DEVELOPING A PARALLEL CLASSIFIER FOR MINING IN BIG DATA SETS

AHAD SHAMSEEN[1], MORTEZA MOHAMMADI ZANJIREH [1*], MAHDI BAHAGHIGHAT [1], QIN XIN[2]

[1]*Computer Engineering Department, Imam Khomeini International University, Qazvin, Iran*
[2]*Faculty of Science and Technology, University of the Faroe Islands, Tórshavn, Faroe Islands*

**Corresponding author: Zanjireh@eng.ikiu.ac.ir*

(Received: 10th July 2020; Accepted: 12th November 2020; Published on-line: 4th July 2021)

ABSTRACT: Data mining is the extraction of information and its roles from a vast amount of data. This topic is one of the most important topics these days. Nowadays, massive amounts of data are generated and stored each day. This data has useful information in different fields that attract programmers' and engineers' attention. One of the primary data mining classifying algorithms is the decision tree. Decision tree techniques have several advantages but also present drawbacks. One of its main drawbacks is its need to reside its data in the main memory. SPRINT is one of the decision tree builder classifiers that has proposed a fix for this problem. In this paper, our research developed a new parallel decision tree classifier by working on SPRINT results. Our experimental results show considerable improvements in terms of the runtime and memory requirements compared to the SPRINT classifier. Our proposed classifier algorithm could be implemented in serial and parallel environments and can deal with big data.

ABSTRAK: Perlombongan data adalah pengekstrakan maklumat dan peranannya dari sejumlah besar data. Topik ini adalah salah satu topik yang paling penting pada masa ini. Pada masa ini, data yang banyak dihasilkan dan disimpan setiap hari. Data ini mempunyai maklumat berguna dalam pelbagai bidang yang menarik perhatian pengaturcara dan jurutera. Salah satu algoritma pengkelasan perlombongan data utama adalah pokok keputusan. Teknik pokok keputusan mempunyai beberapa kelebihan tetapi kekurangan. Salah satu kelemahan utamanya adalah keperluan menyimpan datanya dalam memori utama. SPRINT adalah salah satu pengelasan pembangun pokok keputusan yang telah mengemukakan untuk masalah ini. Dalam makalah ini, penyelidikan kami sedang mengembangkan pengkelasan pokok keputusan selari baru dengan mengusahakan hasil SPRINT. Hasil percubaan kami menunjukkan peningkatan yang besar dari segi jangka masa dan keperluan memori berbanding dengan pengelasan SPRINT. Algoritma pengklasifikasi yang dicadangkan kami dapat dilaksanakan dalam persekitaran bersiri dan selari dan dapat menangani data besar.

KEY WORDS: *Data mining, Big data, Decision tree, Parallel classifier, SPRINT*

1. INTRODUCTION

Recently, too much data is generated and stored on servers per day. This data has great amounts of essential and useful information and therefore, searching through this data can be one of the most essential topics in data science. The process of extracting information from large data sets and discovering patterns is termed *data mining* and it involves methods from artificial intelligence, machine learning, and database systems. One of the main goals of the data mining process is to build a logical structure from the extracted information and prepare this information for further use. The data mining process involves many sub-processes like data collection, database management, data pre-processing, and online updating [1].

In data mining, classification is one of the learning models, and the classifications are used to build a model that classifies the unclassified data in the dataset. These models are rules or patterns that connect the input data with the output target and used to predict the future output of the system that the data was taken from [2] and can be used in several fields such as marketing, advertisement, and medical science [3].

The data mining process steps that were explained by Berry and Linoff in 2004 [2] are in Figure 1.

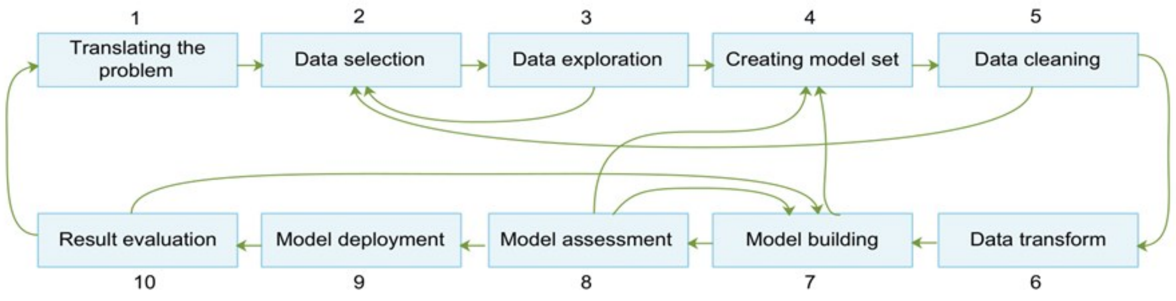


Fig. 1. Data mining process [2].

To create data mining models, we can use data mining algorithms. First, the algorithm starts by analysing the data set and extracting trends, results, and specific patterns, and these will be used next to determine the priorities and the parameters of the data mining model. Finally, these parameters are applied to the data set to find the patterns and statistical needs [4]. Selecting the best algorithm for the application depends on the data mining tasks, available data, data kind, and the size of data [5].

The decision tree algorithm is one of the most popular data mining algorithms because it has high accuracy, fast training performance, and is easy to understand. Dividing the data into several subsets is the primary purpose of this algorithm. The first step is to evaluate the attributes to determine how to divide the data between the classes. Then, this process is applied over all subsets until the decision tree is formed. The decision tree algorithm is a hybrid algorithm because it supports classification and regression tasks. The classification task can be used to predict continuous and categorical variables where the regression task can be used to predict only continuous variables. Decision tree algorithms can also be applied for association analysis [4]. The decision tree algorithm has many advantages over all data mining algorithms, which are easy to understand, quick to build the tree, and efficient to a high degree [6].

Naive Bayes algorithm is another data mining algorithm that achieves the learning phase based on evidence. This algorithm learns the evidence by counting the connection between the critical variables and all other variables [7]. Naive Bayes can detect the factors in a production line or how the products are related. However, this algorithm cannot classify nonlinearly separable classes correctly [4].

The neural network algorithm is a data mining algorithm that works by creating relationships between the input and output data and then uses these relationships to create prediction patterns [4]. The algorithm result is difficult to explain, thus, a data mining process should start with decision trees, and then use the neural network algorithm [8].

