Candlestick Pattern Detection Using String Matching Algorithms with Symbolic Representation

Helena Kristela Sarhawa - 13524109
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: hkristela03@gmail.com , 13524109@std.stei.itb.ac.id

***Abstract*—In the capital market, a candlestick chart is commonly used in technical analysis. The chart is used to observe price movements and transaction volume. A candlestick chart can visually summarize the opening price, highest price, lowest price, and closing price. From previous studies, candlestick chart patterns have been attempted to be detected using image-based deep learning methods such as YOLO. However, in those experiments, visual pattern identification still required training data and complex computational resources. Therefore, this paper proposes a simpler approach by converting candlestick charts into sequences of symbols that will be matched using string matching algorithms.**

Keywords—candlestick; string matching algorithms; symbolic representation

I. INTRODUCTION

Technical analysis is one method in capital market analysis used to monitor price movements through price information, volume, and open interest. Through candlestick chart representation, price information is shown in a form similar to bars or candles that can form certain patterns. The patterns obtained from the shape and position of candlesticks can be used for analysis and prediction related to market conditions and the direction of price movements.

Technical analysis performed manually without the help of a program has several limitations and weaknesses. Each person's visual assessment can be subjective, inconsistent, and vulnerable to differences in individual perception. Therefore, technical analysis with the help of a program can be a solution to overcome several of these limitations.

Before this paper was written, there had already been studies that attempted to detect candlestick patterns automatically. These studies utilized computer vision and deep learning such as YOLOv5 and YOLOv8. These studies proved effective in detecting patterns and overcoming the limitations of manual analysis. However, the approach used in those studies depends on image datasets, labeling, training configuration, and evaluation metrics that are closer to object detection. Therefore, this paper will use a different method and perspective. In this paper, candlestick chart data will be transformed into sequences of symbols arranged based on time order. Each available candlestick will be converted into a symbol according to body size, direction, and wick dominance. After being converted into a sequence of symbols, candlestick pattern detection will be carried out through string matching algorithms.

The purpose of this paper is to design a simple candlestick pattern detection method, implement string matching algorithms in the context of the capital market, and compare the performance among algorithms in detecting candlestick patterns through symbolic representation. The string matching algorithms compared are Brute Force, Knuth-Morris-Pratt, and Boyer-Moore. The comparison focuses more on execution time, the number of comparisons performed, and the number of matches. This study does not produce recommendations related to buying and selling, but the candlestick patterns obtained from the program in this study can be used for analysis related to the capital market.

II. THEORETICAL FOUNDATION

A. Candlestick Chart and OHLC Data

A candlestick describes price movement within a time interval using four numerical values, namely open, high, low, and close. The body of a candlestick is formed from the open and close prices, while the upper and lower wicks of a candlestick describe the distance from the candlestick to the highest and lowest prices. To detect patterns from candlesticks, the process usually begins with OHLC data (Open, High, Low, Close) and converts it into a chart image for visual pattern detection. However, in this paper, OHLC data will be converted into a sequence of symbols.

Fig. 1. Example of a Candlestick Chart used in Technical Analysis (source: https://www.freepressjournal.in/business/how-to-read-a-candlestick-chart)

B. Candlestick Pattern Detection

Previous studies have shown that candlestick pattern detection can be carried out using deep learning methods. These studies successfully detected double top and double bottom patterns using YOLOv5. From these studies, it was concluded that the more training data used, the higher the detection accuracy level, even reaching an accuracy level of 75.9%. Other studies have also used YOLOv8 with 4,435 candlestick images and obtained precision, recall, F1-score, and mAP values for candlestick object detection. In this paper, a simpler method using a symbolic representation approach will be applied to detect candlestick patterns. With this approach, image labeling and other training are not required, making it simpler.

C. String (Pattern) Matching

String or pattern matching is a method for finding a string or pattern within a collection of text. The requirement for performing string matching is that the length of the source text must be longer than the string or pattern to be searched. String matching can be applied in searches in Text Editors, website search engines, image analysis, bioinformatics, and others. In this paper, string matching is applied by treating the sequence of symbols resulting from candlestick transformation as the source text and the candlestick pattern as the pattern to be found in the source text.

In strings, there are the concepts of prefix and suffix. For example, a string of size m is symbolized as $S = x_0x_1\ldots x_{m-1}$. The prefix of S is the substring S[0..k], while the suffix of S is the substring S[k..m-1], where k is any index between 0 and m-1.

