Multi-threaded approach in associative classification

Nguyen Quoc Huy[1], Tran Anh Tuan[1], and Do Nhu Tai[2]

[1] Saigon University, Vietnam
[2] University of Economic HCMC, Vietnam
nqhuy@sgu.edu.vn

Abstract. Associative classification is one interesting approach of supervised learning that are easily explainable. This approach is often constructed based on both classification and association rule mining techniques, to find out a set of rules called the classification association rules (CAR), to classify target attribute. There are many methods to improve the model following the feature selection approach and achieve quite good results. This article focuses on improving running time in multi-threaded approach. There are many experiments performed on many different kinds of dataset, and there are very reasonable results when we could explain which types of datasets are suitable for multi-threaded implementation as well as parallelization. Based on that, we can choose the appropriate number of threads for each type of dataset.

Keywords: Associative Classification · Feature Selection · Multi-threaded.

1 Introduction

Association rules are specific types of rules that drive associations between attributes. They are mainly used in pattern mining, where they could extract frequent itemsets from transaction data. The association classifier is a supervised learning model that uses association rules to assign labels. The model can therefore be viewed as a list of “if-then” statements: if a new data satisfies the attributes on the left of the rule, then that data will be classified according to the value on the right of rule.

The association classifier inherits several metrics from the association rules, such as Support or Confidence, that can be used to rank or filter the rules in the model and evaluate their quality. There are many different associative classification methods such as CBA, CMAR, CPAR.

CBA uses techniques in association rules to classify data, this method has higher performance than traditional classification techniques. The limitation of this method is that the number of generated rules is too large in case the minsupp is low.

CMAR applies FP-trees efficiently, consumes less memory and space than the CBA method. However, the limitation of FP-tree occurs if the data has a large number of attributes and the memory capacity is not enough to store it.

CPAR is a type of association classifier based on predicted association rules. This method combines the advantages of hybrid classification and traditional rule-based classification by generating rules directly during the training phase to avoid missing important rules.

Table 1. The improvement of feature selection approach in mushroom

	CPAR	CBA	CMAR
Accuracy	0.5429/0.9951	0.9828/0.9397	0.8012/0.8816
Rules	55/9	22/8	453/14
Time(ms)	1120/169	945343/146	1500/200

Table 2. The improvement of feature selection approach in mailspam

	CPAR	CBA	CMAR
Accuracy	NA/0.9305	NA/0.848	NA/0.6916
Rules	NA/278	NA/184	NA/158
Time(ms)	NA/13292	NA/146	NA/200

Because data is growing exponentially in every domain. Extracting useful information from this data is becoming a challenge as large numbers of rules are generated, as well as combinatorial complexity. There are two main approaches for improvement: (1) pre-processing data to select the minimum set of rules but gain the highest classification accuracy. (2) processing data in a non-sequential way such as multi-thread, parallel methods.

The first approach has had many recent publications such as [1, 6, 7, 8], and has many encouraging results on many different types of dataset. Tables 1 and 2 show the improvement of CAR models towards feature selection. The models are improved in terms of accuracy, number of rules, and execution time. Table 1 is the model comparison values between before and after improvement on Mushroom data, which is the data that original CAR algorithms often use. Table 2 is the model comparison values before improvement and after improvement on Spammail data, this is relatively complex data that original CAR algorithms could not perform (with NA value).

This paper tries to make some assumptions to predict what types of data might be effective for multi-threaded processing methods (see figure 1). Because the non-sequential solution is only suitable for some kinds of data, not for all kinds. The disadvantage of multi-threading is that each thread requires its own stack, which increases memory costs. Additionally, frequently context switching between threads can lead to increased CPU usage, reducing overall efficiency.