The clustering algorithm finds the variable that classifies the data, and it is often used as a step in a big data analysis project. The clustering algorithm uses two methods to assign the clusters. The first method is the K-means method. In this method, the centre is moved to the mean of all assigned objects after the assigned object's phase. The second method is the EM method (Expectation-Maximization method), that uses a probabilistic measure to assign objects to the clusters and uses a bell curve for the dimension from the low centre and a standard deviation [4, 9].

An association algorithm is a rule-based method in data mining. This method is used to discover interesting relations between the attributes in big data sets [10]. This algorithm uses impressive measures to identify the discoverable rules in the data set.

The sequence clustering algorithm is a combination of two techniques, the clustering technique and sequence technique [4]. A time-series algorithm is a series of the data collection, which is collected in a timeframe. This algorithm is used to forecast the future based on history. This algorithm can provide many services and the most commonly used specify the seasonality and multiple periods. This algorithm has an excellent advantage, and it can do a great job with minimum information [4].

The data classification process divides the data into classes where each class has a unique symbol. The data in each class have common properties. To classify the data, we have first to determine how many tasks we want, and we have to determine the methods for breaking data into ranges, this method is related to the data that we are classifying and the project that we are working on. Data classification has some challenges, such as how to determine the best method to break and split the input data, and the challenge of determining the number of classes and what these classes are.

The main goal of classifying data in data mining is to generate a predicted model, by learning from input data. This model can predict and divide the new input data between the classes. Predicted models discover relationships between the attributes that would make it possible to predict the outcome [11].

The first step in classification is prepared data in the training set, which is done in three parts [2]:

- Clean the data: we have to use smoothing techniques to remove the noise from data, and we have to solve the missing data problem by replacing it with the commonly occurring value for that attribute.
- Irrelevant attribute problem: in some databases, the attributes are not related. Therefore, correlation analysis is used to know if there is any connection between attributes.
- Database normalization: The process of organizing (such as the attributes and tables) for reducing data redundancy and improve data integrity is called normalization.

A decision tree is a compact data structure that can be built quickly and provides significant descriptions of the relationships between the input variables and the target attributes. The decision trees can provide accurate predictions if applied to the problems that can be represented with contrapuntal expressions. It can handle continuous and categorical attributes and is very powerful with separate attributes. The decision tree classifier has many advantages over all other classifiers [6]. The first advantage is that the distribution in the decision tree is free, consequently, there is no prior assumption about the previous data distributions. The second advantage is that the decision tree does not require as much training as the other classifiers. Finally, the most important advantage is that the decision tree is easy to understand. These trees can be a set of rules that describe the connection between the input and target attributes.

Sometimes the decision tree generates some unwanted and meaningless rules while it is growing deeps. This problem called the overfitting problem. The overfitting problem increases the tree size and data noise in the tree, and reduces the efficiency and accuracy of the data. To solve the overfitting problem, we have to use pruning methods to avoid a large tree size. There are two kinds of pruning methods in the decision tree. Post sorting is a kind of pruning that first builds the tree, and then reduces unwanted branches in the decision tree. The second kind of pruning method is pre-sorting. In this kind, the tree keeps on checking in each step of the building phase [12].

Each kind of pruning method has many techniques. For post sorting, there are many methods such as reduced error, error complexity, minimum error, and cost-based. For pre-sorting, there are a minimum number of object and chi-square pruning [13].

Nowadays, data mining is a critical technology in the world because all the information and data in different fields of work such as business and science are loaded and stored in servers and have much valuable information that needs to be instructed. The classification task in data mining is useful in many fields of life, and classifying data and creating the predictive model allows us an opportunity to have advanced knowledge of events. In the classification process, we are dealing with big data sets. Therefore, we have to improve the classification algorithms to be able to handle this size of data and to be more comfortable for implementation. Data size is growing very fast every day, faster than the growth in hardware abilities. To analyse and classify this size of data, we have to make classification algorithms more effective in both the time that they take to classify the data and the data structure of the algorithms.

The data mining process has many tasks like association, classification, clustering, regression, anomaly detection, and summarization. These tasks can formulate the problem and discover knowledge from data. Classification is the most common task in data mining. This technique is used to create a predictive model for future behaviour. Decision tree is the most common classification algorithm in data mining and is easy to implement, has a high degree of accuracy, and has easily recognizable patterns.

There are many algorithms to build the decision tree; each one of them has advantages and disadvantages. However, in data mining, the size of data is very big, thus in the implementation phase, all of these algorithms struggle with the amount of data that must stay in active memory until the mining process ends.

Due to the strain on computing assets caused by the massive amounts of data involved, the ideal classifier should take less time to complete the classification process, thus requiring less computer memory. In this research, we propose a new classifier to enhance classification tasks in data mining, and we will compare our proposed classifier with the SPRINT classifier in parallel computations.

The goal of this research is to design and implement a new classifier to build the decision tree for mining in big data sets. The aim is to make the classification process easier and quicker in both parallel and serial implementation and to decrease the data size that should stay in the memory. In this proposed classifier, we solve the problems in the other classifiers such as runtime and memory requirements, and also we fix the incorrect results obtained by some classifiers that are proposed to improve the runtime and memory requirements.

In our research, we have to implement two decision tree algorithms to test their runtime and the memory requirements. The training data have to be datasets consisting of several attributes. If we download a data set from the internet, we have to make many changes to it to make it more suited to the parallel programs. Consequently, we created a dataset with three attributes and use it as a training set for the implementation phase.

The rest of this article is as follows. In Section 2, we review the literature survey of related researches in the data classification algorithms. The methodology of our proposed classifier method and a comparison with the best previously proposed parallel classifier have been presented in Section 3. The simulations and experimental results of the proposed algorithm have been presented in Section 4. Finally, in Section 5, we discuss the conclusion of this research and future work.

2. RELATED WORK

Decision trees are one of the most crucial classification algorithms that can be constructed quickly and possess simple and easy-to-use models [6, 14]. The most common decision trees builders are CHAID (CHI-squared Automatic Interaction Detection), ID3 (Iterative Dichotomies 3), C4.5, CART (Classification and Regression Tree) and MARS (Multivariate Adaptive Regression Splines). Each one of them can be implemented in serial and parallel situations, but SLIQ (Scalable Classifier for Data Mining) and SPRINT (Scalable Parallel Classifier for Data Mining) algorithms have a perfect performance in parallel implementation.

