



Accuracy and diversity-aware multi-objective approach for random forest construction

Nour El Islem Karabadji, Abdelaziz Amara Korba, Ali Assi, Hassina Seridi,
Sabeur Aridhi, Wajdi Dhifli

► To cite this version:

Nour El Islem Karabadji, Abdelaziz Amara Korba, Ali Assi, Hassina Seridi, Sabeur Aridhi, et al.. Accuracy and diversity-aware multi-objective approach for random forest construction. Expert Systems with Applications, 2023, 225 (1), pp.120138. 10.1016/j.eswa.2023.120138 . hal-04079595

HAL Id: hal-04079595

<https://inria.hal.science/hal-04079595v1>

Submitted on 9 Jul 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

Accuracy and Diversity-Aware Multi-Objective Approach for Random Forest Construction

Nour El Islem Karabadji^{a,b,c,*}, Abdelaziz Amara Korba^{d,e}, Ali Assif^f, Hassina Séridi^c, Sabeur Aridhi^g and Wajdi Dhifli^{h,**}

^aHigher School of industrial technologies, Annaba, P.O. Box 218, 23000, Algeria

^bLaboratoire De Technologies Des Systemes Energetiques (LTSE), E3360100, Annaba, P.O. Box 218, 23000, Algeria

^cElectronic Document Management Laboratory (LabGED), Badji Mokhtar-Annaba University, P.O. Box 12 Annaba, Algeria

^dNetworks and Systems Laboratory (LRS), Badji Mokhtar-Annaba University, P.O. Box 12 Annaba, Algeria

^eL3i, University of la Rochelle, La Rochelle France

^fHouse of Commons, 181 Queen Street, Ottawa, Ontario K1A 0A6, Canada

^gLORIA, Campus Scientifique, University of Lorraine, BP 239, 54506 Vandoeuvre-lès-Nancy, France

^hUniv. Lille, CHU Lille, ULR 2694 - METRICS : Évaluation des technologies de santé et des pratiques médicales, F-59000 Lille, France

ARTICLE INFO

Keywords:

Random Forest
Decision Tree
Classification
Genetic Algorithm
Diversity
Internet of Things

Abstract


Random Forest is an ensemble classification approach. It aims to design a discrete finite group of decision trees constructed based on bootstrap samples and random attribute selection. Random Forests have strong generalization capacities due to the variance in the training and attribute couple subsets used for constructing different decision trees in the forest. However, to construct a robust and effective random forest, two main issues need to be taken into account namely: 1) increasing the accuracy and diversity of decision trees; 2) decreasing the number of decision trees. In this paper, a genetic algorithm-based approach to tackle the aforementioned challenges related to random forest construction is proposed. Three objectives are taken into consideration. First, strengthening the classification accuracy of individual decision trees as well as that of the forest. Second, making use of diversity measures among the decision trees to improve the generalization of the constructed model. Third, minimizing the number of trees in the forest and finding an optimal subset of the random forest. An experimental evaluation on several datasets from the UCI Machine Learning Repository is conducted. The obtained results show that the proposed approach outperforms state-of-the-art classical as well as evolutionary random forest construction methods. Finally, the proposed approach is used to build a reliable random forest model for detecting Botnet traffic in Internet of Things environment.

1. Introduction

Random Forest (RF) is one of the most widely used classifiers (Resende and Drummond, 2018). It consists of a set of decision trees (called *weak learners*) trained in parallel then used to predict the final output following a majority-voting system (Mohapatra, Shreya and Chinmay, 2020). While each individual *weak learner* (decision tree) can be interpreted easily by the user, it is unstable. It strongly depends on the training dataset and it is affected by the uncertainties of the data (e.g., particularity, noise and residual variation) (Karabadji, Seridi, Bousetouane, Dhifli and Aridhi, 2017). In fact, slight changes in the training data can cause considerable changes in the generated decision tree. RF overcomes the limitations of the decision tree algorithm by using an ensemble of decision trees. In theory, such a strategy will allow RF to achieve a better classification accuracy compared to a single decision tree and to have stronger generalization capacities. However, in practice, multiple of the generated decision trees in a RF model can be very correlated. This could limit the accuracy and generalization capacities of the overall RF model. Moreover, data particularity and diversity of the generated decision trees must be taken into consideration. To optimize the constructed RF model, generating a large number of accurate and heterogeneous decision trees can lead to building a more robust and efficient RF. This solution is not trivial due to the increasing in memory requirement and computational cost (Adnan and Islam, 2016). To overcome such limitations, reducing the number of the generated decision trees with

*Corresponding author: Nour El Islem Karabadji Tel.: +213 (0) 38-43-84-02; Fax: +213 (0) 38-43-84-01;

**Corresponding author: Wajdi Dhifli Tel.: +33 (0)3 20 96 40 40; Fax: +33 (0)3 20 95 90 09;

 n.karabadji@esti-annaba.dz (N.E.I. Karabadji)

ORCID(s):

respect to their diversities can be considered. Such a diversity leads to eliminating the identical decision trees during the RF construction phase. Indeed, some of these identical (and highly similar) decision trees could even be hindering the RF's performance. The contributions of the decision trees in a RF are often different to improve the generalization of the model.

Let us have a set of training samples Tr described by a set of attributes A and a function f . This function allows to select from A the best attribute that optimizes splitting Tr to new subsets. If all attributes must be used, then the couple (Tr, A) will always generate the same decision tree. However, adding or removing a training sample or an attribute may produce a different decision tree (Karabadjji, Khelf, Seridi, Aridhi, Remond and Dhifli, 2019). Following this definition and according to a bootstrap sampling and random attributes selection, a RF can be defined by a group G consisting of training samples Tr_i and subsets of attributes A_i . However, searching for the best group G^* from an extremely large search space of candidates, denoted by \mathcal{P} , requires a significant computational cost. Suppose that we have n training samples described by m attributes, there could be $\mathcal{O}(n^n)$ possible non-empty training sets, and $\mathcal{O}(2^m - 1)$ non empty attribute sets. Therefore, determining G^* composed of k decision trees will be a very complex combinatorial problem since $\mathcal{O}(k * n^n * 2^m)$ possible solutions will be available. This problem can be tackled by using meta-heuristic strategies (such as genetic algorithms, particle swarm optimization, *etc.*). These techniques have been widely used as powerful optimization algorithms for finding an "optimal" solution in multiple complex combinatorial problems (Dhifli, Karabadjji and Elati, 2020).

In this paper, we present a genetic algorithm-based approach, termed GA_RF, for optimizing the construction of a RF model. This approach mainly focuses on increasing the accuracy and the diversity of the generated decision trees within a RF while also minimizing their numbers. The major contributions of this paper could be summarized as follows:

1. A binary encoding schema is proposed to consider all possible candidate trees. Then, an optimal RF configuration is selected according to a multi-objective function.
2. Four metrics are defined to evaluate the individual accuracy and diversity at the tree level during the forest construction.
3. An experimental evaluation on multiple datasets from the UCI Machine Learning Repository (Dua and Graff, 2017) is conducted and empirically proves the efficiency of the proposed approach.
4. The superiority of our approach compared to state-of-the-art as well as evolutionary RF construction approaches is empirically shown.
5. Finally, a real-world cybersecurity application is used to demonstrate how the approach could be applied to generate a robust RF for detecting Botnet traffic in a Internet of Things environment.

The rest of the paper will be organized as follows. Section 2 presents related work from the literature. Section 3 gives the preliminaries and problem formalization. Section 4 describes our approach for evolutionary mining of an optimal RF. Section 5 reports the evaluation and experimental results on benchmark datasets from UCI repository. It also shows a comparison with state-of-the-art classifiers. In Section 6, we present a practical application for detecting malicious traffic related to botnet activity in Internet of Things environment. Finally, Section 7, concludes the paper and discusses our future work.

2. Related work

RF is among the most popular algorithms in machine learning. The idea introduced by Ho (Ho, 1995) then improved in (Breiman, 2001) is still considered as a reference algorithm for the construction of RFs. Indeed, it is commonly used to develop and benchmark many real-world applications including Gene Expression (Walker, Cliff, Romero, Shah, Jones, Gazolla, Jacobson and Kainer, 2022), Forecasting Bitcoin price (Syed Abul and Perry, 2022), fault classification in industrial processes (Zheng, Liu and Ge, 2022), text classification (Jalal, Mehmood, Choi and Ashraf, 2022), *etc.*

Several approaches have been proposed in the literature for the construction of RFs (Biau and Scornet, 2016). In (Rokach, 2016), the authors have reviewed popular methods for generating RFs by merging and reducing individual tree outputs, and decreasing the size of large forests.

GARF (Bader-El-Den and Gaber, 2012) is an evolutionary approach that uses a genetic algorithm-based strategy to improve RF accuracy. The results have revealed that this method performs better than AdaBoost (Freund and Schapire, 1997) and other classification methods.

OptimizedForest (Adnan and Islam, 2016) is a sub-forest selection technique that minimizes the number of trees while keeping the correct classification rate high. It used a genetic algorithm where the initial population consists of individual high quality trees carefully selected to improve the efficiency of the constructed forest. The authors evaluated their approach on 20 datasets from the UCI repository. They concluded that the sub-forest selected by their approach is smaller than the original forest with a better (or comparable) classification performance.