This paper has 4 parts, part one introduces the main problem of the paper and introduces related work, part two presents the Multi-threaded algorithm for

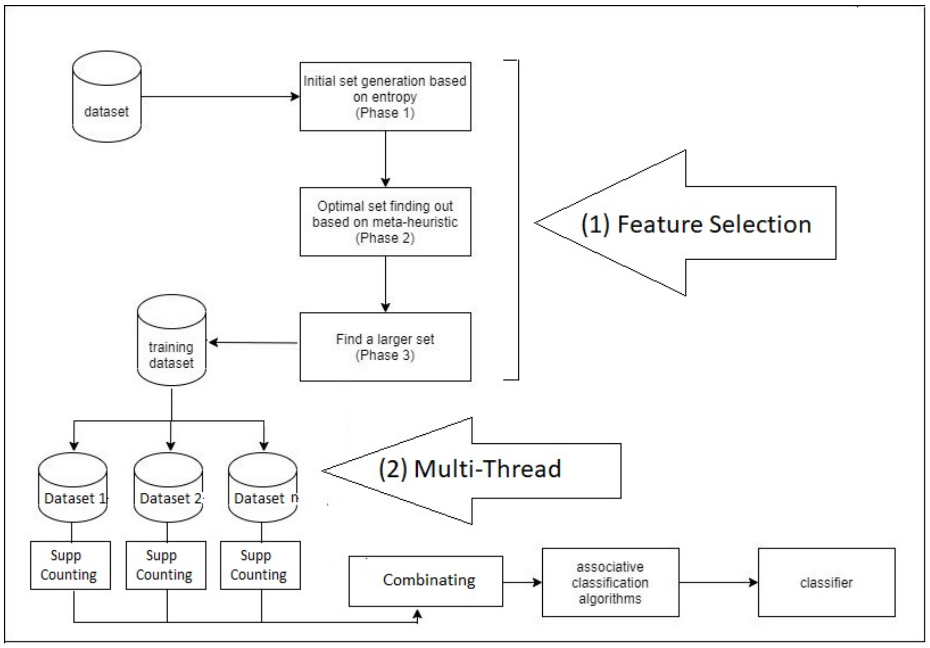
dataset
Initial set generation based on entropy (Phase 1)
Optimal set finding out based on meta-heuristic (Phase 2)
Find a larger set (Phase 3)
(1) Feature Selection
training dataset
(2) Multi-Thread
Dataset 1
Dataset 2
Dataset n
Supp Counting
Supp Counting
Supp Counting
Combinating
associative classification algorithms
classifier

Fig. 1. Two improvement approaches for Associative Classification

Associative Classification, and an illustrative example, part three presents the experiments on many different types of datasets, and part four is the conclusion.

2 Multi-Threaded Approach

2.1 Multi-threaded Algorithm

Executing many threads can sometimes be slower than executing a few threads. In reality, tasks are complex, so multi-thread programming will take advantage of the power of multi-processors. For simple datasets, single-thread is sometimes a better solution. The multi-thread problem has two main characteristics:

- Assuming the following task will wait for the previous task, if the previous task is multi-threaded, the time to complete the previous task will be as the maximum for all threads being performed in parallel (see Figure 2b).
- Aggregating threads will consume additional memory (see Figure 3).

The classification rule set is the most important part in association classification models. Algorithm 1 performs the selection for the classification rule set. In this algorithm, the function in line 3 will be performed multi-thread according to the Single instruction stream, multiple data stream (SIMD) architecture. The function in line 3 will calculate the Support, Confidence, allConfindence on a

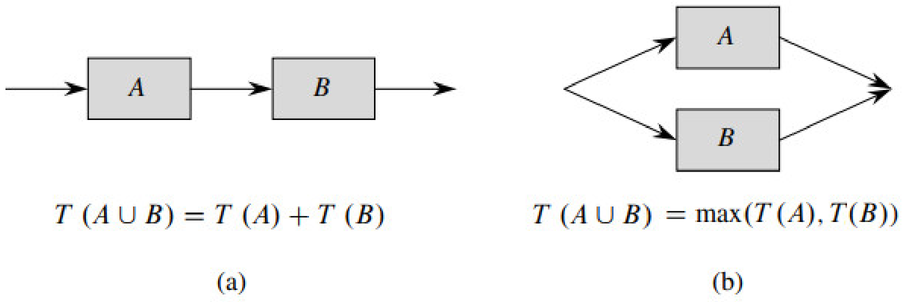

Fig. 2. Time for sequential (a) vs parallel (b)