CHAID classifier is a decision tree builder for dependent variables proposed in [15]. This classifier uses CHI-squared Automatic Interaction Detection (CHAID), which is a technique for analysing a large amount of data by dividing it into separated subsets. The classifier then detects the interactions between categorized variables of a data set that are the dependent variables.

The CART algorithm is the classification and regression tree published in [16]. This algorithm is used to build a prediction model from data. The prediction model is created by partitioning the data recursively and fitting a simple prediction model in each partition, then, a decision tree is built to represent and describe the partition graphically.

In [17], the authors presented a framework called CLS (Concept Learning System). This system builds a decision tree with the minimum cost of classifying. The cost of classifying consists of two components: the cost of determining the property of an object and the cost of deciding. In [6], Ross Quinlan presented ID3, which is one of the serial algorithms developed from the CLS algorithm, which is used to build the decision tree invented. It handles category attributes only, begins with the original set as the root node, and all the attributes in the set belong to the same class. ID3 algorithm is suffering from an overfitting problem. To solve the overfitting problem, Ross Quinlan created a new algorithm in 1993 and named it the C4.5 algorithm.

In [18], the MARS algorithm, which is a group of techniques implemented together, was presented. This algorithm solves the regression-type problems and has the same purpose as the classification algorithms for predicting the values of the dependent (outcome values) and independent (income values) values.

Most of the classification algorithms are designed for resident memory data, and this issue limits the ability to mine in large data sets. In [19], the SLIQ algorithm was proposed to build the decision tree to solve the resident memory data problem by using a new data structure. In [20], the authors designed SPRINT. This algorithm has the same advantage of SLIQ, but a different data structure. SPRINT data structure is an attribute list with three rows, the first row is the attribute value, the second row is the class, and the third row is the index of the record. This list is sorted in the first step and only one time.

In [21], the ScalParC algorithm (Scalable Parallel Classifier algorithm) proposed for enhancing the runtime and the memory required in SPRINT. ScalParC has the same data structure of SPRINT, and it creates a separate list for each attribute consisting of the attribute, the class, and the index. It then creates another data structure to stay in the memory during execution time that determines to which node each datum belongs to. The difference between the two algorithms is that ScalParC distributes the data structure, which stays in memory between the processors to increase the memory request and execute the split by level not by not to increase the communication time [21]. However, in 2000, Kevin Bowyer has proved that ScalParC gives incorrect results in some situations [22].

CLOUDS algorithm is a breadth-first strategy to build the decision tree and uses the Gini index for evaluating the split points [23]. It uses either sampling the splitting method or sampling the splitting points with estimation method to determine the splitter at each node of the tree, and it evaluates the split points for categorical attributes as in SPRINT.

In [24], the authors presented a Random Forest classifier. It consists of numbers of simple trees, each one of these trees is capable of producing a predicted response. For the classification tasks, the response of these trees has a class membership form. This task classifies the independent variables in categories. For the regression task, the response is an estimation of the dependent variable, given the predictors. Each tree in the forest has an individual responsibility for the queries. This response depends on the predictor values which have the same distribution for all the trees and are selected independently.

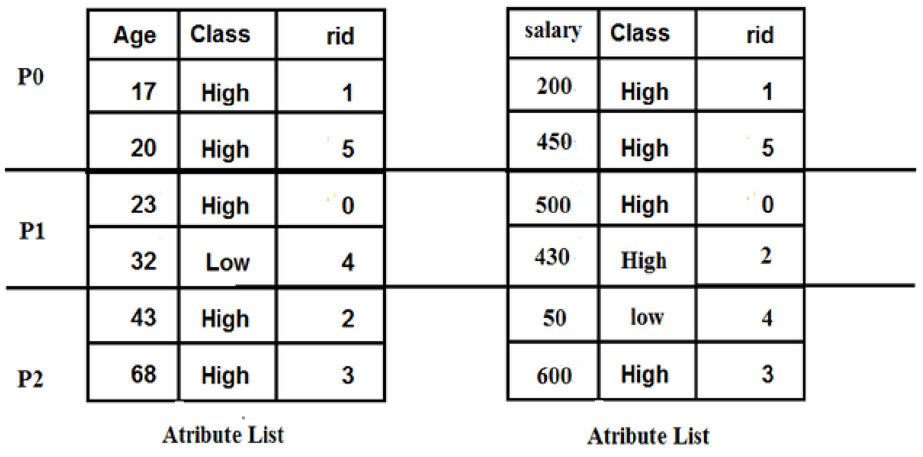
The public classifier is developed from SPRINT and proposed in [25]. It has the same data structure, the same steps, and the phases. The main goal of the public is dealing with data that has much noise and suffers from missing and incorrect values, the kind of problems that can be fixed using the pruning method.

3. METHODOLOGY

The new algorithm has the same data structure for SPRINT. It creates an attribute list for each attribute. This list consists of three rows: the attribute, the class, and the rids. The sorting process is done in the first phase, and it is done once. Our algorithm also uses the hash table to determine the rids in each node and use the Gini index to found the best split point in each attribute list.

In the first step, the parallel pattern attribute records are distributed equally over all the processors in SPRINT and the proposed algorithm. Thus, in this case, each processor processes the same number of records. Figure 2 shows parallel data placement in the proposed algorithm in the first step.

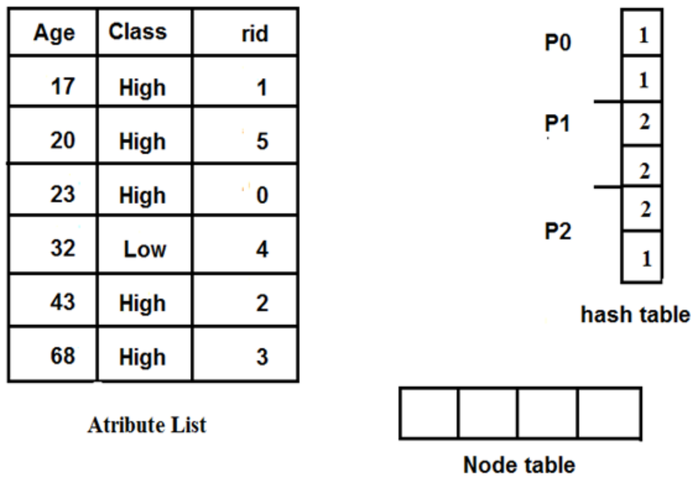
In our algorithm, we distribute the hash table between the processors and create a new data structure called the node table. This data structure determines the number of classes in each node because in the calculation phase, each processor calculates (Gini) for its rids. In this phase, the processor needs information about the rids in each node and the number of each class in the node; thus, this new data structure determines the number of classes in nodes in each level of the tree. Figure 3 shows the new data structure in the proposed algorithm.