D. Brute Force Algorithm

The simplest string matching algorithm is the Brute Force algorithm. This algorithm checks every position in the source text one by one from left to right to match whether the pattern starts at that position. In the worst case, the number of character comparisons performed by this algorithm is m(n-m+1) or O(mn). The advantage of the Brute Force algorithm is that it is simple, but this algorithm can have a very large number of character comparisons, making it less efficient, especially when applied to a small alphabet size or when partial matches occur frequently.

E. Knuth-Morris-Pratt Algorithm

The Knuth-Morris-Pratt algorithm, or KMP, is a string matching algorithm developed by Donald Knuth, Vaughan Pratt, and James Morris. This algorithm is designed to improve the efficiency of the search process as an improvement over the simple Brute Force algorithm. The KMP algorithm processes the pattern first through preprocessing into a border function or failure function. When a mismatch occurs in matching the pattern with the source text, KMP uses information from the preprocessing stage to avoid repeating comparisons on pattern characters that have already matched the source text. The computation of the border function requires O(m) time, and string searching requires O(n) time, so the total time complexity of the KMP algorithm is O(m+n).

F. Boyer-Moore Algorithm

The Boyer-Moore algorithm, or BM, is a string matching algorithm that has a matching mechanism quite different from other algorithms. The BM algorithm matches the pattern from right to left using the looking-glass technique. When a mismatch occurs, two heuristics are used. First, using the bad-character rule, the pattern is shifted so that the character in the source text that was last declared mismatched is aligned with its last occurrence in the pattern. Second, using the good-suffix table, the pattern undergoes preprocessing to determine its longest border. The border referred to is finding the longest prefix that is the same as the suffix of the pattern used. Therefore, when a mismatch occurs after part of the end of the pattern has already matched, comparisons will not be performed from the beginning, but the pattern will be shifted so that the pattern prefix occupies the suffix position from the previous iteration.

From these two heuristics, the shift value actually used is taken from the maximum value between the two heuristics. With this method, several positions can be skipped directly, making this algorithm very efficient. In the worst case, the BM algorithm requires O(nm + A) time. A large value of A will make Boyer-Moore work faster.

III. Proposed Method

A. Overall Workflow

The program created in this paper consists of five main stages, namely data loading, symbolic encoding, pattern definition, string matching, and evaluation. First, OHLC data is loaded from a CSV file. Second, each row will be transformed into a one-character symbol. Third, the user selects a predefined pattern or a raw pattern. Fourth, the program applies the Brute Force, KMP, or Boyer-Moore algorithm to find all occurrences of the pattern. Finally, the program records the number of matches, the number of character comparisons, and the execution time of each algorithm.

B. Symbolic Representation of Candlesticks

In the symbolic representation of candlesticks, information about morphology such as candle range, body size, upper wick, and lower wick is still maintained, and the program will produce a simple string. Each candle will be represented by exactly one symbol. The use of one symbol for each candle makes the sequence of symbols processable by string matching algorithms.

Table I. Symbolic Representation of Candlesticks

Symbol	***Meaning***	***Summary of Rules***
G	Strong bullish candle	Close > Open and body ratio ≥ strong threshold
g	Weak bullish candle	Close > Open and body ratio below strong threshold
R	Strong bearish candle	Close < Open and body ratio ≥ strong threshold
r	Weak bearish candle	Close < Open and body ratio below strong threshold
D	Doji or neutral candle	Body ratio ≤ doji threshold
U	Long upper wick candle	Upper wick dominates the candle range

Symbol	*Meaning*	*Summary of Rules*
L	Long lower wick candle	Lower wick dominates the candle range

The body ratio is calculated as the absolute value of Close minus Open divided by High minus Low. In categorizing symbols, the translator or converter from candle to symbol will first check whether the upper or lower wick dominates the candle range more. If the lower wick is more dominant, then the candle is categorized as "L". If the upper wick is more dominant, then the candle is categorized as "U". Wick dominance checking is performed first because a candle can have a very small body with a very long wick, so the wick shape will better describe the characteristics of the candle. After that, if no wick is more dominant, a candle with a very small body can be categorized as "D" or Doji. If the candle does not meet the conditions above, the translator will use the direction and body ratio to categorize the candle as a strong or weak bullish or bearish movement.

C. Pattern Definition

The program in this paper is designed to accept predefined candlestick patterns as well as special or custom patterns. The predefined patterns have already been translated into symbolic sequences. The symbolic translations of these patterns will be used by the program as patterns whose occurrences will be searched in the OHLC data. For example, the symbolic sequence RG describes a strong bearish candle followed by a strong bullish candle.

