An Analysis of a Branch and Bound Approach with Apriori Algorithm in Finding Frequent Item Sets for an E-Commerce Recommender System

Enrique Alifio Ditya - 13521142
Informatics Engineering Major
Electrical and Informatics Engineering Department
Bandung Institute of Technology, Bandung, West Java
alifioditya@gmail.com

***Abstract*—This paper aims to analyze the use of the branch and bound method in the Apriori algorithm to efficiently find frequent item sets for e-commerce recommender systems. A dataset of user clickstream in an e-commerce website is analyzed and applied an unsupervised machine learning technique with Apriori algorithm to find the relations between items and generate useful frequent item sets. A recommendations system can then be based upon the information collected, potentially improving the overall user experience while searching the website.**

Keywords—Branch and Bound; Apriori; E-Commerce; Recommendations System; Machine Learning.

I. INTRODUCTION

In this modern world, the environment in which transactions are conducted has been revolutionized by digitalization through the E-Commerce industry. The rise of online shopping has led to a significant increase in the number of e-commerce websites, making it easier for customers to search for their needs and purchase products or services online. The global E-Commerce market is estimated to be one of the most profitable, as it is projected to reach 1,356.88 billion USD in revenue by 2025 in the US alone [1]. As the data in the e-commerce industry grows exponentially, demands of reliable personalized recommendations systems have arisen.

One approach is to build a data driven system by analyzing frequent patterns in customer transactions. This can be done through applying machine learning algorithms to predict items that might be of interest based on user behavioral data. An effective method for achieving this is market basket analysis, which involves identifying items that are frequently purchased together. In order to attain these frequent item sets, a form of unsupervised machine learning such as Apriori is used.

Apriori is a popular algorithm for Association Rule Learning to find frequent item sets in large datasets. It is based upon a branch and bound algorithm in which the algorithm generates candidate item sets and prunes infrequent ones. The algorithm first scans the dataset to identify frequent single items, then it iteratively generates larger item sets and prunes them based on their frequency. This process continues until no new frequent item sets can be found.

These data-driven approaches are increasingly popular in e-commerce and have been shown to improve customer engagement and sales. A study by McKinsey & Company found that personalization based on customer data can increase sales by up to 15% [2]. Additionally, according to a survey by Accenture, 91% of consumers are more likely to buy from companies who make personalized offers and suggestions [3]. E-commerce companies are now investing heavily in data analysis and machine learning to develop recommendation systems that provide a personalized shopping experience for customers. In this context, algorithms such as Apriori are becoming increasingly important for finding frequent item sets and generating recommendations based on customer behavior. In this paper, we analyze the effectiveness of a branch and bound approach with Apriori algorithm for finding frequent item sets in an e-commerce recommender system.

II. FUNDAMENTAL THEORY

A. Branch and Bound

The branch and bound algorithm is a general optimization technique that is used to solve combinatorial problems. The basic idea behind this algorithm is to systematically search through all possible solutions while keeping track of the best solution found so far, and to prune branches of the search tree that are guaranteed to not contain an optimal solution.

The general steps for the branch and bound algorithm are as follows:

1. Initialization: Set up the initial search space and define the initial upper bound on the objective function.
2. Branching: Divide the search space into smaller subproblems and choose one of the subproblems to explore.
3. Lower bounding: Compute a lower bound on the objective function for the current subproblem.
4. Pruning: If the lower bound on the current subproblem is greater than the current upper bound, then prune the current subproblem and move back up the search tree. Otherwise, continue to explore the current subproblem.
5. Termination: Stop the search when all subproblems have been explored or when a satisfactory solution has been found.

In the context of the Apriori algorithm, the branch and bound technique is used to prune search space and speed up the process of identifying frequent item sets. The algorithm identifies all frequent item sets in a dataset by using a candidate generation process to incrementally build itemset of increasing size, and then pruning those that do not meet the minimum support threshold. By using branch and bound, the algorithm can eliminate many of the candidate item sets early in the process, making it more efficient.