P0
P1
P2
Atribute List
Atribute List

	Age	Class	rid
P0	17	High	1
	20	High	5
P1	23	High	0
	32	Low	4
P2	43	High	2
	68	High	3

	salary	Class	rid
P0	200	High	1
	450	High	5
P1	500	High	0
	430	High	2
P2	50	low	4
	600	High	3

Fig. 2. The parallel data placement in the proposed algorithm.



Age	Class	rid
17	High	1
20	High	5
23	High	0
32	Low	4
43	High	2
68	High	3

Fig. 3. The new data structure in the proposed algorithm.

The proposed algorithm has many advantages over SPRINT, and it enhances many weak points in SPRINT. In SPRINT, the size of the hash table is equal to the size of the data set, and the hash table is memory resident. Each processor has to frequently connect to the hash table to determine the rids in each node of the tree, and the hash table must be updated after each split.

The total communication overhead (per processor) in SPRINT is O(N), where N is the number of records in the dataset, and the total memory requirement (per processor) is O(N), as the size of the hash table is the same order as the size of the training dataset for the upper levels of the decision tree, and it resides on every processor. Therefore, SPRINT is unscalable in runtime and memory requirements.

The communication cost in SPRINT is very high, especially when the data set is vast. In this case, the size of hash table will be a problem because it should stay in memory and the frequent connection with the hash table increases the runtime of the algorithm. Figure 4 shows the communication cost in SPRINT.

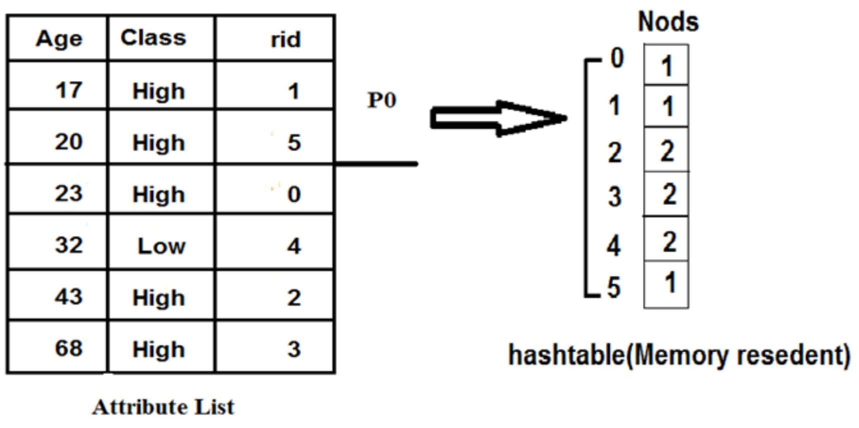


Fig. 4. Communication cost in SPRINT.

In the proposed algorithm, the hash table is distributed between the processors. In the calculation phase, each processor will connect with part of the hash table to determine the rids in the node and will connect to the node table to determine the number of classes in the node. Each processor will update its part of the hash table.

In this way, the total communication; over each processor is O(N/P+M/2), P is the number of processors, and M is the size of the node table (M=number of classes* 2), in this case, we decrease the communication cost and the time of the update. Figure 5 shows the communication cost in the proposed algorithm.

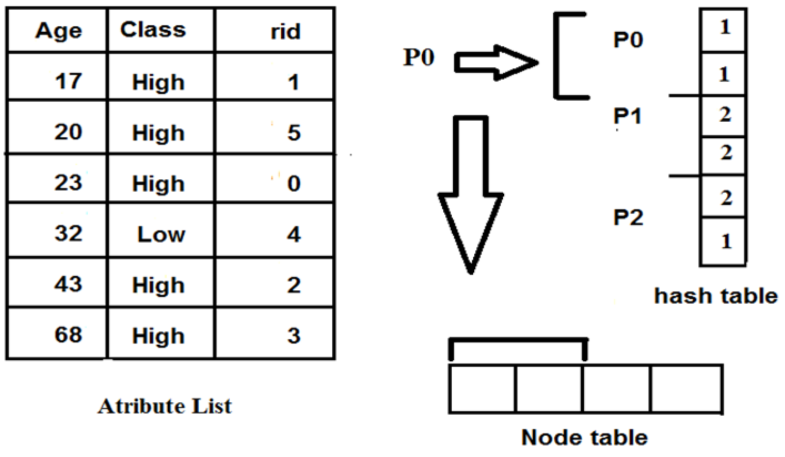


Fig. 5. Communication cost in the proposed algorithm.

In SPRINT, the records in the attribute list are distributed between the processors equally in the first step after the sorting process, and this distribution continues until the last step. Each processor processes the records that were assigned to it in the first step.

This approach caused a communication problem when the records that were assigned to one processor are located in more than one node. In this case, the fast processor will wait until the slower processor finishes its work, increasing the runtime of the classification process.

The reason for the problem is that the process is done per level. Thus, the records in more than one node are evaluated to determine the best split in each node, in this case, waiting time will be considerable because each processor is waiting for another processor to finish its work and each processor is evaluating records for more than one node; thus, the creation of a new node will take much more time. Figure 6 shows the communication problem in SPRINT.

