Al-Rafidain Engineering Journal (AREJ) Vol.31, No.3, September 2026, pp. 84-103

Machine Learning Multi-Class Intrusion DetectionSystem Using Embedded Feature Selection and Multi-Level Hierarchy Classification

Tarfa Hamed * tyaseen@uomosul.edu.iq
Stefan C. Kremer** skremer@uoguelph.ca
*Department of Networks, College of Computer Science and Mathematics,University of Mosul, Mosul, Iraq
**School of Computer Science, College of Computational, Mathematical, and Physical Sciences, University of Guelph, Guelph, Canada

Received: March 16th, 2026 Received in revised form: June 5th, 2026 Accepted: June22th, 2026

ABSTRACT

In recent years, network security has attracted significant research attention due to the growing frequency of attacks on computer networks. Among existing solutions, "Network Intrusion Detection Systems (NIDS)" play a central role. While binary NIDS effectively distinguish between normal and malicious traffic, they cannot accurately classify specific attack types. Multi-class intrusion detection systems address this limitation by enabling tailored responses for each attack category. This paper presents the development of a "multi-class Intrusion Detection System (IDS)" that integrates "Recursive Feature Addition (RFA)", an embedded feature selection technique, with multi-level binary classifiers. The proposed system was evaluated using five machine learning algorithms: "Support Vector Machines (SVM)", "Random Forests", "J48", "Random Tree", and "J48Graft". A hierarchical binary tree structure was employed to implement the multi-class detection, with performance assessed under two dataset distribution scenarios: "balanced" and "imbalanced". To construct the overall "multi-class confusion matrix", a novel estimation method that combines intermediate matrices from each binary sub-tree has been introduced.The ISCX 2012 intrusion detection benchmark dataset was used for evaluation, with results measured across five metrics: "Accuracy", "F1-score", "Detection Rate", "False Alarm Rate (FAR)", and "Precision". Experimental findings show that "Random Tree" consistently achieved the best performance across both scenarios. For "balanced" datasets, it excelled in detecting DoS attacks, while for "imbalanced" datasets, it demonstrated superior efficiency in identifying Internal attacks. In addition, the proposed approach has demonstrated superiority over the baseline in terms of time required for (feature selection + classification hierarchy) and key metrics. These results highlight the effectiveness of the proposed multi-class IDS framework for enhancing intrusion detection in complex network environments.

Keywords:*Feature Selection, Intrusion Detection, J48, Multi-class IDS, Network Security, Random Tree.*

This is an open access article under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).
https://jamh.uomosul.edu.iq/index.php/rengj
Email:alrafidain_engjournal3@uomosul.edu.iq

==========

1. INTRODUCTION

Due to the high reliance on the Internet and its applications, it has become a target for many types of threats. These threats have caused significant damage to the Internet and the information it holds. Consequently, many organizations have recognized the growing need for an advanced tool to protect networks against these attacks. Therefore, "cyber security" is one of the major areas attracting the attention of all concerned parties ("governments", "corporations", "organizations", "internet service providers"and "individuals"). "Network security" is a collection of technologies and procedures incorporated together to protect computer networks from attacks and unauthorized access. Network security tools include "firewalls", "antivirus software" and "intrusion detection systems (IDSs)" [1].

An "IDS" is a network security tool that has become a significant research topic, especially given the sophisticated methods contemporary attacks use to avoid detection. In general, an "IDS" is tasked with identifying any activity that attempts to breach the security goals of a computer system: "Confidentiality", "Integrity" and "Availability". Now, some authors are suggesting adding the "IDS" alongside antivirus software as a complement to provide better security for an organization's infrastructure [2].

Authors have been using "machine learning" for "network security" to build "IDSs", as this can help minimize the threat posed by attackers [3]. Intrusion detection is seen as an intriguing classification challenge, primarily because of the rising frequency of attacks on computer networks. [4]. Therefore, many machine learning models have been utilized to detect intrusions. Those models vary between "neural networks" [5], "support vector machines" [6], "decision trees" [7], "extreme learning" [8], [9], "genetic algorithms" [10], and "fuzzy logic" [11]. In addition, since the attacks are of different categories, intrusion detection can be considered a "multi-class classification" problem. Therefore, another strategy is to distinguish the specific type of attack rather than merely triggering an alarm without disclosing its type. The benefit of using the latter method is that it simplifies administrators' task of choosing the appropriate response to that intrusion. The ultimate objective in both cases ("binary classification" or "multiclass classification") is to construct a classifier capable of generalization, which means accomplishing the classification task effectively on novel ("unfamiliar")examples[4].