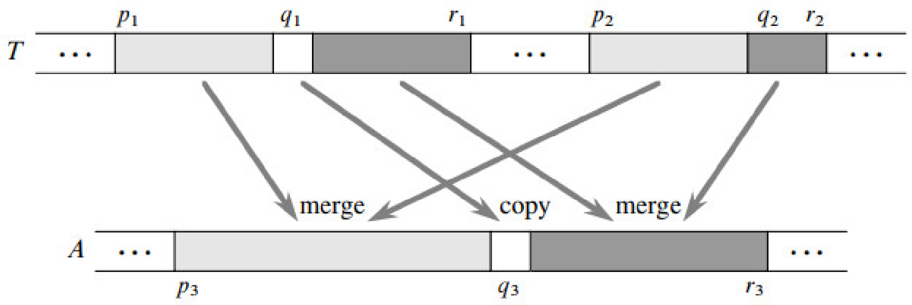

Fig. 3. Additional memory (A) needs for Aggregating threads

dataset in parallel instead of sequentially. When the original dataset is divided into many fragments, Algorithm 2 will calculate the support for each fragment, and Algorithm 3 will aggregate the support for each itemset across all fragments.

To select a good set of rules to build a model, Associative Classification methods[1, 9, 10, 11, 12, 13] relies on a few measures called the support, confidence and all-confidence:

- The support of a rule X $\rightarrow$ Y means how many records of the dataset contains the values X and Y together, divided by the total number of records.
- The confidence of a rule X$\rightarrow$ Y is how many records of the dataset contains the values X and Y together, divided by the number of records that contains the values of X.
- The all-confidence of a rule X $\rightarrow$ Y is the number of transactions containing the values X and Y together divided by the number of records containing the most frequent values that appear in either X or Y.

The all-confidence of the itemset is defined as follows

$$\text{Allconf}(X) = \frac{\text{sup}(X)}{\max(\text{sup}(a_{i_1}), \ldots, \text{sup}(a_{i_k}))} \qquad (1)$$

Where n itemset X=ai1 ,...,aij ,...,aik is a set of values of different attributes. Thus, the 1-itemset is defined by a value of any attribute, and is called aij.

Algorithm 1 ruleSelection(C_k,F_k,R)

```
 1: for all candidate $c \in C_k$ do
 2:     for all class[i] do
 3:         compute c.allconf[i], c.conf[i];// parallelize by using MultithreadAC
 4:         if c.condsup/|T| $\geq$ minsup and c.allconf[i] $\geq$ minallconf then
 5:             if c.conf[i] $\geq$ minconf then
 6:                 push $c \rightarrow class[i]$ into R
 7:             else
 8:                 push c into $F_k$
 9:             end if
10:         end if
11:     end for
12: end for
```

Algorithm 2 MultithreadAC

Input: rule(rule), subDatasets[n](n sub-dataset was divided), n(number of threads)
Output: ruleDone(rule after evaluation)

```
1: ruleEvaluated := []
2: for $i \leftarrow 0$; $i \leq n-1$; $i++$ do
3:     ruleEvaluated := ruleEvaluated U Evaluation(rule, subDataset[i]);
4: end for
5: ruleDone := Aggregating(ruleEvaluated);
6: return ruleDone;
```

Algorithm 3 Aggregating

Input: ruleEvaluated[n](Evaluation of rule on each sub-dataset)
Output: evaluated of rule after combinating

```
1: rule := ruleEvaluated[0];
2: for $i \leftarrow 1$; $i \leq n-1$; $i++$ do
3:     rule := rule + ruleEvaluated[i];
4: end for
5: return rule;
```

2.2 An example of rule selection

Suppose we have a dataset as Table 3. Let the support threshold be 2, all-confidence threshold 50% and confidence threshold 100%. According to Equation 1, all-confidence values of 1-itemsets are 100%.