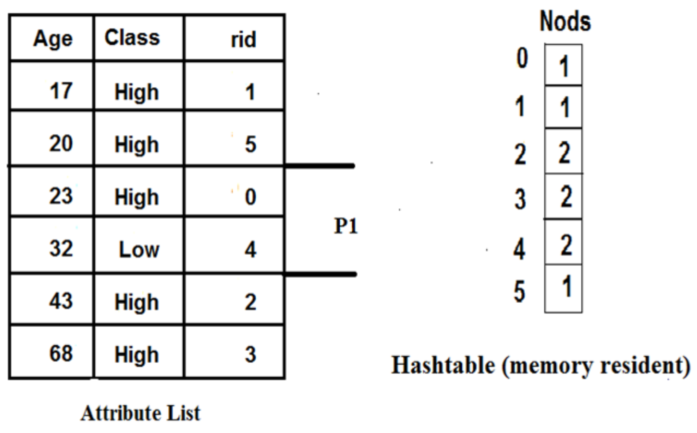


Fig. 6. Communication problem is SPRINT.

In our algorithm, distributing the records between the processors is based on two rules. The first rule is that the distribution is done per node, not per level. Assigning the record is done in each node, therefore, there is no previous record assignation in the nodes and each processor connect with its part of hash table to determine how many records are in its responsibility in each node and the max number of records that each process handles is (N/P).

In this way, the split process in each node is done entirely before going to the next node, and the processors evaluate the split point only for one node, and this will decrease the waiting time in each processor.

To determine the best attribute for a split in the node, each processor connects with its part of the hash table and determine the rids that belong to the node. After determining the rids in the node, each processor performs calculation operations on these rids for the records in remaining attributes to select the best attribute for the next split step. Figure 7 shows the communication per node in the proposed algorithm.

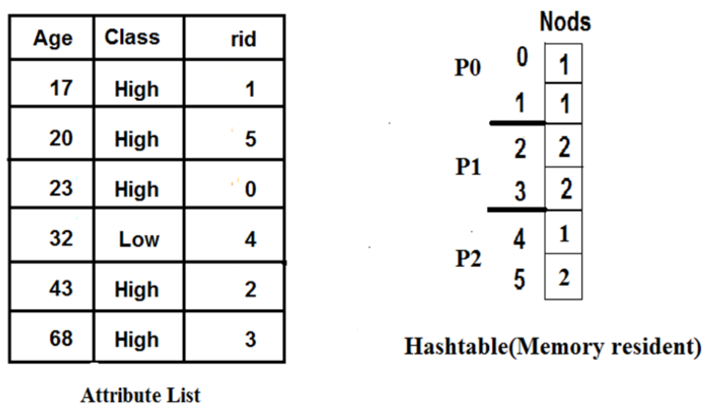


Fig. 7. Communication per node in the proposed algorithm.

The size of the hash table is a critical issue when dealing with a big data set because, in the high level of the tree, the size of the hash table is in the same order as the size of the dataset.

In SPRINT, the entire hash table stays in the memory of each processor, and in each evaluation step, the processor needs to connect to all the elements in the hash table to determine the rids in each node. In our proposed algorithm, the hash table is distributed among all the processors; thus, the memory-resident in each processor is (N/P), and the new data structure also stay in memory; thus, the memory-resident of our algorithm is (N/P + M), where M is the size of the node table, and in each evaluate step, the processor connects only with its part of the hash table. Figure 8 shows the memory requirements in both SPRINT and the proposed algorithm.

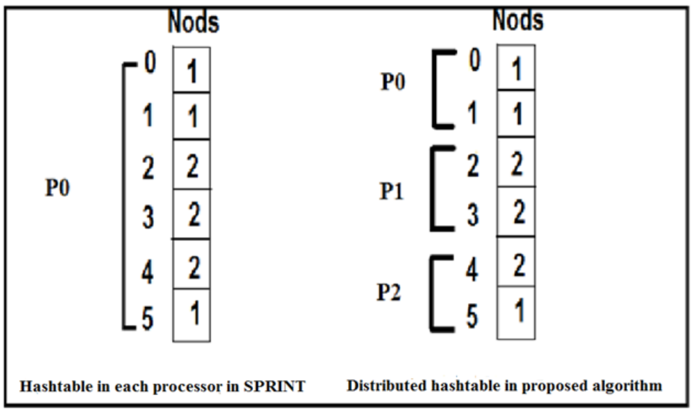

Nods
P0
0 1
1 1
2 2
3 2
4 2
5 1
Hashtable in each processor in SPRINT
Nods
P0
0 1
1 1
P1
2 2
3 2
P2
4 2
5 1
Distributed hashtable in proposed algorithm

Fig. 8. Hash table (resident memory) in SPRINT and the proposed algorithm.

In SPRINT, each processor has to stay in contact with all the hash tables in each step to determine the rids in each node of the tree. Therefore, all the hash tables should stay in memory until the build process is completed. However, in the proposed algorithm, there is no need to keep the entire hash table in memory, the hash table is divided between the processors, and each one of them connects with its part of the hash table to determine the rids in the tree node.

Processors update the hash table in each level of the tree in the proposed algorithm after the split step. Because the hash table is distributed between the processors, each processor updates its part of the table. Figure 9 shows how hash table parts are updated in each processor.

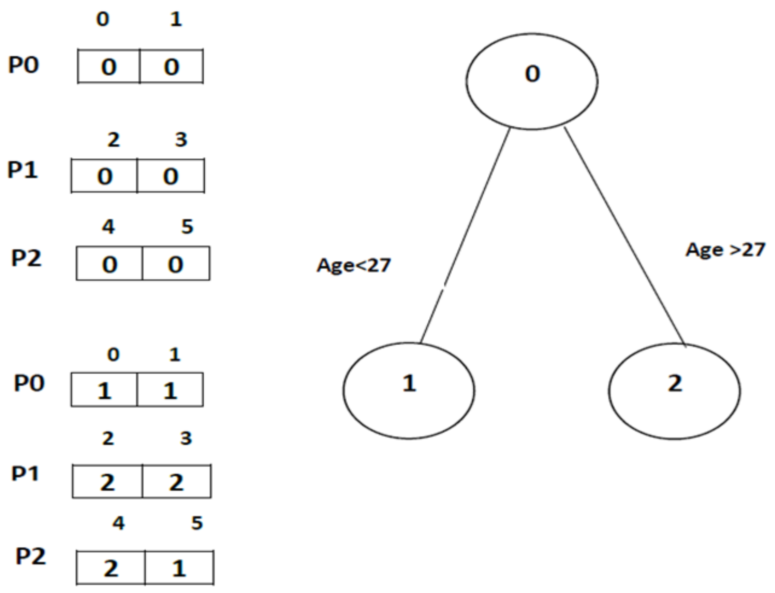


Fig. 9. Update of the hash table in the proposed algorithm.