In (Adnan and Islam, 2017), the same authors proposed PAForest, a new RF construction algorithm that uses a weighting strategy on all predictive attributes to generate trees that are individually accurate. To ensure a high diversity between the individual trees, this algorithm imposes penalties on the attributes that have been used in the last constructed tree when generating a new one.

In (Siers and Islam, 2015), the authors proposed CSForest. This approach is based on a cost-sensitive voting called CSVoting. It finds an optimized set of decision trees by minimizing the classification cost.

Another approach called BDF for decision forest construction was proposed in (Adnan, Ip, Bewong and Islam, 2021). It combines the number of unique samples and the subspace size to construct individual trees that simultaneously are accurate and diverse. This approach starts by randomly initializing a variable based on which the percentage of unique records is computed. Then, a subset of these records is randomly reselected using bootstrap sampling. Finally, the sample space is constructed from both the selected and the reselected records.

Recently, (Ganaie, Tanveer, Suganthan and Snasel, 2022) proposed two methods: oblique and rotational double random forests using several techniques. In both methods, the data at each node is transformed via different transformations using principal component analysis, linear discriminant analysis, or multi-surface proximal support vector machine. Experimental evaluation over 121 UCI datasets have shown the efficacy of the proposed approaches.

In (Bai, Li, Li, Yang, Jiang and Xia, 2022), a new RF framework called multinomial random forest (MRF) was proposed. In this approach, two multinomial distributions based on impurities are used to pick at random both a splitting feature and a specific splitting value. Several experiments over 24 UCI dataset have been performed and have shown that MRF has comparable performance to the standard RF.

3. Preliminaries and definitions

This section presents preliminary definitions and basic terminologies about the powerset system. The construction of a RF is based on the generation of k decision trees using random sampling with replacement on both the training set Tr and the attributes set A . This sampling scheme consists of a selection of a subset S_i composed of n' instances/attributes with $n' \leq |Tr|$ ($n' \leq |A|$). On the training set, the process consists of selecting a subset of instances S_i for which a given instance may appear multiple times and that is not the case for attributes.

Definition 1. (Powerset system). Let X be a finite set. The powerset, denoted \mathcal{P}_X , is a partially ordered set composed of all the subsets of X .

The set of potential solutions of the attributes selection from A is \mathcal{P}_A . In addition, Tr_i can be represented by an ordered subset of (samples) S_i and their respective repetition sequence R_i . In fact, R_i represents the number of repetitions of each instance within the potential solution S_i where $S_i \in \mathcal{P}_{Tr_i}$.

The powerset is ordered hierarchically according to the size of its subsets (see Figure 1). Moreover, in each level, all the subsets are ordered (*i.e.*, they form a chain). Thus, according to these two orders, each subset $Y \subseteq X$ in the powerset \mathcal{P}_X can easily be identified using an index couple (L_i, id) where L_i is the size of Y (see Figure 1) and id is its position in the range. In other words, the couple (L_i, id) indicates the range $|Y|$ and the position of Y in this range, respectively. Example 1 provides an idea about our powerset system.

Example 1. Let $X = \{1, 2, 3, 4, 5\}$ be a set of five elements (*i.e.*, instances or attributes). In Figure 1, the hierarchy illustrates the powerset \mathcal{P}_X where:

1. The subsets Y_i are arranged by their size from 0 to $|X| = 5$;
2. At each level, subsets Y_i are indexed by their position at this level (*i.e.*, chain).

Thus, according to this representation, each subset can be identified by its size and position (L_i, id) . For instance, the couples $(1, 4)$, $(2, 6)$, $(3, 3)$, and $(4, 0)$ represent the subsets $\{5\}$, $\{2, 5\}$, $\{1, 3, 4\}$, and $\{1, 2, 3, 4\}$, respectively.

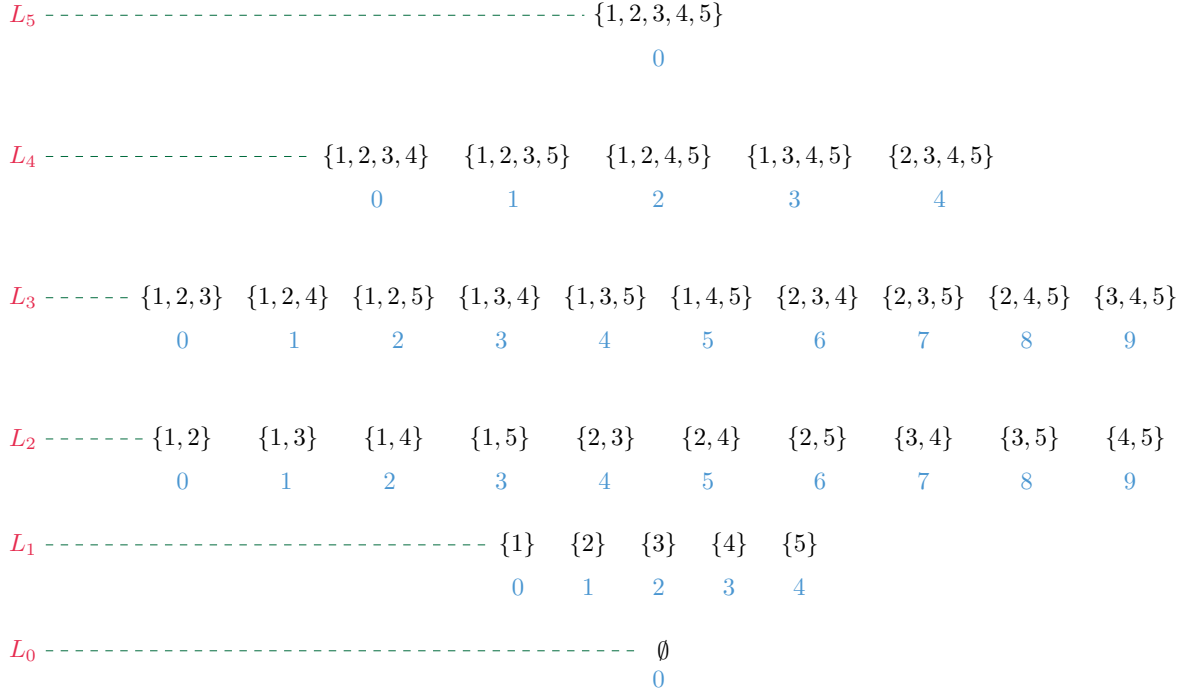


Figure 1: An illustrative example of a powerset system.

In the case of instance samples, we additionally need a sequence of integers R_i representing the number of repetitions of each instance of the subset S_i . The number of instances n' must be equivalent to that of the original training set Tr where $|Tr_i| = |Tr| = n$. Thus, the generation problem of R_i is a partition of a positive integer n with respect to $|S_i|$. The possible ways of writing n as a sum of k positive integers is $p(n, |S_i|) = \binom{|n|-1}{|S_i|-1}$ when considering two sums that differ only in the order of their summands. Therefore, for each S_i , there is $p(n, |S_i|)$ possible repetition sequences which form a chain.

Example 2. Let the instances set be $S_i = \{1, 3, 5\}$, $n = 5$ and $p(n, |S_i|) = 6$ for which the sequences of possible repetitions are (1, 1, 3), (1, 2, 2), (1, 3, 1), (2, 1, 2), (2, 2, 1), and (3, 1, 1). The repetition (1, 3, 1) refers to the case where we have 1 repetition for the instance 1, 3 repetitions for the instance 3 and 1 repetition for the instance 5.

In addition to the \subseteq order between subsets at different levels, three other linear orders are defined. The first order, denoted by \leq , is defined over the subsets of the same level (powerset level). It is introduced as a lexicographic order between training samples/attributes in the same subset where $S_v \leq S_w \Leftrightarrow v \leq w$ and $A_h \leq A_g \Leftrightarrow h \leq g$. The second linear order, denoted by \preceq , introduces a linear lexicographic order between ordered sample/attribute sets having equivalent sizes. An ordered samples/attributes set $S_v = \{s_{v_0}, s_{v_1}, s_{v_2}, \dots, s_{v_t}, \dots, s_{v_n}\}$ is lexicographically more general than an ordered samples/attributes set $S_w = \{s_{w_0}, s_{w_1}, s_{w_2}, \dots, s_{w_t}, \dots, s_{w_n}\}$, if and only if:

$$\exists t, 0 \leq t \leq n, s_{v_k} = s_{w_k} \text{ and } s_{v_t} \leq s_{w_t}, \forall k < t \quad (1)$$

The third order, denoted by \triangleleft , is defined between the sequences of the possible repetitions. It is imported as a lexicographic order between repetition sequences where $R_i = \{r_{v_0}, r_{v_1}, r_{v_2}, \dots, r_{v_t}, \dots, r_{v_n}\}$ is lexicographically more general (i.e., \triangleleft) than $R_w = \{r_{w_0}, r_{w_1}, r_{w_2}, \dots, r_{w_t}, \dots, r_{w_n}\}$, if and only if:

$$\exists t, 0 \leq t \leq n, r_{v_k} = r_{w_k} \text{ and } r_{v_t} \triangleleft r_{w_t}, \forall k < t \quad (2)$$

4. Evolutionary mining of an optimal random forest

The main objective of this work is to build a RF optimized in terms of classification performance, number of trees in the forest, and diversity between these trees. This objective will be achieved by using a group of appropriate couples of sets of learning samples and attributes. These couples must be carefully selected from a very large number of possibilities. This Section illustrates a genetic algorithm-based approach to select an optimal group of these couples. This approach includes the following phases. First, the encoding and decoding procedures are introduced. Then, a fitness function is defined. Finally, a genetic optimization phase is adopted to discover the optimal group.