Table 3. The training dataset

ID	A	B	C	D	Class
1	1	5	8	10	1
2	2	4	8	11	0
3	2	4	8	11	0
4	2	5	7	9	1
5	3	5	7	9	0
6	3	5	6	9	0
7	1	5	8	12	1
8	2	5	7	9	1

Sequential processing

First, we find a set of rules whose left-hand side is a 1-item that meets the Support, Confidence, and All-confidence threshold values. There are 4/24 rules that meet the thresholds as shown in table 4. The running cost to get table 4 is 8 (total number of data lines) x 24 = 192.

Table 4. Rules with 1-item in left-side

	Rule	Supp	Conf	AllConf
1	[A1] $\rightarrow$ 1	2	100	100
2	[A3] $\rightarrow$ 0	2	100	100
3	[B4] $\rightarrow$ 0	2	100	100
4	[D11] $\rightarrow$ 0	2	100	100

Continue to find a set of rules whose left-hand side is 2-items that satisfy the Support, Confidence, and All-confidence threshold values. There are 5/18 rules that meet the thresholds as shown in table 5. The running cost to get table 5 is 8 (total number of data lines) x 18 = 144.

This process is repeated until the k-ruleitem candidate set is empty. Then, there is no 3-item rule so the algorithm stops. The total cost to gain table 4 and table 5 is 192 + 144 = 352.

Table 5. Rules with 2-item in left-side

	Rule	Supp	Conf	AllConf
1	[A2, C8] $\rightarrow$ 0	2	100	100
2	[A2, B5] $\rightarrow$ 1	2	100	100
3	[A2, C7] $\rightarrow$ 1	2	100	100
4	[A2, D9] $\rightarrow$ 1	2	100	100
5	[B5, C8] $\rightarrow$ 1	2	100	100

Multi-Thread processing

The data set in table 3 is divided into 3 parts. Part 1 has 3 lines, part 2 has 3 lines, part 3 has 2 lines. The cost to scan data instead of 8 as sequentially, in multi-thread way will be max(3,3,2) = 3.

First, we find a set of rules whose left-hand side is a 1-item that meets the Support, Confidence, and All-confidence threshold values. There are 4/24 rules that meet the thresholds. The running cost to get table 4 (in multi-thread) is 3 x 24 = 72.

Continue to find a set of rules whose left-hand side is 2-items that satisfy the Support, Confidence, and All-confidence threshold values. There are 5/18 rules that meet the thresholds. The running cost to get table 5 (in multi-thread) is 3 x 18 = 54.

there is no 3-item rule so the algorithm stops. The total cost to gain table 4 and table 5 is 72 + 54 = 134. Suppose we ignore the cost of fork and join operations. These are two must-have operations in multi-thread process. Technically, multi-thread processing time should always be less than sequential processing time.

3 Experiment

The experimental environment is processed centrally on a computer configured with Intel(R), Core(TM) i7-6820HQ CPU @ 2.70GHz (4 CPUs), 2.7GHz and Windows 10 operating system. This experimental use ExecutorService class of concurrent library in java to divide thread to evaluate rule. There is another class can be used in this approach is Thread, The method was used to divide thread is newFixedThreadPool, submit is the method to execute a Thread created by newFixedThreadPool, method shutdown make sure all threads were done before combinating. We implemented experiments on 4 datasets as following.

Default of credit card clients.

This dataset[2] has 23 columns and 30000 instances in each columns. In that dataset, the last attribute is a target attribute named "default payment next month" to classify credit card clients as 0 or 1 (Yes = 1, No = 0).