Communication cost is one of the most critical issues in the classification process when we are dealing with big data sets. The changes that we have made in the SPRINT algorithm will decrease this cost and make the algorithm more scalable in runtime and memory size. The new data structure will make the algorithm more efficient for massive data sets, and the connection with the hash table will be at the minimum level.

To execute the evaluation step, the split step per node will let the tree be constructed faster, and there will much less waiting time. This approach of work distribution between the processors will make them focus on finishing the tree node by node.

The proposed algorithm data structure and the per-node build strategy will enhance the decision tree building process. The communication problem that is happening when the processor evaluates the records in more than one node in one time will be solved.

4. EXPERIMENTAL RESULTS

4.1. Evaluating the execution time

We have implemented the SPRINT algorithm and the proposed algorithm using MPI, and data sets in up to (10^5) and we used six, nine, and twelve processors in a cluster with total processing hardware of 80 cores/10 nodes and Ram 16 GB, we got the attribute value from the database and created an attribute list for each attribute, then we sorted the attributes.

After the sorting phase, we executed the steps of the algorithm. First, we found the Gini for each attribute, then selected the best split and the split point. Finally, we executed the split and repeated this circle for the remaining attributes. Figure 10 shows the execution time in both SPRINT and the proposed algorithm.

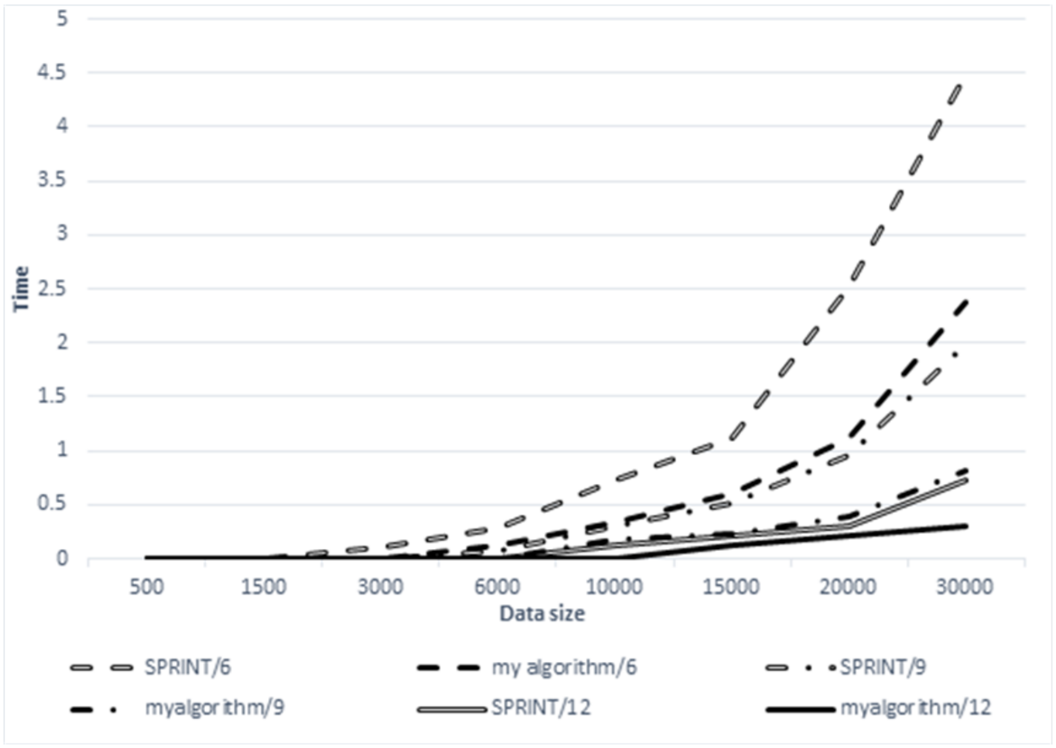


Fig. 10. The execution time (in second).

From Figure 2, we can divide the different runtime between two algorithms into three parts, the first part is when data is between (500-3000), the proposed algorithm gives a better result than SPRINT but the range of execution time is not significant, the second part when data is from (3000-6000), proposed algorithm also gives a better result than SPRINT and the range in execution time is bigger than the first part. We can see that every time the size of the data gets bigger, the difference range in runtime between the algorithm gets better and the proposed algorithm gives a better result.

In SPRINT, when the data size gets bigger, the hash table does not fit in memory, so we have to do many rounds to update the hash table, these rounds take much time and increase the communication cost because the next step is dependent on the hash table rids to determine the node for each attribute value.

In our algorithm, a hash table is distributed between the processors, so even if the data size gets bigger, the hash table in each processor fits in memory, and we need no more rounds. The results of implementation show that the proposed algorithm is faster than SPRINT in small and big data sets. Table 1 shows the results of implementation.

Table 1: Comparing the runtime of SPRINT and the proposed algorithm (in second).

Data Size (records)	**SPRINT 6**	**SPRINT 9**	**SPRINT 12**	**My algorithm 6**	**My algorithm 9**	**My algorithm 12**
500	0.00	0.00	0.00	0.00	0.00	0.00
1500	0.00	0.00	0.00	0.00	0.00	0.00
3000	0.10	0.00	0.00	0.00	0.00	0.00
6000	0.29	0.08	0.00	0.12	0.00	0.00
10000	0.72	0.3	0.12	0.34	0.18	0.00
150000	1.12	0.51	0.22	0.6	0.24	0.12
20000	2.52	0.96	0.3	1.12	0.39	0.22
30000	4.48	1.98	0.72	2.28	0.82	0.31

4.2. Evaluating the memory requirement

In SPRINT, the size of the hash table is equal to the order of the size of the training dataset for the upper levels of the decision tree. It is resident in all processors and the number of processors does not have any effect on the size of the hash table. Thus, when we are dealing with massive data sets, the hash table size is a series problem.

In the proposed algorithm, the hash table is distributed between the processors, therefore, if the data set order is O(N), then the hash table in each processor is O(N/P); by this solution, we solved the memory requirement problem. Figure 11 shows memory requirements in each processor for both algorithms.