B. *Association Rule Learning*

Association Rule Learning is a data mining technique that is used to discover underlying patterns and relationships between variables in a set of data by identifying items that frequently occur together. Association Rule Learning requires transactional data where transactions consist of sets of items purchased or viewed by a customer. The main premise of association rule learning in market basket analysis is to conclude, "if a customer buys product A, they are likely to also purchase product B". Commonly used metrics in association rule learning are:

1. Support: Measures the frequency with which the itemset appears in the dataset. It is calculated as the ratio of the number of transactions containing the itemset to the total number of transactions.

$$Support(X) = \frac{(\#\ transactions\ containing\ X)}{(total\ number\ of\ transactions)}$$

Note: '#' denotes "Number of"

2. Confidence: Measures the likelihood that an itemset Y will be purchased when itemset X is purchased. It is calculated as the ratio of the number of transactions containing both X and Y to the number of transactions containing X.

$$Confidence(X \rightarrow Y) = \frac{(\#\ transactions\ containing\ both\ X\ and\ Y)}{(\#\ transactions\ containing\ X)}$$

3. Lift: Measures the extent to which the occurrence of one itemset is dependent on the occurrence of another itemset. It is calculated as the ratio of the observed support of both item sets to the expected support if they were independent of each other.

$$Lift(X \rightarrow Y) = \frac{Support(X \cup Y)}{Support(X) \times Support(y)}$$

C. *Apriori*

Apriori is a bottom-up approach that discovers frequent item sets by identifying the support of each item in the dataset and then iteratively generating larger and larger candidate item sets based on a minimum support threshold. The basic idea behind Apriori is that if an itemset is frequent, then all its subsets must also be frequent. For example, if {A, B} occurs frequently, then both {A} and {B} must also occur frequently. This property is known as the "Apriori principle" and forms the basis for the algorithm.

The Apriori algorithm works in two main phases:

1. Generation of Frequent Item Sets: In the first phase, the algorithm scans the dataset to determine the support of each item in the dataset. It then generates candidate item sets of length two by combining frequent 1-item sets. The support of each candidate itemset is then computed and those that meet the minimum support threshold are retained as frequent 2-itemsets. This process is repeated to generate frequent k-item sets until no more frequent item sets can be generated. The following are the step by step of the first phase:

   a. Initialize the minimum support threshold.

   b. While the number of frequent item sets is greater than 0:

   c. Generate candidate item sets of length k from the frequent item sets of length k-1.

   d. Count the support for each candidate.

   e. Prune each candidate that is below the minimum support threshold.

   f. Repeat the process until no more frequent itemset can be found.

   g. Return the frequent item sets.

2. Generation of Association Rules: In the second phase, association rules are generated from the frequent item sets by finding all possible non-empty subsets of each frequent item set and computing the confidence of each rule. Rules with a confidence greater than or equal to a minimum confidence threshold are retained as strong association rules. The following are the step by step of the second phase:

   a. Initialize the confidence threshold.

   b. For each frequent item set X generated by phase one:

   c. Generate all possible non-empty subsets of X.

   d. For each potential antecedent A, calculate the confidence of the rule A → (X-A). The consequent (X-A) is the set of items in X that are not in A.

   e. Prune each candidate that is below the minimum confidence threshold.

   f. Repeat the process for all frequent item sets.

   g. Return the association rules.

III. Implementation

In this section, implementation of Apriori algorithm towards E-Commerce clickstream data will be conducted. An analysis will also be presented to evaluate the usability of the information collected and the performance of the algorithm. This section aims to answer the following questions:

1. What does the data look like?

2. Which steps are taken to prepare the data for Association Rule Learning?
3. How does the Apriori algorithm work in implementation?

A. *Tools and Environment Specification*

The tools, both hardware and software, used to implement Association Rule Learning are specified as the following:

1. Hardware
    a. Machine: Dell XPS 9320
    b. Processor: Intel Evo i7
    c. Core: 16
    d. Threads: 12
    e. RAM: 32GB
2. Programming Language
    a. Python
    b. Jupyter Notebook
3. Library
    a. Pandas
    b. Numpy
    c. Matplotlib
    d. Mlxtend

B. *Dataset*

In this paper, two E-Commerce clickstream data are used to analyze user behavioral pattern on viewing and conducting purchases. The datasets used for this analysis are:

1. Retail Rocket Recommender System Dataset: This dataset consists of user behavioral data collected from a real-world e-commerce website. The behavior data consists of clickstream events classified as views, add to cart, or transactions that were collected over the period of 4.5 months. The dataset itself contains 2756101 rows for each clickstream, with 1407580 unique customers and only 11719 (0.83%) conducting a purchase. This dataset will be mainly used to analyze user viewing patterns as transactions only occur less than 1% of the time. Hence, user viewing patterns can be recognized through Association Rule Learning. From this point on forwards, this dataset will be referred to as the first dataset. A sample of this dataset is as presented on Table I.