Figure 4 depicts the results of table 6, this is the experimental result of the dataset "Default of Credit card clients", the best results are when the number of threads is 3. Memory capacity increases with the number of threads. In this experiment, we see that performing Feature Selection with Multi-Thread has

Table 6. The result of Default of credit card clients dataset

Number of threads	Multi-Thread(ms)	Multi-Thread + FS (ms)	Memory(MB)
1	564	561	1839.2
2	463	455	1873.2
3	419	412	1920.7
4	476	471	2013.5
5	512	497	2020.0

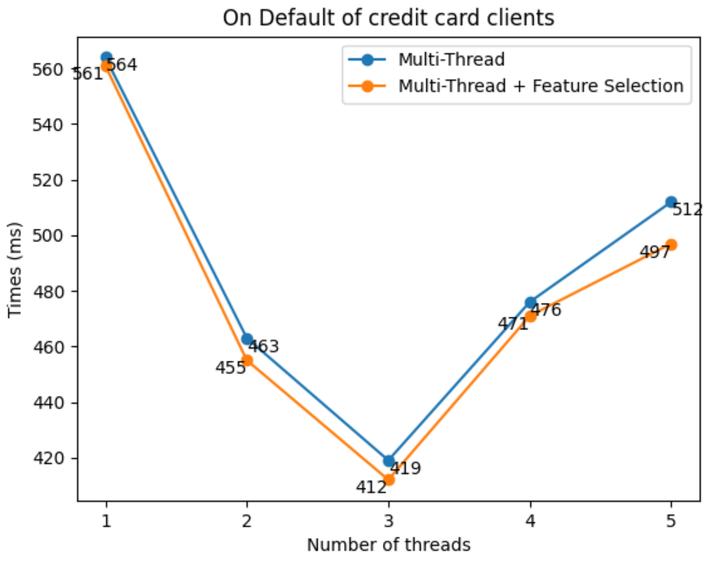

Fig. 4. The result of Default of credit card clients dataset

no significant difference because the number of redundant attributes has been eliminated quite a bit.

Smoking and Drinking Dataset with body signal.

This is a great complexity dataset [3]. There are 24 attributes and about 1 million instances with the last attribute is the target attribute named ‘DRK_YN’ to classify instance as Drink or not-Drink (Drink = y, not-Drink = N).

Figure 5 depicts the results of table 7, this is the experimental result of the dataset ”Smoking and Drinking Dataset with body signal”, the best results are when the number of threads is 5. Memory capacity increases with the number of threads. In this experiment, we see that performing Feature Selection with Multi-Thread gains significantly better results because a lot of redundant attributes have been eliminated.

Mushroom Dataset

Table 7. The result of Smoking and Drinking Dataset

Number of threads	Multi-Thread(ms)	Multi-Thread + FS (ms)	Memory(MB)
1	33763	7124	3251.6
2	27341	5314	3432.0
3	24548	3926	3579.4
4	22671	3692	3664.7
5	21205	3210	3781.3
6	24217	3361	3858.9
7	28620	3527	3971.2