Table II. Predefined Candlestick Patterns

Pattern Name	*Symbolic Pattern*	*Interpretation*
bullish_reversal	RG	Strong bearish followed by strong bullish
bearish_reversal	GR	Strong bullish followed by strong bearish
doji_continuation	DgG	Doji followed by bullish continuation
upper_rejection	UgR	Long upper wick followed by bearish pressure
lower_rejection	LrG	Long lower wick followed by bullish pressure
mixed_volatility	GrRg	Alternating strong and weak movements
morning_star	RDG	Strong bearish candle followed by doji and strong bullish candle, a three candle bullish reversal signal
evening_star	GDR	Strong bullish candle followed by doji and strong bearish candle, a three candle bearish reversal signal
three_white_soldiers	GGG	Three consecutive strong bullish candles appearing after a downtrend, indicating a bullish reversal signal
three_black_crows	RRR	Three consecutive strong bearish candles appearing after an uptrend, indicating a bearish reversal signal
hammer	LG	Long lower wick followed by a strong bullish confirmation candle as a reversal signal
shooting_star	UR	Long upper wick followed by a bearish pressure confirmation candle as a reversal signal

Note: In the hammer and shooting_star patterns, an adaptation is applied as two candles from the one-candle definition (Bulkowski, 2008) by adding a confirmation candle to support multicharacter pattern matching.

IV. IMPLEMENTATION DESIGN

The program code implementation in this paper is written using the Python language. The reason for choosing the Python programming language is that Python provides features that are easy to use for processing CSV, numerical computation, and visualization. This program is built from several main modules or files to separate the functionality of data loading, candle categorization into symbols, string matching algorithm implementation, application of algorithms to find matches, and visualization. The program code created will be uploaded to GitHub, whose link will be attached at the end of the paper.

Table III. Program Files and Their Functions

File	*Function*
src/encoder.py	Translates rows of OHLC data into a sequence of candlestick symbols
src/brute_force.py	Implements the Brute Force string matching algorithm and counts the number of character comparisons performed
src/kmp.py	Implements the Knuth-Morris-Pratt string matching algorithm and performs longest prefix suffix preprocessing
src/boyer_moore.py	Implements the Boyer-Moore string matching algorithm with bad character and good suffix heuristics
src/data_loader.py	Loads CSV data and generates synthetic OHLC data
src/patterns.py	Stores predefined symbolic candlestick patterns
src/experiment.py	Performs string matching using all implemented algorithms with OHLC data as the source text, and the results will be exported to results/benchmark.csv
src/visualizer.py	Generates a close price chart and marks the time range where the pattern is found with a shaded area
src/main.py	Provides a Command Line Interface based interface for testing one combination of pattern and algorithm selected by the user
src/make_charts.py	Generates charts comparing the performance of the Brute Force, KMP, and Boyer-Moore algorithms

V. EXPERIMENT SETUP

A. Dataset

The program created in this paper supports two types of data sources, namely CSV files containing real OHLC data and synthetic OHLC data automatically generated by the program. In the study conducted, the experiment will be applied to synthetic data. Synthetic data is created deterministically using a fixed seed so that the experimental results can still be viewed and used by other users without needing to prepare real capital market data first.

B. *Evaluation Metrics*

The evaluation of the experiment conducted uses three main types of metrics. First, match count measures the number of pattern occurrences found in the source text. Second, comparison count measures the number of character comparisons performed by each algorithm. Third, execution time measures the execution time required for each algorithm using high-resolution time recording, averaged from 10 repetitions for each text and pattern pair to reduce noise from the operating system. These three metrics are used in this program because this paper aims to compare efficiency among algorithms, not to produce market price predictions.

C. *Experiment Scenarios*

The experiments conducted in the program are designed with several scenarios to compare performance among algorithms. The first scenario is varying the text length by using different numbers of candles. The second scenario is varying the pattern length. The third scenario is using symbolic patterns that have been defined in pattern.py. The fourth scenario is using patterns that are deliberately created and impossible to appear in the data to evaluate the behavior of the three algorithms under no-match conditions.