TABLE I. USER VIEWING SAMPLE DATA

timestamp	visitorid	event	itemid
1433221332117	257597	view	355908
1433224214164	992329	view	248676

2. UK Retailer E-Commerce Dataset: This dataset retrieved from UCI Machine Learning Repository consists of transactional data over the course of a year (2010-2011) from a UK Retailer. Each row is a single item transaction consisting of the invoice number, stock code, date, product description, quantity, unit price, customer id, and the country of the customer. This dataset will be used to analyze user purchases and to find association rules to recommend items of interest for identified item sets. From this point on forwards, this dataset will be referred to as the second dataset. A sample of this dataset is as presented on Table II.

TABLE II. USER PURCHASES SAMPLE DATA

InvoiceNo	StockCode	Description	InvoiceDate	CustomerID
536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	12/1/2010 8:26	17850
536365	71053	WHITE METAL LANTERN	12/1/2010 8:26	17850

C. *Data Preparation*

Prior to training the Apriori algorithm on both transactional data, proper steps of preprocessing need to be conducted to prepare the data in the correct format. A reduction of the search space is also done to simplify the analysis.

The first dataset has the “timestamp” feature containing the time a viewing event occurred in a UNIX time format. This is then converted into a more human readable form in the form of standard datetime format (YYYY-MM-DD). After the conversion, an aggregation is done to group the viewings data by visitor ID and further grouped by viewings within a one-week time window. This is done to collect all the viewings data for each unique user in the given period. A user may have more than one row of data if the viewings are separated for more than one week apart. The time window of one week is chosen as a user may have different interests after one week of scouring the website, which is purely assumptive.

After processing the first dataset, the second dataset is also aggregated. Each row is grouped by InvoiceNo representing a single purchase. This is done to collect each item purchased at a time.

After properly grouping the viewings and purchases data, one of its columns should already be in a transactional format (“itemid” for the first dataset, “description” for the second dataset). This can then be extracted and converted to a list of lists containing different transactions that occurred in each dataset.

At this point, the first dataset contains around 1.5 million transactions while the second contains roughly 22.000 transactions. To simplify the analysis, the first dataset is truncated to only contain the same number of rows as the second dataset. This is also done to avoid over allocation of memory caused by combinatorial explosion in both phases of the Apriori algorithm. A divide and conquer technique could be applied to solve this problem by training the first dataset in batches, but for the sake of time and simplicity, this paper will only focus on generating association rules based on the first 22.000 rows of viewing data.

Before feeding the Apriori algorithm with both the transactional data, it needs to be encoded with Transaction

Encoding, which works similarly to One-Hot Encoding by creating binary columns for each singular items and marking it as true if a transaction contains it, false if it does not. With the datasets encoded, it is now in a form ready to be processed for frequent item set mining.