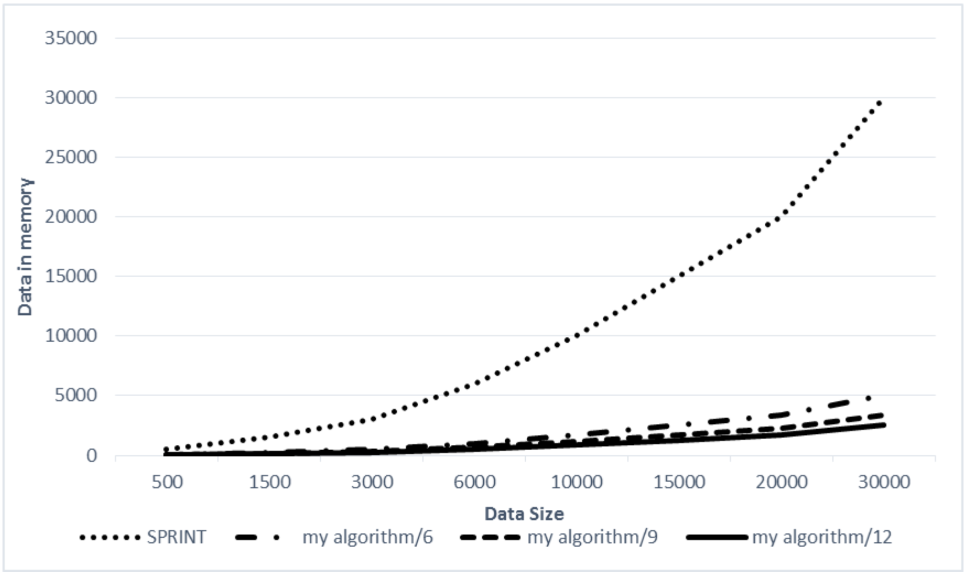


Fig. 11. The memory requirement in each processor.

Data size is a severe problem because hardware cannot handle the fast growth of data size, therefore, developing the existing classification algorithms and making them able to handle a huge data size, without the need of potent hardware, is very important. Table 2 shows the memory requirement in SPRINT and our algorithm.

Table 2: Comparing the memory requirement by the record in each processor.

Data Size (records)	**My algorithm 6**	**My algorithm 9**	**My algorithm 12**	**SPRINT**
500	88	60	46	500
1500	254	171	129	1500
3000	504	338	254	3000
6000	1004	671	504	6000
10000	1671	1116	838	10000
15000	2504	1671	1254	15000
20000	3338	2227	1671	20000
30000	5004	3338	2504	30000

In the previous experiment, the proposed algorithm showed a better result than SPRINT. It needs a smaller memory size per processor than SPRINT. In our algorithm, the hash table should stay in memory in all the steps, because it is essential to determine the record rids in each node. In SPRINT, the hash table in each node has the same size order as the data set, thus, if the dataset is huge, SPRINT suffers from a memory requirement problem and needs more time to complete the classification process. However, in our algorithm, hash table size is distributed equally among the processors. Thus, each processor has (N/P) records which stay in memory, and the node table also stays in memory and has size M which is the number of classes, therefore, the memory requirement for our algorithm is (N/P +M).

5. CONCLUSION AND FUTURE WORK

A decision tree is one of the most popular algorithms in data mining. It has many advantages such as being easy to understand, having a high degree of accuracy, and building the tree very quickly. The decision tree algorithm can handle both categorical and continuous attributes. This algorithm classifies the data set records and builds a predictive model that can be used to predict future outputs.

There are many techniques to build the decision tree such as C4.5, ID3, CART, and CHAID. Each one of them has a unique method and has many advantages, which make the building process easier and more efficient. In data mining, the decision tree building algorithms deal with very big data sets; thus, these algorithms must be able to handle this size of data and consider the ability of the computer's hardware.

Most of the decision tree algorithms are designed for resident memory data, this approach decreases the ability of these algorithms when the data sets are significant. Therefore, many studies have been done in this area, and many algorithms have been proposed to solve the limited ability of the existing algorithms to handle big data sets. SPRINT algorithm is designed to solve the resident memory problem using a new data structure, which decreases the resident memory data size and makes the parallel designing of the decision tree easier.

In this paper, we proposed a new parallel builder for the decision tree, which solves the runtime problem and the memory requirement problem in other algorithms. We tested the new algorithm using MPI and compared the results with the best parallel algorithm, SPRINT.

Our algorithm showed a better result than SPRINT with different sizes of data sets. It also showed a better result in runtime because the communication cost of the split step was decreased from O(N) to O(N/P +M/2) after we distributed the hashtable among the processors.

Distributing the hashtable also solved the memory requirement problem because all the processors share the size of the dataset equally. In this case, the algorithm will not suffer from overfitting problems when the size of the hashtable is too big and there will be no need for many rounds to update the hashtable.

Also in SPRINT, the initial distribution of the attribute lists records between the processors was not changed. This approach caused communication problems when the rids for one processor existed in more than one node. However, in the proposed algorithm, we solved this problem by building the tree node by node; thus, in the evaluation step, each processor evaluated its rids in one node, in this case, the process completed the node and went to another node.

By evaluating the implementation results for SPRINT and the proposed algorithm, we can see that the proposed algorithm is better than the SPRINT in runtime and memory requirements. These advantages make the proposed algorithm more useful when the size of the dataset is significant. Selecting the best algorithm to build the decision tree is based on many rules, the size of the dataset is one of the critical issues that should be considered in the selection process. There are many algorithms that are very useful and powerful when the amount of data is not very big such as C4.5 and CART, but, when the size of dataset is big, SPRINT and the proposed algorithm are better than other algorithms. The new data structure of the proposed algorithm helps to handle this big size, and it is even better than the SPRINT data structure.

In parallel computing, load balancing between the processors is essential, and the data load must be divided equally between the processors, otherwise, some processors will work more than others. In the decision tree, some nodes reach a dead end, and this happens when all the records in the node are related to one class. In this case, there is no need to evaluate and process the records in these nodes. Our algorithm suffers from an unbalancing problem when some nodes reach a dead end, and some processors are processing more than the others, which increases the runtime of the algorithm. For the future work, we will solve the unbalancing problem when some nodes reach a dead end and divide the work equally between the processors to decrease the runtime. This step will enhance the building process and make the proposed algorithm more powerful and more suitable for massive data sets.