4.1. Encoding and decoding of chromosomes

This phase is a key part of the genetic algorithm-based solution. It allows to make an appropriate representation of the candidate solutions for the genetic algorithm. In the following subsections, the encoding and decoding of chromosomes are detailed.

4.1.1. Encoding

The encoding phase consists of defining a one-to-one function that represents each potential solution as a binary sequence (*i.e.*, binary string). As described in Section 1, a RF can be represented by a set \mathcal{G} consisting of k couples of sets of training samples Tr_i and attributes A_j used to generate the corresponding decision trees. Formally, $\mathcal{G} = \{(Tr_a, A_b)_1, (Tr_x, A_y)_2, \dots, (Tr_i, A_j)_k\}$. To identify each couple in \mathcal{G} , three decision variables are required for the training samples and two for the attributes (see Figure 2). For a training set, the three variables are: the size L_i of the samples set without repetition (denoted by S_i), its identifier id_{S_i} , and the identifier id_{R_i} of the repetition sequence R_i . For the set of attributes, the two variables are: the size L_i of the set A_i , and its identifier id_{A_i} . In summary, $5 * k$ variables are required to represent a RF consisting of k decision trees.

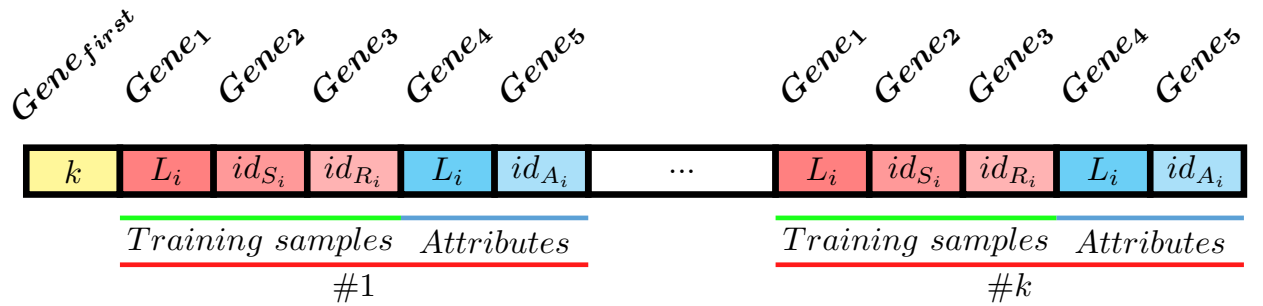


Figure 2: Representation of a chromosome encoding a RF.

In the following, a binary encoding of information is defined regarding the subsets representing each of the trees as well as the size of the forest. Each chromosome is represented by an array of 0's and 1's (*i.e.*, a binary string). Each decision tree within the RF is encoded on 5 genes. For example, in Figure 2, the first gene (*i.e.*, $Gene_{first}$) on the chromosome encodes information about the number of decisions trees in the forest (*i.e.*, k decisions trees).

The genes encode different pieces of information, each one will use X_i bits. The total number of bits depends on the maximal size of the forest, the number of the training samples $|Tr|$ and the size of the set of attributes A . More precisely:

- $Gene_{first}$ represents the number of decision trees which is presented as the binary representation of an integer k in $[\lambda .. \mu]$, where λ and μ are respectively the minimal and the maximal sizes of the RF model. To encode k , X_{first} bits are used where X_{first} is the minimum required number of bits to encode μ .
- $Gene_1$ represents the size L_i of the subset S_i of training samples without replacement. L_i is represented by a binary representation of integers in $\left[\frac{|Tr|}{3} .. |Tr|\right]$. To encode L_i , X_1 bits are used where X_1 is the minimum required number of bits to encode $|Tr|$.

- $Gene_2$ represents the indicator integer id_{S_i} of the subset of training samples S_i without replacement. To encode id_{S_i} , X_2 bits are used where X_2 is the minimum required number of bits to encode $\left(\frac{|Tr|}{2}\right)$ (i.e., the highest value of id_{S_i}).
- $Gene_3$ represents the indicator id_{R_i} of the repetition sequence R_i . This indicator requires X_3 bits which is the minimum required number of bits to encode $p(|Tr|, |S_i|)$ (i.e., the highest value of id_{R_i}).
- $Gene_4$ and $Gene_5$ are used to encode the subset of attributes A_i . These genes require X_4 and X_5 bits, respectively. X_4 encodes the size L'_i of A_i which is encoded by a binary representation of integers in $[1 .. |A|]$. X_5 is the minimum required number of bits to encode the indicator id_{A_i} of A_i without replacement that can at most be equal to $\left(\frac{|A|}{2}\right)$.

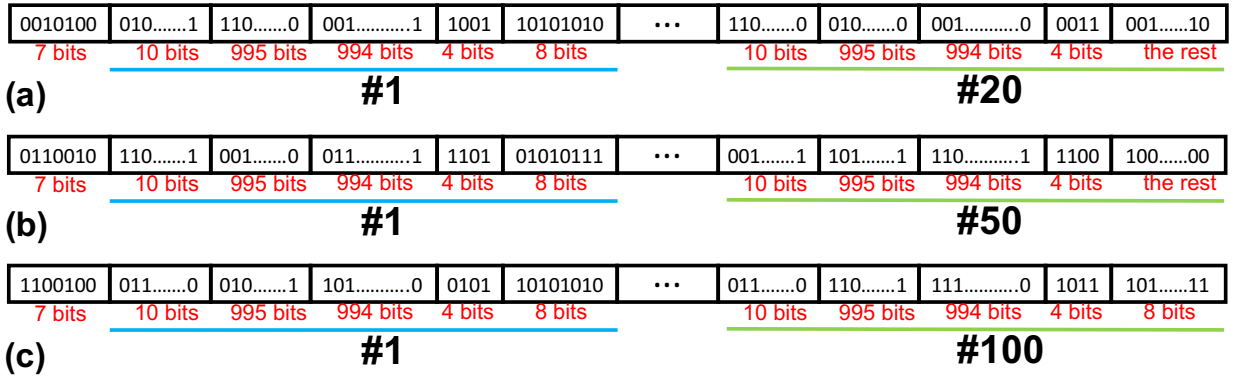


Figure 3: An example of three chromosomes encoding three RFs with 20, 50 and 100 trees.

Example 3. Assume that $\lambda = 10$ and $\mu = 100$ (the minimum and maximum sizes of the RF) and a set of training samples Tr with $|Tr| = 1000$ described using a set of attributes A with $|A| = 10$. The required bits, to encode each decision tree in the forest when all possibilities are considered, are equivalent to $\log_2\left(\left(|Tr| + \left(\frac{|Tr|}{2}\right) + p(|Tr|, \left(\frac{|Tr|}{2}\right))\right) + \left(|A| + \left(\frac{|A|}{2}\right)\right)\right)$. Thus, for encoding the complete information and considering all possibilities at most $\log_2(\mu) + \mu * \log_2\left(\left(|Tr| + \left(\frac{|Tr|}{2}\right) + p(|Tr|, \left(\frac{|Tr|}{2}\right))\right) + \left(|A| + \left(\frac{|A|}{2}\right)\right)\right)$ bits are required. In this setting, encoding each tree requires 2008 bits. Indeed, the five genes are encoded as follows: X_{first} requires 7 bits, X_1 requires 10 bits, X_2 requires 995 bits, X_3 requires 994 bits, X_4 requires 4 bits, and X_5 requires 5 bits.

Figure 3 illustrates an example of 3 different chromosomes encoding 3 RFs with 20, 50 and 100 trees with respect to the input setting cited in Example 3. The main difference between the three chromosomes (a), (b) and (c) is the number of bits allowed to encode $Gene_5$ used for encoding the last individual tree (i.e., tree #20, #50 and #100). For the chromosome (c) encodes 100 trees which is equivalent to the maximal number of trees (i.e., $\mu = 100$), 8 bits are used to encode $Gene_5$ of the tree #100. However, for the chromosomes (a) and (b) more bits are used to encode the $Gene_5$. Indeed, the number of bits of the latter, denoted by *the rest*, is composed of 160 888 and 100 558 bits, respectively for the chromosomes (a) and (b). In this way, all the chromosomes will have the same length. This is necessary for a proper functioning of the genetics operations (i.e., crossover and mutation).