D. Training the Apriori Algorithm

Referring to the explanation of the Apriori algorithm in section IIC, the model works in two phases. Phase one to identify the frequent item sets, and phase two to identify the association rules. In phase one, the algorithm traverses the search space similar to a breadth-first search, with the cost function being the support of the current item set, and the bounding function is determined by the minimum threshold. The searching stops once there are no item sets left that meet the minimum support threshold. The solution node in the tree search space is the leaf node that represents the itemset which satisfies the minimum support threshold. There can be multiple solution nodes in the tree search space. Each solution node corresponds to a frequent itemset that meets the minimum support threshold. The following provides the Apriori phase one implementation in pseudocode:

```
input
D: a list of transactions where each transaction is a list of items
minSup: a minimum support threshold for an item to be considered frequent
output
L1: Dictionary that holds the frequent 1-itemsets
function
frequent_1_itemsets(D, minSup):
    L1 = {}
    for transaction in D:
        for item in transaction:
            if item not in L1:
                L1[item] = 0
            L1[item] += 1
    // Filter out items with support count less than minSup
    L1 = {k:v for k,v in L1.items() if v >= minSup}
    return L1
```

```
input
L: a list of frequent itemsets from the previous level represented as a list of items.
output
Ck: candidate itemsets
function
apriori_gen(L):
    Ck = {}
    for i in range(len(L)):
        for j in range(i+1, len(L)):
         // If the first k-1 items in
         // itemset i are equal to the first
         // k-1 items in itemset j
            if L[i][:-1] == L[j][:-1]:
            // Create a candidate itemset
            // by combining itemsets i and j
                Ck_item = L[i] + [L[j][-1]]
            // If any subset of Ck_item is
            // not in L, skip to next
            // iteration
                if has_infrequent_subset(Ck_item, L):
                    continue
                Ck.add(Ck_item)
    return Ck
```

```
input
L: a list of frequent itemsets represented as a list of items.
Ck: candidate itemsets
output
flag: True if subset Ck it not in L, false otherwise
function
has_infrequent_subset(Ck_item, L):
    for item in Ck_item:
        subset = Ck_item - [item]
        if subset not in L:
            return True
    return False
```

```
input
D: a list of transactions where each transaction is a list of items
minSup: a minimum support threshold for an item to be considered frequent
```

```
output
frequentItemsets: all frequent itemsets found in the search space
function
PhaseOneApriori(D, minSup):
    L1 = frequent_1_itemsets(D, minSup)
    L = L1
    k = 2
    while L is not empty:
        // Generate candidates
        Ck = apriori_gen(L)
        for transaction in D:
            for candidate in Ck:
                if candidate is subset of transaction:
                    candidate.support += 1
        Lk = {}
        for candidate in Ck:
            // Pruning step
            if candidate.support >= minSup:
                Lk.add(candidate)
        L = Lk
        k += 1
    return all frequent itemsets found
```

Phase one of the Apriori algorithm successfully generated over 1622 frequent item sets for the first dataset using a minimum support of 0.00025 and 856 for the second dataset using a minimum support of 0.01. It should be noted that the minimum supports were first selected arbitrarily and then further refined to generate a sufficient amount of frequent item sets.

Phase two Apriori is focused on generating association rules from the frequent item sets obtained in phase one. The process involves iterating over the frequent item sets generated in phase one and generating all possible non-empty subsets of each frequent itemset. These subsets are referred to as the antecedent of the association rule. The remaining items in the frequent itemset that are not part of the antecedent form the consequent of the association rule. Similar to phase one, the searching is done in a breadth-first manner, with the algorithm exploring different combinations of antecedents and consequents to generate potential association rules. For each frequent itemset, the algorithm calculates various metrics such as support, confidence, and lift to evaluate the strength of the association rule. During the search process, the algorithm prunes association rules that do not meet user-defined minimum support and confidence thresholds. The following is the pseudocode implementation for phase two Apriori:

```
input
itemset: a set of items
size: size of the subset to be extracted
output
subsets: subset of the itemset
function
generate_subsets(itemset, size):
if size == 1:
    for item in itemset:
        subsets.append([item])
else:
    for i in range(len(itemset) - size + 1):
        current_item = itemset[i]
        remaining_items = itemset[i+1:]
        subsubsets = generate_subsets(remaining_items, size - 1)

        for subsubset in subsubsets:
            subsets.append([current_item] + subsubset)

return subsets
```

```
input
frequentItemsets: a list of frequent itemsets generated from phase one
minSup: a minimum support threshold for an item to be considered frequent
minCon: a minimum confidence threshold for an association rule to be considered significant
output
associationRules: a list of association rules
function
PhaseTwoApriori(frequentItemsets, minSup, minCon):
    associationRules = {}
    for itemset in frequentItemsets:
        for i in range(1, len(itemset)):
            antecedents =
generate_subsets(itemset, i)
            consequents =
generate_subsets(itemset, len(itemset) - i)
            for antecedent in antecedents:
                for consequent in
consequents:
                    rule = antecedent +
consequent
                    support =
calculate_support(rule)
                    confidence =
calculate_confidence(rule)
                    lift =
calculate_lift(rule)
                    if support >= minSup and
confidence >= minCon:
associationRules.add((antecedent,
consequent, support, confidence, lift))


return associationRules
```

After training the Apriori algorithm on the given dataset, the frequent item sets and association rules generated will have been collected and may infer commonalities in viewing or purchase patterns of customers.