Table IV. Experiment Scenarios

Scenario	***Variable***	***Purpose***
Dataset Size	100, 500, 1000, 5000 candles	Measuring scalability as the value of n or the source text length increases
Pattern length	2,3,5,8 characters	Measuring the impact of pattern length m
Predefined Patterns	RG, GR, DgG, UgR, LrG, GrRg, RDG, GDR, GGG, RRR, LG, UR	Testing predefined symbolic patterns
Custom Pattern (Absent)	GRGRGRGR, RRRRRRRRR, GGGGGGGGG	Measuring algorithm behavior when the pattern is not found at all in the source text

VI. RESULTS

The experiment was conducted using synthetic OHLC data consisting of 5000 candles transformed into a sequence of symbols with a length of 5000 characters. All 93 rows of experimental results are stored in results/benchmark.csv, covering four testing scenarios, namely scalability against source text length, the effect of pattern length, detection of 12 predefined patterns, and scenarios where the pattern is not found.

A. *Scalability Results*

The first test measures how the number of character comparisons increases as the source text length increases (n = 100, 500, 1000, 5000 candles) with a fixed pattern length (m = 3).

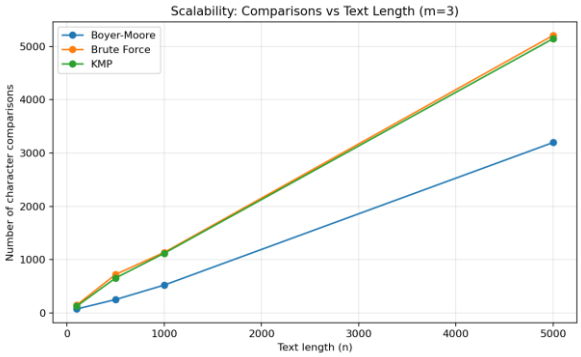

Fig. 2. Scalability Comparison based on Source Text Length

Table V. Scalability Results based on Source Text Length

n	***Brute Force***	***KMP***	***Boyer-Moore***
100	150	129	76
500	725	657	253
1000	1137	1121	524
5000	5205	5144	3197

The three algorithms show growth that is close to linear with respect to n. However, Boyer-Moore consistently performs fewer comparisons than Brute Force and KMP at every n size.

B. *Effect of Pattern Length*

The second test varies the pattern length (m = 2, 3, 5, 8) at a fixed text length (n = 5000) to observe how each algorithm processes longer patterns.

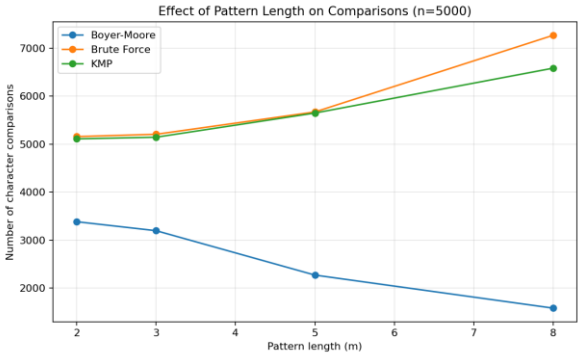

Fig. 3. Effect of Pattern Length on Character Comparisons

Table VI. Effect of Pattern Length on Character Comparisons

m	***Brute Force***	***KMP***	***Boyer-Moore***
2	5157	5109	3384
3	5205	5144	3197
5	5670	5647	2273
8	7265	6581	1585

From the results obtained, Brute Force and KMP require more character comparisons as the pattern length increases. Conversely, Boyer-Moore appears to become more efficient as the pattern length increases. This result is consistent with the characteristics of the Boyer-Moore algorithm, namely that the longer the pattern, the greater the potential to skip many positions at once.

C. *Detection of 12 Predefined Patterns*

The third test measures the ability of the three algorithms to detect 12 candlestick patterns that have been defined in the synthetic OHLC source data with a length of 5000 candles.

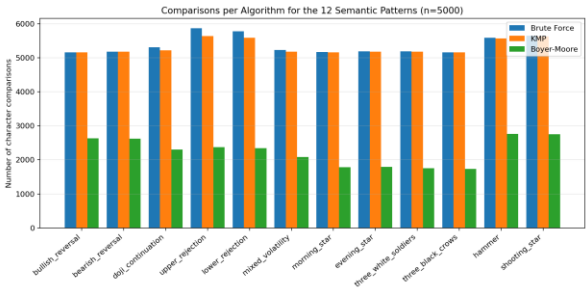

Fig. 4. Algorithm Comparison for Predefined Candlestick Patterns

Table VII. Detection Results for Predefined Candlestick Patterns

Pattern Name	*Symbol*	*Match Count*	*BF*	*KMP*	*BM*
bullish_reversal	RG	2	5157	5156	2633
bearish_reversal	GR	6	5181	5176	2619
doji_continuation	DgG	4	5305	5220	2307
upper_rejection	UgR	11	5866	5637	2372
lower_rejection	LrG	0	5778	5588	2344
mixed_volatility	GrRg	0	5228	5182	2085
morning_star	RDG	0	5166	5158	1787
evening_star	GDR	1	5193	5181	1795
three_white_soldiers	GGG	0	5190	5182	1756
three_black_crows	RRR	0	5157	5158	1738
hammer	LG	23	5587	5565	2764
shooting_star	UR	17	5648	5632	2755