4.1.2. Decoding

The decoding phase converts the values of genes as follows:

- 1) Identify the number k of decision trees by converting $Gene_{first}$ to $\lambda + Gene_{first} \text{ modulo } (\lambda - \mu)$.
- 2) The five genes identify the used couple of training instances Tr and attributes A representing each tree within the forest (see Section 4.1.1), such that:
 - a) $Gene_1$ is composed of X_1 bits (i.e., $X_1 = \log_2(|Tr|)$). Decoding $Gene_1$ gives an integer $L_i \in \left[\frac{|Tr|}{3}..|Tr|\right]$ defined as follows: $L_i = \frac{|Tr|}{3} + \text{Integer_In}(X_1) \text{ modulo } \left((|Tr| + 1) - \frac{|Tr|}{3}\right)$. The function Integer_In converts the binary string in X_1 into an integer.
 - b) $Gene_2$ is composed of X_2 bits. Decoding $Gene_2$ gives an integer in the interval $\left[0.. \binom{|Tr|}{L_i} - 1\right]$ defined as follows: $Id_{S_i} = \text{Integer_In}(X_2) \text{ modulo } \left(\binom{|Tr|}{L_i}\right)$.
 - c) $Gene_3$ is composed of X_3 bits. It encodes an integer Id_{R_i} that allows identifying the repetition sequence of instances R_i . Decoding $Gene_3$ gives an integer in the interval $\left[0..p(|Tr|, L_i) - 1\right]$ as follows: $Id_{R_i} = \text{Integer_In}(X_3) \text{ modulo } \left(p(|Tr|, L_i)\right)$.
 - d) $Gene_4$ is composed of X_4 bits. It encodes an integer L'_i that represents the size of the set of attributes A_i . Decoding $Gene_4$ gives an integer in the interval $\left[1..|A|\right]$ as follows: $L'_i = 1 + \text{Integer_In}(X_4) \text{ modulo } (|A| - 1)$.
 - e) $Gene_5$ is composed of X_5 bits. It encodes an integer Id_{A_i} that allows identifying a subset of attributes A_i . Decoding $Gene_5$ gives an integer in the interval $\left[0.. \binom{|A|}{L'_i} - 1\right]$ defined as follows: $Id_{A_i} = \text{Integer_In}(X_5) \text{ modulo } \left(\binom{|A|}{L'_i}\right)$.

Algorithm 1: | Generating training samples/attributes subset

Input: ID_S , k and P .

Output: An array of integers S_i/A_i of length k .

```

1:  $E[k] \leftarrow \{1..1\}$ ;
2:  $ID = 0$ ;
3: for ( $a = 1 \rightarrow k$ ) do
4:    $cID \leftarrow ID$ ;
5:    $x \leftarrow 0$ ;
6:   while ( $cID \leq ID_S$ ) do
7:      $ID \leftarrow cID$ ;
8:      $e \leftarrow \binom{P - (E[a-1] + x)}{k-a}$ ;
9:      $cID \leftarrow cID + e$ ;
10:    if ( $cID \leq ID_S$ ) then
11:       $x \leftarrow x + 1$ ;
12:    end if
13:  end while
14:   $E[a-1] \leftarrow E[a-1] + x$ ;
15:  if ( $a-1 < k-1$ ) then
16:     $E[a] \leftarrow E[a-1] + 1$ ;
17:  end if
18: end for
19: return  $E$ ;

```

According to the found identifiers, couples of sets of instances and attributes will be generated. First, a training sample Tr_i will be generated by using L_i , Id_{S_i} and Id_{R_i} as follows: 1) A set S_i of instance indexes is generated by

using the Algorithm 1 and the equation 3 (E is an array that represents a set S_i). 2) The sequence R_i of repetition of instances is generated by using the Algorithm 2 and the equation 3 (R_i is an array that represents a sequence R_i). Second, a set of attributes is generated based on its two identifiers L'_i and Id_{A_i} by using the Algorithm 1 and the equation 3 (E is a subset array that represents a set A_i).

we start from the first subset S_0 i.e., E_0 with $ID = 0$ (presented as an array of length L_i (respectively L'_i) in the range L_i (respectively L'_i)). We identify all the elements one by one. From $E_i[1]$ to $E_i[k]$, each element will be revised if there is a given $x > 0$ for which the equation 3 is valid, where $E_i[a] = E_i[a] + x$.

$$ID + \sum_{x=1} \frac{(n - (E[a] + x))!}{(k - a)! \times ((n - (E[a] + x)) - (k - a))!} \leq ID_E \quad (3)$$

Example 4. Given a dataset D with 20 samples and 10 attributes, and the two following decision trees $x_1 = (10, 175500, 50000, 5, 243)$ and $x_2 = (5, 10500, 1000, 3, 115)$. Each decision tree is encoded in a given chromosome. The subsets of training samples and attributes are composed of:

1. 10 and 5 distinct samples, respectively; but the resultant training sets are composed of 20 samples with the repetition.
2. 5 and 3 attributes, respectively.

Following the Algorithm 1 and according to the x_1 identifiers $Id_{S_1} = 175500$ and $Id_{R_1} = 10$, $S_1 = \{4, 7, 8, 11, 14, 16, 17, 18, 19, 20\}$ and $A_1 = \{4, 6, 7, 9, 10\}$ will be generated. Moreover, with respect to the repetition sequence identifier $Id_{R_1} = 10$ and following Algorithm 2, $R_1 = (3, 2, 1, 2, 1, 2, 11, 9, 2, 3, 1, 3, 9, 1, 5, 8, 6, 9, 13, 9)$. For x_2 , $S_2 = \{4, 8, 10, 14, 15\}$, $A_2 = \{5, 8, 9\}$ and $R_2 = (2, 2, 6, 5, 5)$.

Algorithm 2: | Generating training samples repetition sequence

Input: ID_R , k and $|Tr|$.

Output: An array of integers R_i of length k .

```

1:  $R_i[k] \leftarrow \{1..1\}$ ;
2:  $ID \leftarrow 0$ ;
3:  $sum \leftarrow 0$ ;
4: for ( $j = 0 \rightarrow k - 2$ ) do
5:    $cID \leftarrow ID$ ;
6:   for ( $i = 1 \rightarrow |Tr| - (sum + (k - (j + 1)))$ ) do
7:      $cID \leftarrow cID + (p(|Tr| - sum - i, k - (j + 1)))$ ;
8:     if ( $cID \leq ID_R$ ) then
9:        $R_i[j] \leftarrow i + 1$ ;
10:       $ID \leftarrow cID$ ;
11:    else break;
12:    end if
13:  end for
14:   $sum \leftarrow sum + R_i[j]$ ;
15:  if ( $cID = ID_R$ ) then break;
16:  end if
17: end for
18:  $R_i[k - 1] \leftarrow |Tr| - sum$ ;
19: return  $R_i$ ;
```

4.2. Evaluation of chromosomes

To evaluate the candidate solutions, a fitness function is required. This function will be evaluated on all chromosomes at each iteration. Only the best chromosomes will be selected for the next iteration. Each chromosome will be evaluated in terms of its ability to generate a small number of accurate and diverse decision trees. Given a set of decision trees \mathcal{G} encoded in a chromosome ch , the fitness function is defined as follows:

$$fitness(ch) = \alpha * ACC(\mathcal{G}) + \beta * APDT(\mathcal{G}) + \gamma * \frac{\mu - k}{\mu - \lambda} \quad (4)$$

where ACC is the accuracy of \mathcal{G} ; α , β , and γ are three constants in $[0, 1]$ with $\alpha + \beta + \gamma = 1$. Each decision tree in \mathcal{G} is evaluated on an independent test set. The Average Predictive Tree Diversity ($APT D$) represents the average diversity score over the predictions on the test set (see Section 5.3). This score is computed as the average of the Kappa value of each decision tree with all the other ones in \mathcal{G} . The component $\frac{\mu - k}{\mu - \lambda}$ of the equation 4 promotes the chromosomes with lower number of trees. The fitness function takes values in $[0, 1]$.

4.3. Genetic operators

In this Section, the genetic operators used in our approach (GA_RF) will be presented.

Crossover. This operator is applied on two randomly chosen individuals from the population. As a result, it gives a chromosome formed from both parents. Two offsprings are generated for the next generation. In our approach, the percentage of crossovers is set to 50% and cross at a point is applied.

Mutation. This operator is applied to an individual by modifying one or more randomly selected genes. The modified genes are chosen randomly from the parent chromosome forming one new child. The mutation percentage in our case is set to 1%. This ratio defines the probability of changing one bit by another (0 by 1 or 1 by 0) at random and with no interaction with any other chromosome.

4.4. The stopping criteria

The genetic algorithm stops at a predefined maximum number of evaluations that is set to 10000 in this work.

4.5. Complexity analysis

This section presents the complexity analysis of the proposed approach (GA_RF). This approach comprised a number of phases, each is performed M times, where M is the number of iterations of the GA. Decoding, computing fitness, going through the selection process, and carrying out genetic operations are the phases that make up this process. The chromosome decoding phase uses the majority of the computing cost. Let C_1 and C_2 be the amounts of time needed to learn and evaluate a particular decision tree model, and k be the largest size a RF can have. Let also θ be the decoding time of a chromosome. In the worst case scenario, the computation of the fitness phase will take $\mathcal{O}(\theta + k * (C_1 + C_2))$. For the selection phase, the problem can be conceptualized as an urn problem, in which $\frac{N}{2}$ of the total population is participating in the drawing. In the worst case scenario, the complexity of this phase is bounded by $\mathcal{O}(N^2)$. The application of genetic operators requires a complexity of $\mathcal{O}(N)$. Overall, the complexity of the proposed GA_RF approach is $\mathcal{O}(N^2 * (\theta + k * (C_1 + C_2)))$ in the worst case scenario.

5. Experiments and discussion

To examine the effectiveness of the proposed approach GA_RF, several experiments are conducted. In the following, the datasets used for the experiments, the baseline approaches that GA_RF compared to, and the obtained results will be described.