IV. Analysis

This section aims to answer business-oriented questions from the application of Apriori algorithm to both transactional data such as the following:

1. What are the frequent items viewed together?
2. What are the frequent items bought together?
3. What can be recommended to a customer that views an item with the highest support?
4. What can be recommended to a customer that bought an item with the highest support?
5. How well does the Apriori algorithm perform in finding frequent item sets with the given minimum support threshold?

The Apriori algorithm implemented in this paper is supported by the mlxtend library. However, this disables the ability to do white box testing and see the internal workings of the Apriori algorithm, therefore a case from one of the datasets will be taken to demonstrate how the Apriori algorithm traverses the search space and prunes each subtree that does not meet the minimum support threshold.

A. *Frequent Item Sets*

The Apriori algorithm successfully generated over 1622. frequent item sets on the first dataset with a minimum support of 0.00025 and 856 on the second dataset with a minimum support of 0.01. Among them, the most frequent item has the support of 0.001742 on the first dataset and 0.090594 on the second dataset. The following can be concluded from frequent item sets analysis:

1. Item with ID of 187946 has the most support of viewings in the first dataset.
2. Item “WHITE HANGING HEART T-LIGHT HOLDER” has the highest support among purchase data in the second dataset.
3. Apriori took 25 seconds to process the first dataset, iterating over 60.000 combinations.
4. Apriori took 1 minute and 4 seconds to process the second dataset, iterating over 200.000 combinations.
5. All frequent item sets found in the first dataset are of length one, meaning the item categories are either too sparse or not correlated with each other.
6. Over 700 frequent item sets are found in the second dataset, with 220 having length of more than one. This means that some recommendations can be made for purchases of select items.

B. *Generated Association Rules*

After frequent item sets are found, association rules can be generated to collect behavioral pattern of customers. The following are the information inferred from the generated association rules:

1. The first dataset does not contain any significant association rules with a minimum threshold of 0.01.
2. Over 560 association rules are found for the second dataset, with four having lengths two or more in the consequent side of the rule.
3. The rule “GREEN REGENCY TEACUP AND SAUCER” → “ROSES REGENCY TEACUP AND SAUCER” is the most prominent, having a confidence metric of 0.76 with 20.22 lift.
4. Association rules generation only took 0.1 seconds, which is quicker than frequent item sets searching in a factor of 640.

C. *Example*

Suppose an item “ALARM CLOCK BAKELIKE GREEN” from the second dataset. In the purchase data, it is found that the item is involved in 806 out of 22187 transactions. This yields a support score of 0.03632. Given a minimum support threshold of 0.01, the Apriori algorithm would have saved this item as a frequent-1-item set and continued the searching for its child node. Suppose another item “ALARM CLOCK BAKELIKE RED” which is contained in 904 transactions. This gives a support score of 0.04074, which is also above the minimum threshold. Out of 806 transactions containing “ALARM CLOCK BAKELIKE GREEN” and 904 transactions containing “ALARM CLOCK BAKELIKE RED”, 533 of which contain both items at the same time, yielding a joint support score of 0.02. Therefore, this will still be considered as a frequent itemset and is a candidate for an association rule.

Say the Apriori tree searches for an itemset that contains "ALARM CLOCK BAKELIKE GREEN", "ALARM CLOCK BAKELIKE RED", and "JUMBO BAG RED RETROSPOT". There are only 63 transactions that contain all three items at the same time, giving a support score of 0.00284. Considering that the support is lower than the given threshold of 0.01, the itemset is pruned from the search tree and will not be considered as a frequent itemset.

In phase two, the itemset "ALARM CLOCK BAKELIKE GREEN" and "ALARM CLOCK BAKELIKE RED" is checked and calculated for the confidence metric. Given the minimum confidence of 0.1, the rule "ALARM CLOCK BAKELIKE GREEN" → "ALARM CLOCK BAKELIKE RED" is considered as a strong enough association rule with 0.66129 confidence. The reverse "ALARM CLOCK BAKELIKE RED" → "ALARM CLOCK BAKELIKE GREEN" is also considered a valid association rule, with calculated confidence metric of 0.5896. With this in mind, a customer that purchased "ALARM CLOCK BAKELIKE GREEN" may be recommended an "ALARM CLOCK BAKELIKE RED" product and vice versa, with an argument that both products are analyzed to be frequently purchased together.