Nowadays, data scientists are more interested in deploying emerging and strong deep learning (DL) algorithms [26-30]. DL can perform big data analytics efficiently. It can be widely utilized for handling some traditional problems in Big Data Analytics such as data labelling, semantic indexing, probing complex and nonlinear patterns from giant volumes of data, fast and efficient information retrieval, and simplifying discriminative tasks. As a result, it would be a progressive solution to our work in the future.

REFERENCES

[1] Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth (1996). From data mining to knowledge discovery in databases. AI magazine. 17(3):37-37. https://doi.org/10.1609/aimag.v17i3.1230

[2] Berry, M.J. and G.S. Linoff (2004). Data mining techniques: for marketing, sales, and customer relationship management. John Wiley & Sons.

[3] Tan, P.-N., M. Steinbach, and V. Kumar (2016). Introduction to data mining. Pearson Education India.

[4] Aggarwal, C.C. (2014). An Introduction to Data Classification. Data Classification: Algorithms and Applications. Chapman and Hall/CRC.

[5] Zaki, M.J. and W. Meira (2014). Data mining and analysis: fundamental concepts and algorithms. Cambridge University Press.

[6] Quinlan, J.R (1986). Induction of decision trees. Machine learning. 1(1):81-106. https://doi.org/10.1007/BF00116251.

[7] Rennie, J.D., et al (2003). Tackling the poor assumptions of naive Bayes text classifiers In Proceedings of the Twentieth International Conference on Machine Learning: August 21-24, 2003; Whasington DC. https://dl.acm.org/doi/10.5555/3041838.3041916.

[8] Hagan, M., et al (2014). Neural Network Design. 2nd Edtion. Oklahoma. Martin Hagan.

[9] Jain, A.K. and R.C (1988). Dubes, Algorithms for clustering data. Prentice-Hall, Inc.

[10] Zhang, C. and S. Zhang (2003). Association rule mining: models and algorithms. Springer.

[11] Aggarwal, C (2014). Data Classification: Algorithms and Applications, ser. Frontiers in physics. Chapman and Hall/CRC.

[12] Esposito, F., et al (1997). A comparative analysis of methods for pruning decision trees. IEEE transactions on pattern analysis and machine intelligence. 19(5): 476-491. https://doi.org/10.1109/34.589207.

[13] Mingers, J (1989). An empirical comparison of pruning methods for decision tree induction. Machine learning. 4(2):227-243. https://doi.org/10.1023/A:1022604100933.

[14] Rokach, L. and O.Z. Maimon (2008). Data mining with decision trees: theory and applications. World scientific.

[15] Kass, G.V (1980). An exploratory technique for investigating large quantities of categorical data. Journal of the Royal Statistical Society: Series C (Applied Statistics). 29(2):119-127. https://doi.org/10.2307/2986296.

[16] Breiman, L., et al (1984). Classification and regression trees. CRC press.

[17] Hunt, E.B , J. Marin, and P.J. Stone (1966). Experiments in induction. Academic Press.

[18] Friedman, J.H (1991). Multivariate adaptive regression splines. The annals of statistics, 19(1):1-67. https://doi.org/10.1214/aos/1176347963

[19] Mehta, M., R. Agrawal, and J. Rissanen (1996). SLIQ: A fast scalable classifier for data mining. In Proceedings of the International conference on extending database technology: 25-29 March 1996; Avignon, France; pp 18-32. https://doi.org/10.1007/BFb0014141.

[20] Shafer, J., R. Agrawal, and M. Mehta (1996). SPRINT: A scalable parallel classifier for data mining. In Proceedings of the 22nd VLDB Conference. 3-6 September 1996; Mumbai(Bombay), India; pp 544-555.

[21] Joshi, M.V., G. Karypis, and V. Kumar (1998). ScalParC: A new scalable and efficient parallel classification algorithm for mining large datasets. In Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing. 30 March-3 April 1998; Orlando, FL, USA. https://doi.org/10.1109/IPPS.1998.669983.

[22] Bowyer, K.W., et al. A parallel decision tree builder for mining very large visualization datasets. In Proceedings of the ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interaction. 8-11 Oct 2000; Nashville, TN, USA. https://doi.org/10.1109/ICSMC.2000.886388.

[23] Ranka, S. and V. Singh (1998). CLOUDS: A decision tree classifier for large datasets. In Proceedings of the 4th Knowledge Discovery and Data Mining Conference. 27 – 31 August 1998; New York, USA. https://dl.acm.org/doi/10.5555/3000292.3000294.

[24] Liaw, A. and M. Wiener (2002). Classification and regression by random Forest. R news. 2(3):18-22.

[25] Rastogi, R. and K. Shim (2000). PUBLIC: A decision tree classifier that integrates building and pruning. Data Mining and Knowledge Discovery. 4(4):315-344.

[26] Bahaghighat, M., et al (2020). Estimation of wind turbine angular velocity remotely found on video mining and convolutional neural network. Applied Sciences. 10(10):35-44. https://doi.org/10.3390/app10103544.

[27] Bahaghighat, M., et al (2020). ConvLSTMConv network: A deep learning approach for sentiment analysis in cloud computing. Journal of Cloud Computing: Advances, Systems and Applications. 9(16). https://doi.org/10.1186/s13677-020-00162-1.

[28] Abedini, F., et al (2019). Wind turbine tower detection using feature descriptors and deep learningFacta Universitatis, Series: Electronics and Energetics. 33(1):133-153. https://doi.org/10.2298/FUEE2001133A.

[29] Bahaghighat, M., et al (2019). Vision Inspection of Bottle Caps in Drink Factories Using Convolutional Neural Networks. In Proceedings of the IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP). 5 - 7 September 2019; City Plaza, Cluj-Napoca, Romania. https://doi.org/10.1109/ICCP48234.2019.8959737.

[30] Bahaghighat, M., et al (2019). A machine learning based approach for counting blister cards within drug packages. IEEE Access. 7: 83785-83796.
https://doi.org/10.1109/ACCESS.2019.2924445.