5.1. Datasets and experimental settings

Our approach is implemented in Java using WEKA (Hall, Frank, Holmes, Pfahringer, Reutemann and Witten, 2009) and jMetal (Durillo and Nebro, 2011) frameworks. In order to test the effectiveness of this approach, we evaluate it on 15 datasets from UCI (Dua and Graff, 2017) using 10 cross-validation. Table 1 reports the statistics about the considered datasets where $\#Classes$ shows the number of classes, $\#Attributes$ shows the number of attributes,

Table 1
Datasets statistics

Datasets	#Classes	#Attributes	#Instances	Majority Class Probability
Balance-scale (BS)	3	4	625	46.08
Breast-cancer (BC)	2	10	286	70.27
Credit-a (CR)	2	15	690	55.50
Dermatology (DE)	6	35	366	30.60
Diabetes (DI)	2	8	768	65.10
Glass (GL)	6	9	214	35.51
Heart-statlog (HS)	2	13	270	55.55
Ionosphere (ION)	2	34	351	64.10
Iris (IR)	3	8	150	33.33
Liver-disorders (LD)	2	6	345	57.97
Lung-cancer (LU)	3	56	27	48.14
Seeds (SE)	3	7	210	33.33
SCADI(SC)	7	205	70	41.42
Teaching Assistant Evaluation (TAE)	3	5	151	34.43
Vehicle (VE)	4	18	846	25.76

Table 2
Comparison of the 10-CV classification accuracy results (in percentage) of our approach (GA_RF) and state-of-the-art approaches.

Datasets	RF	OptimizedForest	ForestPA	CSForest	BDF	MRF	GA_RF
BS	81.76	81.60	85.12	80.80	82.10	13.58	86.89
BC	69.93	70.27	73.07	37.41	80.00	76.55	75.22
CR	86.52	86.95	86.95	85.50	86.70	85.07	91.59
DE	96.99	96.17	98.36	80.60	87.00	96.44	81.41
DI	76.69	75.39	74.86	66.01	76.70	65.10	80.85
GL	74.10	80.37	74.29	64.95	74.50	33.20	60.03
HS	83.70	80.74	82.96	70.37	83.00	81.11	90.07
ION	93.70	92.87	92.87	88.31	93.70	64.09	88.60
IR	96.66	95.33	96.00	92.66	96.00	78.66	98.00
LD	73.04	74.20	73.33	60.28	72.70	57.98	80.28
LU	74.07	74.07	77.77	70.37	83.90	71.66	87.49
SE	94.28	94.28	91.90	91.90	92.90	25.23	96.18
SC	84.28	84.28	85.71	84.28	83.30	82.85	82.85
TAE	68.87	67.54	56.95	50.33	56.30	43.16	68.96
VE	76.00	76.24	75.17	72.33	75.00	57.33	62.88

#Instances illustrates the number of examples in each dataset and *Majority Class Probability* refers to the accuracy percentage obtained if the most frequent class is always predicted.

5.2. Comparison with existing RF approaches

Table 2 reports the classification accuracy results of our approach compared to the basic RF implementation (with 100 trees) in WEKA, OptimizedForest (Adnan and Islam, 2016), ForestPA (with 100 trees) (Adnan and Islam, 2017), CSForest (with 100 trees) (Siers and Islam, 2015), BDF (with 100 trees) (Adnan et al., 2021), and MRF (with 100 trees) (Bai et al., 2022) on each of the considered datasets. In this experiment, the three constants α , β and γ are fixed at 0.8, 0.19 and 0.01 respectively. The obtained results clearly indicate that our approach is generally more accurate than all the other competitors in terms of classification accuracy. Using our approach to build a RF leads to a significant improvement on the 15 datasets compared to the basic RF approach as well as to the other approaches which outperform ours at most only on 5 datasets in terms of one to one comparison. These losses could be due to the

Table 3

Pairwise win–tie–loss count of GA_RF.

	RF	OptimizedForest	ForestPA	CSForest	BDF	MRF
GA_RF	[10, 0 ,5]	[10, 0 ,5]	[10, 0 ,5]	[12, 0 ,3]	[9, 0 ,6]	[12,1,2]

fact that our algorithm converged to a solution with a low number of very diverse trees which could make the model unstable. A more fine tuning of the fitness function parameters for each of these datasets could help in boosting their accuracy results. Table 3 shows a summary of how many times GA_RF compared to other RF classification models won, tied, or lost. One can see that the proposed RF had more wins than the models that were already in place.

5.3. Optimal RF characteristics

In Table 4, we evaluate the individual accuracy and diversity at the tree level for the forest constructed by our approach. For this purpose, the following measures are computed:

1. **Average Tree Accuracy (ATA):** It quantifies the average of the accuracies (in percentage) for each tree in the forest.
2. **Average Tree Diversity (ATD):** It assesses the average diversity (using Kappa measure) of the couple of training and attribute sets used to construct each tree of the forest. ATD takes values in [0, 1].
3. **Average Predictive Tree Diversity (APTD):** It is the average diversity (using Kappa) of the predictions (predicted classes) of each tree on the test set. APTD is in [0, 1].
4. **Average Number of Selected Decision Trees (ASDT):** It shows the average number of remaining decision trees that were selected by our approach over the cross-validation evaluations to construct the RF.

Compared to the obtained values of ATA in Table 4, the accuracy values of GA_RF in Table 2 are much higher which confirms the advantage of using the RF strategy. Despite the high accuracy of the RF models constructed by our approach on the experimental datasets, we notice significantly low values of APTD which reflects a high diversity between the individual trees in each model. Note that APTD and ATD are both based on the Kappa measure which means that the lower is the value the more is the diversity between the individual trees of the model. Note also that it is expected to have high values of ATD since the training sets used by each tree in the forest share a large number of the training samples and attributes. We also notice in ASDT that the average values of selected trees per forest are significantly lower than the initial maximum number of trees which was set to 100 in our experiments. This emphasizes the high selectivity of the fitness used in our approach. Indeed, the size of the RF was reduced with an average reduction of 33% compared to the 100 trees generated by the other methods.

5.4. Comparison with other state-of-the-art classifiers

To further assess the classification efficiency of our approach, a comparison with other existing state-of-the-art classifiers is conducted. Three of the most widely used classifiers are used namely: Naive Bayes (NB), Multilayer Perceptron (MLP), and Support Vector Machine (SVM). The implementation in WEKA with the default settings for each of the three classifiers is used. For SVM, a polynomial kernel (SVM(POLY)) and a radial basis function kernel (SVM(RBF)) are used.

Table 5 shows the obtained accuracy results. Indeed, it clearly demonstrates the robustness of the our approach compared to these well established classifiers. Our approach outperforms the other ones in 9 out of the used 15 datasets. However, NB, MLP and SVM(RBF) scored best respectively for 1, 4 and 1 dataset.

6. Application to the detection of Botnet traffic in Internet of Things environment

The Internet of Things (IoT) is a network of interconnected digital, mechanical, and computing objects that can transmit data without human intervention. IoT based solutions deliver significant data and insights that improve the way we work and live. It has a wide range of applications, for instance, for enhancing the safety of roads, automobiles, and houses to fundamentally reshaping the way we create and consume things. However, the fast spread of unsecured, poorly configured and flawed IoT devices represent privileged attack vectors for hackers. Bot malwares are among the most prevalent security risks hindering the deployment of IoT. Indeed, a bot is a piece of malware that infects a

Table 4

10-CV results of our approach in terms of individual tree accuracy (ATA), diversity (ATD and APTD) and number of selected trees (ASDT).

Datasets	ATA	ATD	APTD	ASDT
BS	63.02	0.80	0.18	72.40
BC	59.74	0.76	0.15	61.60
CR	66.82	0.77	0.11	73.80
DE	58.73	0.76	0.14	79.80
DI	67.03	0.80	0.24	84.80
GL	42.05	0.79	0.15	61.40
HS	62.36	0.79	0.06	70.60
ION	74.16	0.75	0.18	63.60
IR	62.05	0.79	0.01	55.00
LD	55.08	0.78	0.05	62.80
LU	75.86	0.66	0.46	23.10
SE	87.39	0.76	0.38	23.90
SC	76.72	0.63	0.63	22.50
TAE	51.13	0.74	0.29	23.60
VE	49.71	0.76	0.11	76.40

Table 5

Comparison of the 10-CV classification accuracy results (in percentage) of our approach (GA_RF) with NaiveBayes, MLP and SVM approaches.

Datasets	NaiveBayes	MLP	SVM(poly)	SVM(RBF)	GA_RF
BS	90.4	90.72	87.68	86.88	86.89
BC	71.67	64.68	69.58	70.27	75.22
CR	77.68	83.62	84.92	85.50	91.59
DE	97.26	96.17	95.35	96.99	81.41
DI	76.30	75.39	77.34	65.10	80.85
GL	48.59	67.75	56.07	35.51	60.03
HS	83.70	78.14	84.07	82.59	90.07
ION	82.62	91.16	88.60	76.63	88.60
IR	96.00	97.33	96.00	89.33	98.00
LD	55.36	71.59	58.26	89.33	80.28
LU	78.12	65.62	65.62	71.87	87.49
SE	91.42	95.23	93.80	89.52	96.18
SC	82.85	80.00	80.00	80.00	82.85
TAE	54.30	54.30	54.30	34.43	68.96
VE	44.79	81.67	74.34	39.83	62.88

computer and executes commands under the attacker's remote control. A botnet (short for "robot network") is a network of compromised computers controlled by a single hacker (bot master) via the command-and-control server (C&C).