The three algorithms successfully produce identical match counts for each pattern. In terms of efficiency, Boyer-Moore appears superior in searching all 12 patterns, with an average reduction in the number of character comparisons of around 50-70% compared with Brute Force and KMP. Some patterns are also not found in the synthetic data. This is natural because the synthetic data is created through a random walk simulation without engineering specific patterns.

D. *Pattern Not Found Scenario (Absent Pattern)*

The fourth test measures the behavior of the three algorithms when a no-match condition occurs, using patterns that are impossible to appear in the data and deliberately created such as GRGRGRGR, RRRRRRRRR, and GGGGGGGGG.

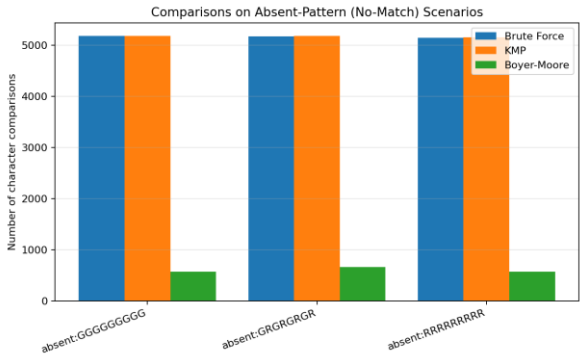

Fig. 5. Algorithm Comparison in Absent Pattern Scenarios

Table VIII. Comparison Results for Absent Pattern Scenarios

Absent Pattern	*Brute Force*	*KMP*	*Boyer-Moore*
GRGRGRGR	5180	5182	668
RRRRRRRRR	5150	5158	575
GGGGGGGGG	5183	5182	571

With this scenario, the advantage of Boyer-Moore is most visible. Brute Force and KMP still require around 5000 comparisons to search for a pattern that does not appear even once in the source text because both algorithms must check almost all positions in the text. Meanwhile, Boyer-Moore only requires 571 – 668 comparisons. This is because shifts occur that skip several positions at once by using the two implemented heuristics. Its mechanism of comparing from right to left also helps Boyer-Moore immediately conclude mismatch conditions without having to check the entire pattern.

E. *Overall Statistical Summary*

Table IX. Overall Statistical Summary of Algorithm Comparisons

Algorithm	*Total*	*Average*	*Minimum*	*Maximum*
Boyer-Moore	42263	1363,3	43	3384
KMP	109580	3534,8	121	6581
Brute Force	111493	3596,5	136	7265

From the statistical summary shown in the table above, Boyer-Moore requires 62.1% fewer total character comparisons than Brute Force and 61.4% fewer total character comparisons than KMP. Meanwhile, KMP is only slightly more efficient than Brute Force, with a total character comparison difference of 1.7%.

In terms of execution time averaged from 10 iterations per text and pattern pair, KMP records the fastest average time at 0.31 ms, followed by Brute Force at 0.36 ms and Boyer-Moore at 0.44 ms. These results show that there is a difference between the efficiency of the number of character comparisons and execution time efficiency. Boyer-Moore requires slightly longer time than the other two algorithms because Boyer-Moore needs to perform preprocessing to process the bad character and good suffix heuristics.

VII. Analysis

A. Brute Force

Based on theory, Brute Force has a worst-case time complexity of O(mn). However, in text with rare partial matches, Brute Force behavior is closer to O(n). To test this, the ratio of the number of comparisons to text length will be calculated in the table below.

Table X. Brute Force Comparison Ratio by Source Text Length

n	*Comparisons*	*Ratio (comparisons/n)*
100	150	1,50
500	725	1,45
1000	1137	1,14
5000	5205	1,04

As n becomes larger, the obtained ratio tends to decrease and approach 1.0. This shows that Brute Force has O(n) complexity for the average case. Because the symbolic representation in this program has a fairly high variation of symbols, the worst-case O(mn) condition will rarely be reached.