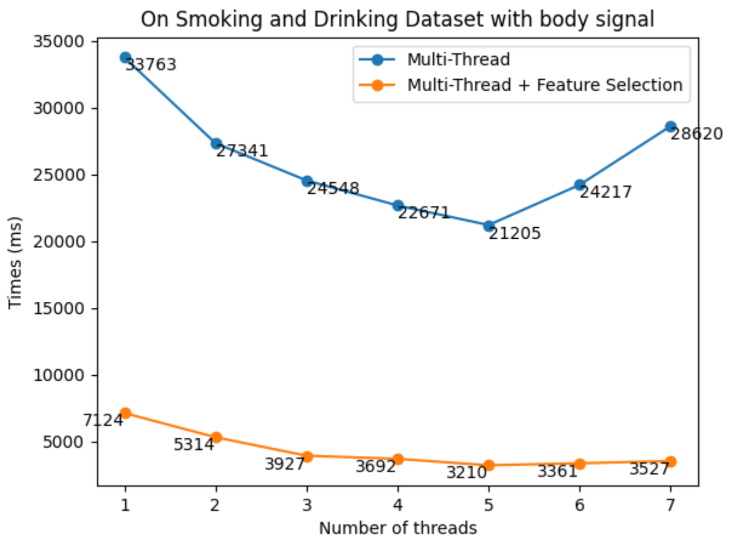

Fig. 5. The result of Smoking and Drinking Dataset with body signal

Mushroom [4] is also used by original ACAC algorithm, it has 8125 instances and 23 columns. In the data, the first attribute is a target attribute named 'class' to classify mushrooms as poisonous or non-poisonous (edible=e, poisonous=p).

Figure 6 depicts the results of table 8, this is the experimental result of the dataset Mushroom Dataset. The best result is still single thread, because this data only has 8125 instances. Too few instances to take advantage of multi-thread. Memory capacity increases with the number of threads. In this experiment, we see that performing Feature Selection with Multi-Thread gains significantly better results because a lot of redundant attributes have been eliminated.

SpamMail Dataset

MailSpam [5] has 58 columns and 6025 instances, the first attribute is a target attribute named 'spam' to classify a mail as spam or non-spam (spam = 1, non-spam = 0).

Table 8. The result of Mushroom Dataset

Number of threads	Multi-Thread(ms)	Multi-Thread + FS (ms)	Memory(MB)
1	1187	153	1839.2
2	1509	192	1873.1
3	1792	198	1920.7
4	2061	212	2013.5
5	2315	226	2020.0

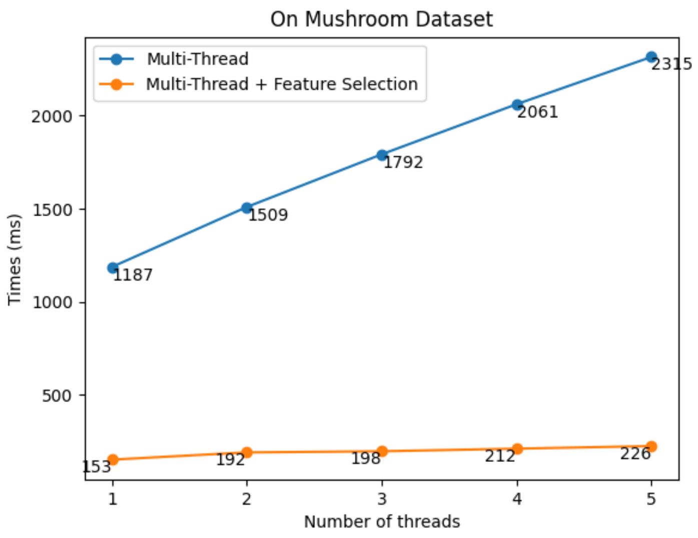

Fig. 6. The result of Mushroom Dataset

Figure 7 depicts the results of table 9, this is the experimental result of the dataset Spammail Dataset. The best result is still single thread, because this data only has 6025 instances. Too few instances to take advantage of multi-thread. Memory capacity increases with the number of threads. In this experiment, we see that performing Feature Selection with Multi-Thread gains significantly better results because a lot of redundant attributes have been eliminated.