V. Conclusion

The Apriori algorithm is a powerful branch and bound approach used to discover frequent item sets from transactional data. By analyzing patterns of item occurrences in relation to each other, the algorithm enables the extraction of valuable information regarding customer behavior and preferences.

In its process, the Apriori algorithm goes through two phases. One to search for item sets that occur frequently together and another to generate association rules with a given minimum confidence. These two phases implement a form of branch and bound method in order to prune the leaf nodes that do not meet the minimum required threshold.

Through a case study using purchase data, several important findings were revealed. In the first dataset, item 187946 exhibited the highest support of viewings, indicating its popularity among customers. However, no significant association rules were discovered, suggesting a lack of strong relationships between items in this dataset.

In the second dataset, the item "WHITE HANGING HEART T-LIGHT HOLDER" emerged as the most supported item among purchase data. The Apriori algorithm performed well in processing both datasets, taking 25 seconds and 1 minute and 4 seconds, respectively. Over 700 frequent item sets were found in the second dataset, with 220 of them having a length greater than one, indicating potential recommendations for purchasing specific items.

Overall, this analysis demonstrates the effectiveness of the Branch and Bound approach with the Apriori algorithm in uncovering frequent item sets and association rules for an E-Commerce Recommender System. The findings provide valuable insights into customer preferences and potential recommendations for enhancing the user experience and driving sales in e-commerce. Future research can further explore these findings and leverage them to optimize recommendation systems and marketing strategies in the e-commerce industry.

Video Link at YouTube

The following video contains explanation of the implementation in code for this paper: https://www.youtube.com/watch?v=qUBqc54OIw4

Acknowledgment

I would like to begin by acknowledging God for His blessings and guidance throughout this research endeavor. I am also immensely grateful to Dr. Nur Ulfa Maulidevi, S.T, M.Sc. for her invaluable mentorship, and expertise in the field of algorithm strategies and machine learning.

Furthermore, I would also like to extend my appreciation to all the researchers and professionals in the field of machine learning whose contributions have paved the way for advancements in this domain. Their work provided valuable insights and inspiration for this study and helped to lay the groundwork for the approach and methods used in this work.

References

[1] Statista Research Department. (2023). *US Retail E-Commerce Sales Forecast*. Retrieved from https://www.statista.com/statistics/272391/us-retail-e-commerce-sales-forecast/

[2] McKinsey and Company. (2021). *The value of getting personalization right—or wrong—is multiplying*. Retrieved from https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/the-value-of-getting-personalization-right-or-wrong-is-multiplying

[3] Accenture. (2018). *Personalization Pulse Check*. Retrieved from https://www.accenture.com/_acnmedia/pdf-77/accenture-pulse-survey.pdf

[4] de Ponteves, H., & Eremenko, K. (2023). Machine Learning A-Z™: AI, Python & R [Online course]. Udemy. https://www.udemy.com/share/101Wci/

[5] Roman Zykov, Noskov Artem, &amp; Anokhin Alexander. (2022). *Retailrocket recommender system dataset* [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/4471234

[6] Chen, D., Sain, S. L., & Guo, K. (2012*). Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining* [Data set]. Journal of Database Marketing and Customer Strategy Management, *19*(3), 197-208. https://doi.org/10.1057/dbm.2012.17

[7] Agrawal, R., Imielinski, T., Swami, A., Road, H., & Jose, S. (1993). *Mining association rules between sets of items in large databases*. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (pp. 207-261). Washington, DC, USA.

[8] Hikmawati, E., Maulidevi, N. U., & Surendro, K. (2020). *Adaptive rule: A novel framework for recommender system.* ICT Express, *6*(3), 214-219. https://doi.org/10.1016/j.icte.2020.06.001

Appendix

The code implementation used in this paper can be seen and retrieved on my Kaggle notebook: https://www.kaggle.com/code/alifioditya/pattern-analysis-for-recommender-system/notebook?scriptVersionId=130439480

Declaration of Originality

I, the undersigned below, the Author of this paper, hereby declare that this paper is my own writing, not an adaptation or translation of someone else's paper, and not plagiarized.

Bandung, 22 May 2023

[illegible]

Enrique Alifio Ditya 13521142