Due to their large number and ubiquity, IoT devices present a tempting target for hackers to expand their botnets. An infected IoT device belonging to the botnet is called a thingbot. IoT Botnets were usually exploited to launch large scale distributed denial-of-service attacks. Multiple major DDoS attacks have occurred since 2016, most of them using Mirai malware (McDermott, Majdani and Petrovski, 2018) installed on a large number of IoT devices. C&C network communication represents a particular type of a botnet traffic. Numerous researches have tackled botnet detection in IoT (Popoola, Ande, Adebisi, Gui, Hammoudeh and Jogunola, 2022; Labiod, Amara Korba and Ghoualmi, 2022; Nguyen, Dumba, Ngo, Le and Nguyen, 2022). Most of them have focused on the detection of botnets traffic in the attack phase. However, only a few works have investigated the detection of C&C traffic (McDermott et al., 2018; Bezerra, da Costa,

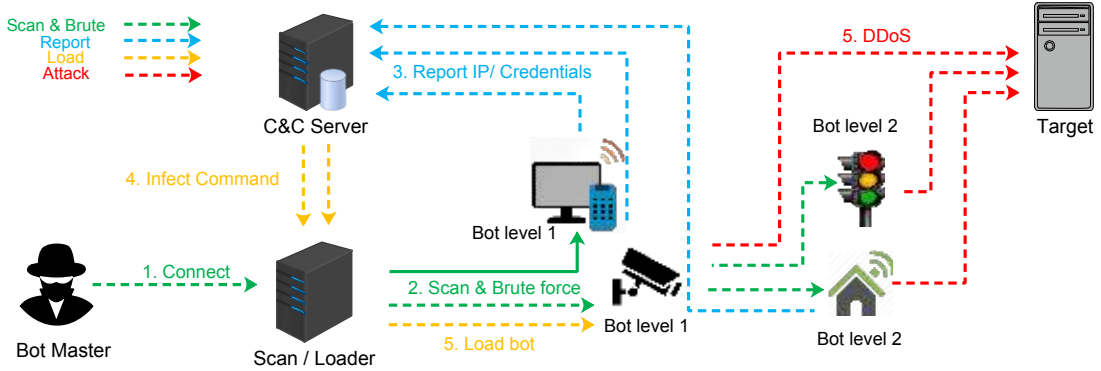


Figure 4: Botnet infection process.

Barbon Junior, Miani and Zarpelão, 2019). Identifying C&C traffic will allow for early ThingBot detection, and thus better attack mitigation.

In this application, we analyze and model the IoT network traffic to detect thingbots. We focus on detecting one of the most prominent IoT Bot malware called Mirai. We are particularly interested in detecting the C&C communication to ensure an early detection of thingbots, and thus mitigate massive DDoS attacks. In this paper, we propose a flow based detection approach, combining features engineering and RF algorithm. Besides having strong generalization which is essential for detecting divers types of botnets, RF also presents a good trade-off between accuracy and interpretability. For an efficient usage of RF, it must provide high accuracy with a small number of decision trees.

In the following, we describe the conducted experiment. We detail the used features, the dataset creation, the network flow processing as well as the obtained results after application of the different algorithms.

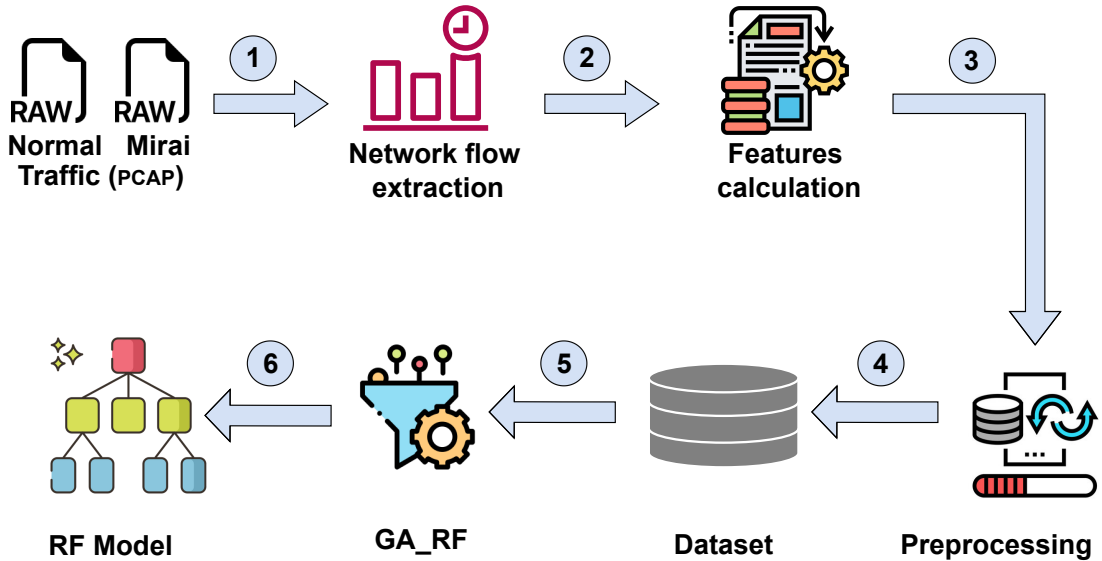


Figure 5: Description of the RF model induction phase.

Table 6
Raw labeled traffic.

No.	Time	Source	Destination	Protocol	Length	Info	Label
311	2.045079	192.168.252.40	176.209.212.31	TCP	64	1691 >2323 [SYN] Seq=0 Win=5259 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]	Mirai
312	2.045711	192.168.252.40	163.105.83.100	TCP	64	1691 >23 [SYN] Seq=0 Win=5259 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]	Mirai
313	2.046218	192.168.252.40	134.242.197.243	TCP	64	1691 >23 [SYN] Seq=0 Win=5259 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]	Mirai
314	2.046502	192.168.252.60	192.168.252.21	TCP	60	52123 >23 [ACK] Seq=1 Ack=94 Win=2049 Len=0	Normal
315	2.046507	192.168.252.60	192.168.252.21	TCP	60	[TCP Dup ACK 314#1] 52123 >23 [ACK] Seq=1 Ack=94 Win=2049 Len=0	Normal
316	2.046696	192.168.252.40	78.29.127.160	TCP	64	1691 >23 [SYN] Seq=0 Win=5259 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]	Mirai
105854	312.210427	192.168.252.50	192.168.252.40	TCP	60	32651 >19237 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	SYN Flood
105855	312.210873	192.168.252.40	192.168.252.50	TCP	74	59587 >30689 [SYN] Seq=0 Win=0 Len=0 MSS=1411 SACK_PERM=1 TSval=3332454126 TSecr=0 WS=64	SYN Flood
105856	312.210875	192.168.252.40	192.168.252.50	TCP	74	[TCP Out-Of-Order] 59587 >30689 [SYN] Seq=0 Win=0 Len=0 MSS=1411 SACK_PERM=1 TSval=3332454126 TSecr=0 WS=64	SYN Flood
105857	312.211129	192.168.252.50	192.168.252.40	TCP	60	30689 >59587 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0	SYN Flood

6.1. Experimental environment

To test the proposed approach, the network traffic collected from the testbed built by (McDermott et al., 2018) is used. The network traffic has been generated in a protected sandboxed environment. The experimental setup includes two IP cameras used as bots to attack a target, a command and control (C&C) server, a Scan/Loader server and an additional utilities server to handle DNS queries and reporting. For more details about the C&C server and Scan/Loader server configurations please refer to (McDermott et al., 2018). The network traffic collection has been carried out for five consecutive days, as 37 pcap captures, each capture took 3600 seconds. We have selected this dataset for analysis in this research due to the consideration of most recent up to date real network traffic with and without attacks, as well as C&C network traffic. Figure 4 shows the Mirai bot operating procedure to infect and to self propagate through IoT devices. After connecting to the Scan/Loader server (step 1), the bot master initiates a sophisticated SYN scanning (on specific range of IP addresses) looking for IoT devices with open Telnet/SSH ports. Then a brute-force attack using a dictionary of usual default usernames and passwords (step 2) is executed. If the brute-force succeeds, Mirai transmits the compromised IoT device's IP address and credentials to the C&C server (step 3). The IoT device's IP address and credentials are also saved to a database for future use. Then, the C&C server sends an infect command to the Scan/Loader server including the IoT device's IP address and credentials (step 4). The C&C server domain name is hardcoded and encrypted and is resolved at runtime to avoid domain name blocking. The Scan/Loader server downloads the malware bot binary to the device. Once the infection process is successful, the compromised IoT device joins the Mirai botnet and can communicate with the C&C server. From now on, the bot master can use the IoT bot to execute DDoS attack by sending to it the target and duration (step 5).

6.2. Dataset creation

Dataset preparation and preprocessing are discussed in this section. Properly documenting the process steps is important in being able to reproduce experiments. Figure 5 summarizes the dataset creation process. From the raw network traffic (collection of labeled packets as illustrated in Table 6), we extract network flows according to a s-second time-window. A flow is defined by a sequence of packets with the same values of source and destination IP, source and destination port, and layer 4 protocol (TCP/UDP). To extract and label flows, we developed scripts that identify and label flows based on the original packet-based labeling. We use CICFlowMeter (Habibi Lashkari, Draper Gil, Mamun and Ghorbani, 2017), a network traffic flow generator distributed by the Canadian Institut for Cybersecurity (CIC). It generates bidirectional flows, where the first packet determines the forward (source to destination) and backward (destination to source) directions. For each flow, a set of 64 network features are calculated, the features as well as their explanation are given in Table 7. We used source and destination IP addresses and ports for flow generation and labeling process. Then, these four environment-specific features are discarded for a more general approach. Otherwise, the classification model would not learn from the data but simply bind normal/malicious traffic to specific IP addresses and ports. Before starting machine learning, we have excluded eight features with constant values of zero for every instance [f52 - f59].