Table 9. The result of Mail Spam Dataset

Number of threads	Times with FS(ms)	Memory(MB)
1	196204	2204.7
2	237102	2240.5
3	251073	2251.2
4	286490	2283.4
5	296705	2297.4

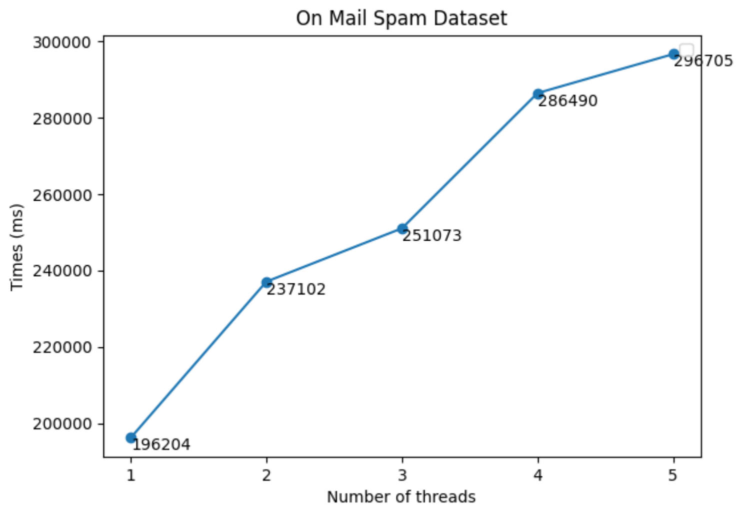

Fig. 7. The result of Mail Spam Dataset

4 CONCLUSION

The problem of multi-thread processing depends on many factors such as the number of processors, scheduling mechanism, fork and join operations.

- Not every multi-thread is fast
- Not all single threads are slower than multi threads
- The important thing is to know how to apply it properly, analyze the problem meticulously, and divide tasks appropriately to achieve the best results.

Experiments show that the SIMD architecture in multi-thread processing is suitable for data with a large number of instances. Memory capacity clearly increases with the number of threads which is a reasonable result. From there we can expand the algorithms to other multi-thread architectures such as MISD, MIMD. Anyway, multi-thread approach is also a good approach to improve associative classification algorithms in terms of running time, while feature selection helps increase classification accuracy.

References

1. Z. Huang, Z. Zhou, T. He, and X. Wang, "ACAC: Associative Classification Based on All-Confidence," *IEEE International Conference on Granular Computing*, pp. 289–293, 2011.
2. *Default of credit card clients*, [Online].
Available: `https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients`
3. *Smoking and Drinking Dataset with body signal*, [Online].
Available: `https://www.kaggle.com/datasets/sooyoungher/smoking-drinking-dataset`
4. *Mushroom Classification Dataset*, [Online].
Available: `https://www.kaggle.com/datasets/uciml/mushroom-classification`
5. *Mail Spam Dataset*, [Online].
Available: `https://www.kaggle.com/datasets/yasserh/spamemailsdataset`
6. H.Q.Huy, T.A.Tuan, D.N.Tai, "A meta-heuristic approach for enhancing performance of associative classification", ICTMAG 2024-01-22.
7. N. Q. Huy, T. A. Tuan, N. T. N. Thanh, "An efficient algorithm that optimizes the classification association rule set", VNICT 26, pp. 13-19, 2023.
8. H. F. Ong, C. Y. M. Neoh, V. K. Vijayaraj, Y. X. Low, "Information-Based Rule Ranking for Associative Classification," ISPACS, 2022.
9. M. Abrar, A. Tze and S. Abbas, "Associative Classification using Automata with Structure based Merging," IJACSAA, vol. 10, 2019.
10. D. L. Olson and G. Lauhoff, "Market Basket Analysis" in Descriptive Data Mining," Springer Singapore, 2019.
11. K. D. Rajab, "New Associative Classification Method Based on Rule Pruning for Classification of Datasets," IEEE Access, vol. 7, pp. 157783-157795, 2019.
12. H. F. Ong, N. Mustapha, H. Hamdan, R. Rosli and A. Mustapha, "Informative top-k class associative rule for cancer biomarker discovery on microarray data," Expert Systems with Applications, vol. 146, 2020.
13. Majid Seyf, Yue Xu, Richi Nayak, "DAC: Discriminative Associative Classifcation", SN Computer Science (2023) 4:401