 at capacity 10, and then to 485.5968 at capacity 15. The number of selected stocks also increases from 5 to 9 and then to 13. This pattern indicates that larger analysis capacity allows the Dynamic Programming solution to include more candidate stocks while still respecting the capacity constraint.

Capacity	5	10	15
Score	212.3005	357.7607	485.5968

Fig. 5. Total risk score obtained by Dynamic Programming under different analysis capacity values.

C. Discussion Against Simple Ranking

The comparison with simple risk-score ranking is used to clarify why subset optimization matters. A simple ranking may select stocks with high individual risk scores without considering how much capacity they consume. Dynamic Programming, on the other hand, evaluates combinations of selected and unselected stocks under the same capacity constraint.

The claim should remain within the mathematical formulation: Dynamic Programming gives the optimal subset under the defined value, cost, and capacity constraints. It should not be interpreted as producing better investment decisions.

D. Limitations

This study has several limitations. First, the risk score depends on the selected indicators and their heuristic weights. Second, the analysis cost is an estimated effort, not a measured manual review time. Third, historical data do not guarantee future stock behavior. Fourth, the output is not an investment recommendation. Finally, the experiment uses a limited candidate set, so the result should be interpreted within the selected stock universe and historical period.

VI. Conclusion

This paper presents a Dynamic Programming approach for stock risk screening under limited analysis capacity. The screening task is formulated as a 0/1 knapsack problem, where candidate stocks are items, risk scores are values, analysis costs are weights, and analyst capacity is the knapsack capacity. Historical market indicators are used to construct a composite risk score, while analysis cost is estimated from triggered risk flags.

The method produces a prioritized watchlist for further risk analysis under the defined score, cost, and capacity constraints. It does not provide buy or sell recommendations, does not predict future prices, and does not optimize a portfolio. Future work may explore adaptive weighting for risk indicators, a larger candidate stock set, different historical periods, or additional non-price indicators while keeping the Dynamic Programming formulation as the main algorithmic component.

Attachment

Github Repo : https://github.com/le1n-gif/Makalah_Stima_13524002

Acknowledgment

First and foremost, I would like to thank Jesus for his blessings and grace, which enabled me to complete this paper. I would also like to express my sincere gratitude to Ir. Rila Mandala, M.Eng., Ph.D. and Dr. Ir. Rinaldi Munir, M.T. as for their guidance and the valuable knowledge shared during the Informatics Engineering class.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

[2] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. Chichester, U.K.: John Wiley & Sons, 1990.

[3] M. Magdon-Ismail and A. F. Atiya, "Maximum drawdown," *Risk*, vol. 17, no. 10, pp. 99–102, 2004.

[4] Yahoo Finance, "Historical market data," 2026. [Online]. Available: https://finance.yahoo.com/

[5] R. Aroussi, "yfinance documentation," 2026. [Online]. Available: https://ranaroussi.github.io/yfinance/

[6] R. Aroussi, "yfinance API reference," 2026. [Online]. Available: https://ranaroussi.github.io/yfinance/reference/index.html

Declaration of Originality

I hereby declare that this paper is my own original work, not an adaptation or translation of another person's paper, and is free from plagiarism.

Bandung, June 18, 2026

Daniel Anindito Nugroho
13524002