B. Knuth-Morris-Pratt

The KMP algorithm is designed to avoid repeating comparisons of pattern characters that have already matched using the border function. Based on theory, this algorithm has a time complexity of O(m+n). Theoretically, KMP will be superior to Brute Force especially for patterns that have internal repetition. However, based on the results obtained from the experiment, KMP only saves 1.7% of the total comparisons when compared with Brute Force overall.

Table XI. KMP Savings Compared to Brute Force

Scenario	*KMP Savings Compared to Brute Force*
Variation of text or pattern length	4,1%
12 predefined patterns	1,0%
Absent pattern	-0,1% (slightly worse)

The minimal savings made by KMP are caused by the nature of the border function, namely that KMP will be superior if the pattern used has a long internal border, so the pattern can jump further when a mismatch occurs. Meanwhile, the patterns used in the capital market context do not have repetitive internal structures. As a result, the border function of most patterns is zero or close to zero, causing KMP behavior to be almost identical to Brute Force. Therefore, the advantage of KMP is only conditional on the pattern structure even though its time complexity is O(m+n)

C. Boyer-Moore

Based on theory, the time complexity of the Boyer-Moore algorithm is O(nm + A). From the results obtained from the experiment, Boyer-Moore is able to save comparisons very efficiently compared with Brute Force. The largest savings made by Boyer-Moore are clearly visible in the absent pattern scenario. This is consistent with the bad character theory, namely that when a character in the text never appears in the pattern, the pattern will be shifted by the length of the pattern itself. Therefore, in the fourth scenario, most positions in the source text can be skipped directly without performing comparisons.

Table XII. Boyer-Moore Comparison Ratio by Pattern Length

m	*Comparisons*	*Ratio (comparisons/n)*
2	3384	0,677
3	3197	0,639
5	2273	0,455
8	1585	0,317

Based on the ratio values attached in the table, as the pattern length increases, the obtained ratio consistently decreases. This decrease in ratio is in line with the good suffix theory, namely that longer patterns allow further jumps when a mismatch occurs.

VIII. Conclusion

The program created in this paper successfully implements a simple candlestick pattern detection system by converting synthetic OHLC data into a sequence of symbols. This program also successfully implements three types of string matching algorithms, namely Brute Force, Knuth-Morris-Pratt, and Boyer-Moore, and applies the three algorithms to search for predefined candlestick patterns in OHLC data. The experimental results show that the number of matches obtained by the three algorithms is exactly the same. The difference found lies in the efficiency of the number of character comparisons and the execution time of each algorithm. Based on the number of character comparisons, Boyer-Moore works very efficiently. Meanwhile, based on execution time, KMP records the fastest time and Boyer-Moore records the longest time. This paper does not produce recommendations for buying and selling in the capital market, but only shows that string matching algorithms can be used to detect candlestick patterns that have been transformed into sequences of symbols. Further development can be carried out by using real capital market data as the dataset source.

Video Link at Youtube

https://youtu.be/3gRnyeVQEVA

APPENDIX

The full source code is available at https://github.com/helenakristela/Candlestick-Pattern-Detection-Using-String-Matching-Algorithms-with-Symbolic-Representation

ACKNOWLEDGMENT

Upon the completion of this paper, the author extends sincere appreciation to:

1. God Almighty, for His blessings and guidance throughout the writing process;
2. Dr. Rinaldi Munir, M.T., lecturer of the K1 IF2211 Strategi Algoritma course, for his valuable guidance and knowledge; and
3. The author's parents, for their unwavering support and encouragement.

REFERENCES

[1] G. A. Prasetia and I. Atastina, "Pendeteksi Pola Candlestick Chart Pada Saham Menggunakan Algoritma YOLO," JITLIT, vol. 1, no. 1, Jun. 2024.

[2] M. Manshur, O. Nurdiawan, A. R. Dikananda, and Fathurrohman, "Deteksi Pola Candlestick Menggunakan YOLOv8 untuk Analisis Teknikal Berbasis Citra," Integrative Perspectives of Social and Science Journal, vol. 3, no. 4, 2026.

[3] R. Munir, "Pencocokan String (String/Pattern Matching)," Bahan Kuliah IF2211 Strategi Algoritma, Program Studi Teknik Informatika, STEI ITB, 2026.

[4] Thomas N. Bulkowski, Encyclopedia of Candlestick Charts, Hoboken, NJ: John Wiley & Sons, 2008. ISBN 978-0470182017.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Juni 2026

Helena Kristela Sarhawa
13524109