6.3. Discussion and comparison

In this section, the experimental results of the four tested algorithms namely random forest, ForestPA, CSForest, and optimized forest are discussed. All the available instances and features are used to analyze the impact of changing the time-window on the detection precision. After finding the optimal time-window size, we apply GA_RF to optimize the detection precision. Finally, our approach is compared with a recent IoT botnets detection mechanism named IoTDS.

The detection delay is critical for an intrusion detection system, it depends on the time-window (TW) size. Considering that setting smaller window allows faster detection, we have tested four time-window (1s, 5s, 10s, and

Table 7
List of network features

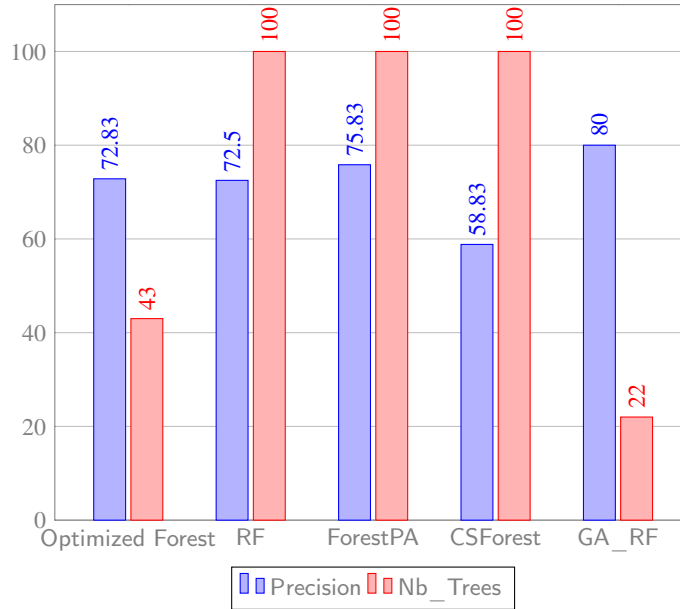
Feature identifier	Feature name	Description
<i>f1</i>	total Fwd Packet	Total packets in the forward direction
<i>f2</i>	total Bwd packets	Total packets in the backward direction
<i>f3</i>	total Length of Fwd Packet	Total size of packet in forward direction
<i>f4</i>	total Length of Bwd Packet	Total size of packet in backward direction
<i>f5</i>	Fwd Packet Length Min	Minimum size of packet in forward direction
<i>f6</i>	Fwd Packet Length Max	Maximum size of packet in forward direction
<i>f7</i>	Fwd Packet Length Mean	Mean size of packet in forward direction
<i>f8</i>	Fwd Packet Length Std	Standard deviation size of packet in forward direction
<i>f9</i>	Bwd Packet Length Min	Minimum size of packet in backward direction
<i>f10</i>	Bwd Packet Length Max	Maximum size of packet in backward direction
<i>f11</i>	Bwd Packet Length Mean	Mean size of packet in backward direction
<i>f12</i>	Bwd Packet Length Std	Standard deviation size of packet in backward direction
<i>f13</i>	Flow Byte/s	Number of flow packets per second
<i>f14</i>	Flow Packets/s	Number of flow bytes per second
<i>f15</i>	Flow IAT Mean	Mean time between two packets sent in the flow
<i>f16</i>	Flow IAT Std	Standard deviation time between two packets sent in the flow
<i>f17</i>	Flow IAT Max	Maximum time between two packets sent in the flow
<i>f18</i>	Flow IAT Min	Minimum time between two packets sent in the flow
<i>f19</i>	Fwd IAT Min	Minimum time between two packets sent in the forward direction
<i>f20</i>	Fwd IAT Max	Maximum time between two packets sent in the forward direction
<i>f21</i>	Fwd IAT Mean	Mean time between two packets sent in the forward direction
<i>f22</i>	Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
<i>f23</i>	Fwd IAT Total	Total time between two packets sent in the forward direction
<i>f24</i>	Bwd IAT Min	Minimum time between two packets sent in the backward direction
<i>f25</i>	SYN Flag Count	Number of packets with SYN
<i>f26</i>	Bwd IAT Mean	Mean time between two packets sent in the backward direction
<i>f27</i>	Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
<i>f28</i>	Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
<i>f29</i>	Fwd Header Length	Total bytes used for headers in the forward direction
<i>f30</i>	Bwd Header Length	Total bytes used for headers in the backward direction
<i>f31</i>	FWD Packets/s	Number of forward packets per second
<i>f32</i>	Bwd Packets/s	Number of backward packets per second
<i>f33</i>	Min Packet Length	Minimum length of a packet
<i>f34</i>	Max Packet Length	Maximum length of a packet
<i>f35</i>	Packet Length Mean	Mean length of a packet
<i>f36</i>	Packet Length Std	Standard deviation length of a packet
<i>f37</i>	Min Packet Length	Minimum length of a packet
<i>f38</i>	down/Up Ratio	Download and upload ratio
<i>f39</i>	Average Packet Size	Average size of packet
<i>f40</i>	Avg Fwd Segment Size	Average size observed in the forward direction
<i>f41</i>	AVG Bwd Segment Size	Average number of bytes bulk rate in the forward direction
<i>f42</i>	Fwd Header Length	Length of the forward packet header
<i>f43</i>	Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
<i>f44</i>	Avg Packet Size	Average size of packet
<i>f45</i>	Fwd AVG Bulk Rate	Average number of bulk rate in the forward direction
<i>f46</i>	Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
<i>f47</i>	Bwd AVG Packet/Bulk	Average number of packets bulk rate in the backward direction
<i>f48</i>	Bwd AVG Bulk Rate	Average number of bulk rate in the backward direction
<i>f49</i>	Init_Win_bytes_forward	The total number of bytes sent in initial window in the forward direction
<i>f50</i>	Init_Win_bytes_backward	The total number of bytes sent in initial window in the backward direction
<i>f51</i>	Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
<i>f52</i>	Active Min	Minimum time a flow was active before becoming idle
<i>f53</i>	Active Mean	Mean time a flow was active before becoming idle
<i>f54</i>	Active Max	Maximum time a flow was active before becoming idle
<i>f55</i>	Active Std	Standard deviation time a flow was active before becoming idle
<i>f56</i>	Idle Min	Minimum time a flow was idle before becoming active
<i>f57</i>	Idle Mean	Mean time a flow was idle before becoming active
<i>f58</i>	Idle Max	Maximum time a flow was idle before becoming active
<i>f59</i>	Idle Std	Standard deviation time a flow was idle before becoming active
<i>f60</i>	Flow duration	Duration of the flow in Microsecond
<i>f61</i>	Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
<i>f62</i>	Act_data_pkt_forward	Count of packets with at least 1 byte of TCP data payload in the forward direction
<i>f63</i>	Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
<i>f64</i>	Protocol	Protocol type

20s). Table 8 presents the average precision of random forest (RF), ForestPA, CSForest, and optimized forest (with 100 trees) by varying the time-window size. It is possible to see that the ForestPA algorithm had the best results, followed by optimized forest, RF, and CSForest. It can be noticed that the precision in the case of RF and optimized forest changes slightly by varying the time window. This is not the case for the two other algorithms, where a clear difference can be observed between TW=5s and the other time-windows. Contrary to what we might initially suppose,

Table 8

Evaluation of time-window size classification accuracy

Time-Window	RF	OptimizedForest	ForestPA	CSForest	MRF	GA_RF
1 s	70.83	70.00	69.67	70.83	63.16	78.67
5 s	72.50	72.83	75.83	58.83	77.33	80.00
10 s	68.83	69.17	70.83	71.83	72.05	79.33
20 s	71.00	71.67	71.67	71.17	71.38	79.67

**Figure 6:** Comparison between GA_RF and state of the art algorithms

the results show that increasing the time-window does not necessarily improve the precision of detection. By increasing the TW from 1s to 5s the results improved, but beyond 5s the precision dropped slightly. This can be explained by the fact that the sessions between the C&C server and the thingbots are established for short periods of time to exchange small amounts of data (mainly commands). By increasing the TW, legitimate packets are considered in the features calculation, which may distort the network pattern of the botnet flow. As three out of the four algorithms (RF, OF, and FPA) give their best performance by setting the TW to 5s, we choose the 5s-time-window to test GA_RF.

Overall, the best detection precision obtained using ForestPA is (75.83%). We run GA_RF with 10 CV on the 5s-Time-Window dataset, the experimental results are illustrated in Figure 6. They show that GA_RF provides a significantly better performance compared to state-of-the-art algorithms. GA_RF yields the best precision (80%) and the lowest number of decision trees. Compared to the RF algorithm, GA_RF presents a 7.5% precision improvement while generating 88 decisions trees less than RF. The false positive rate can be explained by the similarity between the C&C and legitimate traffic. The good classification performance obtained using the proposed approach makes it possible to detect C&C traffic and allow for early identification of thingbots. We have shown that using pertinent network features, along with GA_RF, allows for an early, fast, and accurate detection of thingbots.

To compare our approach with state-of-the-art domain approaches and further evaluate its effectiveness against other types of botnets, we evaluate GA_RF on the dataset proposed by Bezerra et al. (2019). This dataset includes the network traffic and the host data of three different IoT devices and seven botnets such as: Hajime, Aidra, BashLite, Doflo, Tsunami, etc. For a fair comparison, we consider the same number and set of features (see Table 7) used by GA_RF. Figure 7 shows a comparison between the proposed approach GA_RF and IoTDS (Bezerra et al., 2019). As shown, GA_RF outperforms IoTDS in both metrics, a significant difference in terms of precision can be noticed. Also, this comparison demonstrated that our approach performs well not only against Mirai, but also against six other types

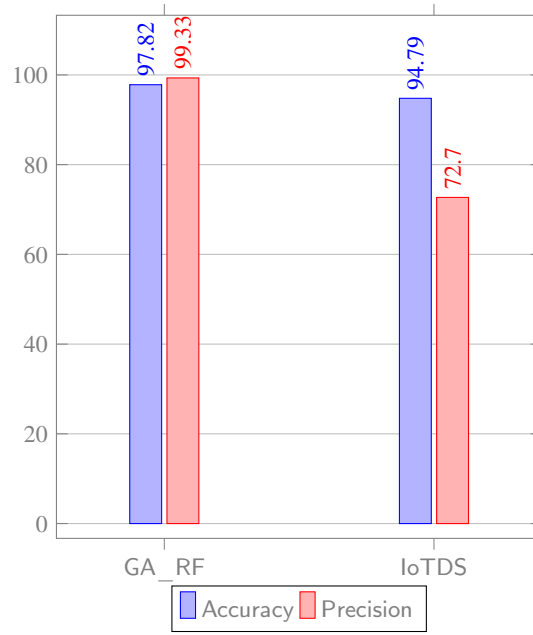


Figure 7: Comparison between GA_RF and IoTDS

of botnets. In addition to its superior performance, the proposed system has no impact on the IoI device. Since, it can run in the network gateway, unlike IoTDS which needs to run in the IoT device.

7. Conclusion

The popularity of RF is strongly related to its simplicity, strong prediction performance and interpretability. In practice a set of n trees are generated to construct the RF model. Random sampling and majority voting are used over the n trees to boost the accuracy, the variance and hence the generalization capacity of the overall RF model. This paper proposes GA_RF algorithm, which produces a forest model that optimizes the classification accuracy, trees number, and diversity. An experimental evaluation on several datasets from the UCI repository was performed. The obtained results showed that our approach outperforms existing approaches from the literature in terms of accuracy while using a smaller and more diverse set of trees.

This paper also has established that it is possible to find a combination of a small number of diversified and accurate decision trees that improves the accuracy and generalization of the random forest model as a whole. Besides, the number of trees in a random forest model might differ from one data set to another unlike the default number of trees of 100 that is usually used in practice.

Finally, an additional experimental evaluation of the proposed approach is performed on a real-world cybersecurity application for the detection of IoT botnets. It showed that using the proposed approach along with pertinent network features and optimal time-window allowed an early, fast, and accurate detection of botnet traffic in an IoT environment. An interesting future direction could be to study the application of proposed method in regression and time series forecasting problems. It could also be interesting to study the impact of the size of the individual trees on the overall performance of the random forest.

Acknowledgements

This research is a result from PRFU project C00L07UN230120200006 funded in Algeria by The Directorate-General for Scientific Research and Technological Development (DGRSDT).

References

- Adnan, M.N., Ip, R.H., Bewong, M., Islam, M.Z., 2021. Bdf: A new decision forest algorithm. *Information Sciences* 569, 687–705.
- Adnan, M.N., Islam, M.Z., 2016. Optimizing the number of trees in a decision forest to discover a subforest with high ensemble accuracy using a genetic algorithm. *Knowl. Based Syst.* 110, 86–97.
- Adnan, M.N., Islam, M.Z., 2017. Forest PA: Constructing a decision forest by penalizing attributes used in previous trees. *Expert Systems with Applications* 89, 389–403.
- Bader-El-Den, M., Gaber, M., 2012. Garf: towards self-optimised random forests, in: *International conference on neural information processing*, Springer. pp. 506–515.
- Bai, J., Li, Y., Li, J., Yang, X., Jiang, Y., Xia, S.T., 2022. Multinomial random forest. *Pattern Recognition* 122, 108331. URL: <https://www.sciencedirect.com/science/article/pii/S0031320321005112>, doi:<https://doi.org/10.1016/j.patcog.2021.108331>.
- Bezerra, V.H., da Costa, V.G.T., Barbon Junior, S., Miani, R.S., Zarpelão, B.B., 2019. Iotds: A one-class classification approach to detect botnets in internet of things devices. *Sensors* 19. URL: <https://www.mdpi.com/1424-8220/19/14/3188>.
- Biau, G., Scornet, E., 2016. A random forest guided tour. *Test* 25, 197–227.
- Breiman, L., 2001. Random forests, in: *Machine Learning*, pp. 5–32.
- Dhifli, W., Karabadj, N.E.I., Elati, M., 2020. Evolutionary mining of skyline clusters of attributed graph data. *Information Sciences* 509, 501–514.
- Dua, D., Graff, C., 2017. UCI machine learning repository. URL: <http://archive.ics.uci.edu/ml>.
- Durillo, J.J., Nebro, A.J., 2011. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software* 42, 760–771.
- Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 119–139.
- Ganaie, M., Tanveer, M., Suganthan, P., Snasel, V., 2022. Oblique and rotation double random forest. *Neural Networks* 153, 496–517. URL: <https://www.sciencedirect.com/science/article/pii/S0893608022002258>, doi:<https://doi.org/10.1016/j.neunet.2022.06.012>.
- Habibi Lashkari, A., Draper Gil, G., Mamun, M., Ghorbani, A., 2017. Characterization of tor traffic using time based features, pp. 253–262. doi:10.5220/0006105602530262.
- Hall, M.A., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The weka data mining software: an update. *SIGKDD Explor.* 11, 10–18.
- Ho, T.K., 1995. Random decision forests, in: *Proceedings of 3rd International Conference on Document Analysis and Recognition*, pp. 278–282 vol.1.
- Jalal, N., Mehmood, A., Choi, G.S., Ashraf, I., 2022. A novel improved random forest for text classification using feature ranking and optimal number of trees. *Journal of King Saud University-Computer and Information Sciences*.
- Karabadj, N.E.I., Khelf, I., Seridi, H., Aridhi, S., Remond, D., Dhifli, W., 2019. A data sampling and attribute selection strategy for improving decision tree construction. *Expert Systems with Applications* 129, 84–96.
- Karabadj, N.E.I., Seridi, H., Bousetouane, F., Dhifli, W., Aridhi, S., 2017. An evolutionary scheme for decision tree construction. *Knowledge-Based Systems* 119, 166–177.
- Labiod, Y., Amara Korba, A., Ghoualmi, N., 2022. Fog computing-based intrusion detection architecture to protect iot networks. *Wireless Personal Communications*, 1–29.
- McDermott, C.D., Majdani, F., Petrovski, A.V., 2018. Botnet detection in the internet of things using deep learning approaches, in: *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. doi:10.1109/IJCNN.2018.8489489.
- Mohapatra, N., Shreya, K., Chinmay, A., 2020. Optimization of the random forest algorithm, in: *Advances in data science and management*. Springer, pp. 201–208.
- Nguyen, G.L., Dumba, B., Ngo, Q.D., Le, H.V., Nguyen, T.N., 2022. A collaborative approach to early detection of iot botnet. *Computers Electrical Engineering* 97, 107525. URL: <https://www.sciencedirect.com/science/article/pii/S0045790621004717>, doi:<https://doi.org/10.1016/j.compeleceng.2021.107525>.
- Popoola, S.I., Ande, R., Adebisi, B., Gui, G., Hammoudeh, M., Jogunola, O., 2022. Federated deep learning for zero-day botnet attack detection in iot-edge devices. *IEEE Internet of Things Journal* 9, 3930–3944. doi:10.1109/JIOT.2021.3100755.
- Resende, P.A.A., Drummond, A.C., 2018. A survey of random forest based methods for intrusion detection systems. *ACM Comput. Surv.* 51. URL: <https://doi.org/10.1145/3178582>, doi:10.1145/3178582.
- Rokach, L., 2016. Decision forest: Twenty years of research. *Information Fusion* 27, 111–125.
- Siers, M.J., Islam, M.Z., 2015. Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem. *Information Systems* 51, 62–71.
- Syed Abul, B., Perry, S., 2022. Forecasting bitcoin price direction with random forests: How important are interest rates, inflation, and market volatility? .
- Walker, A.M., Cliff, A., Romero, J., Shah, M., Jones, P., Gazolla, J.G.F.M., Jacobson, D., Kainer, D., 2022. Evaluating the performance of random forest and iterative random forest based methods when applied to gene expression data. *Computational and Structural Biotechnology Journal*.
- Zheng, J., Liu, Y., Ge, Z., 2022. Dynamic ensemble selection based improved random forests for fault classification in industrial processes. *IFAC Journal of Systems and Control* 20, 100189.