Numerous machine learning methods have been applied to "multi-class intrusion detection" to enhance classifier effectiveness. One machine learning technique used to improve the performance of intrusion detection systems is "feature selection," as demonstrated by [12]. In this context, the objective of "feature selection" is to identify the most suitable subset of features, ensuring no loss of information while attaining the highest classification performance. In terms of their operational location, "Intrusion Detection Systems (IDSs)" can be categorized into two types: "Host-based IDS (HIDS)" and "Network-based IDS (NIDS)". In addition, IDSs can be divided into model types, including "Anomaly-based IDS" and "signature-based IDS" (also referred to as "misuse-based IDS").

In recent times, it has become evident that sustained efforts in research, design, and implementation in the domain of "NIDS" are imperative to defend the "Internet" against ever-changing threats. These attacks have grown progressively more sophisticated, combining a high degree of destructiveness with a covert approach, making them difficult to identify. Consequently, the creation of an efficient "NIDS" has become markedly imperative to shield the Internet from these harmful attacks.

The development of "IDSs" in the fields of computer and network security has evolved in recognition of the infeasibility of ensuring absolute immunity for a system against security vulnerabilities. In this context, the term "system" or "target system" refers to the information system under surveillance by the "IDS", which may encompass a range of entities, including "workstations", "network terminals", "servers", "web servers", or any other computing system [13].

In this paper, a machine learning classifier has been incorporated along with an "embedded feature selection" technique known as "Recursive Feature Addition (RFA)", as detailed in [14] and [15], to construct a hierarchical "multi-class IDS".

The rest of the paper is organized as follows: Section 2 presents the motivation for this work. Section 3 lists the main contributions of this work. In Section 4, the previous work in this area and the use of feature selection in intrusion detection are explored, and the challenges faced by IDSs are discussed. Section 5 provides the reader with a sufficient understanding of each classifierused in the experiments. In Section 6, the "ISCX 2012 benchmark dataset" is presented. The binary tree structure used to implement the "multiclass IDS" is explained in Section 7, along with dataset preparation. Moreover,details of our feature selection method are providedin Section 8. The experimental design is discussed in Section 9. The experimental results for each dataset distribution are presented in Section 10. Lastly, the conclusions and future work are explained in Section 11.

2. MOTIVATION

The motivation for this research is to implement a "multi-class IDS" with lower computational complexity. A "multi-level binary classifier" is proposed for attack classification using the "ISCX2012 dataset". At each level, a binary classifier distinguishes between two attack classes. This will also help in minimizing the complexity of "multi-class feature selection".

3. CONTRIBUTIONS

This paper involves several contributions which can be listed as follows:

1. A "multi-class intrusion detection system" is implemented by building a "binary tree structure" that uses five binary classifiers to classify incoming traffic into five classes (four attack classes and one for normal traffic). The "binary tree" consists of three levels of classification. At the first level, the system distinguishes between normal and attack connections. If the connection is an attack, it is sent to the second level to determine whether it is from the "DoS" or "non-DoS" attack groups. If the connection originated from the "Denial of Service (DoS)" attacks group, it is sent to the

third level to determine whether it is an "HTTP DoS" or a "Distributed Denial of Service (DDoS)" attack. At the same level, if the connection originated from the "Non-DoS" attack group, it proceeds to the third level to determine whether it is an "internal" attack or a "brute-force" attack.

2. Since a binary tree hierarchy was used to classify incoming traffic, it resulted in multiple binary "confusion matrices". Therefore, an estimation procedure for constructing the final confusion matrix from the "multiple binary confusion matrices" that result from each "binary sub-tree" has been proposed.
3. Two different scenarios have been built based on dataset distribution to increase the problem's complexity ("balanced" and "imbalanced"). The "balanced" dataset has equal numbers of training and test examples, while the "imbalanced" dataset has a test set that is 9 times the size of the training set.
4. The "feature selection" has been integrated into the "IDS" to enhance its performance in identifying incoming attacks.
5. Five different "machine learning classifiers" have been employed to detect intrusions, and the performance of each classifier has been measured using five evaluation metrics.
6. The work also involved investigating which classifier performed better on each dataset and diagnosing which factors were degrading the classifiers' performance.
7. The proposed model showed superior performance compared to six baseline papers across time required for (feature selection + classification hierarchy) and key metrics.

4. PREVIOUSWORKS

The authors in [16] used a hybrid feature selection method using "Correlation based Feature Selection (CFS)" and "Information Gain" to reduce the number of attributes ("features") while simultaneously enhancing performance of a 5-class intrusion detection problem. The other authors in [17] proposed an intrusion detection model based on "chi-square feature selection" and "multi-class Support Vector Machines (SVM)". The authors employed a parameter-tuning technique for two important SVM parameters: gamma for the "Radial Basis Function (RBF)" kernel and the overfitting constant C. The results of the proposed model showed a better detection rate and lower false alarm rate. SVMs were also used for "multi-class classification" for intrusion detection as in [18]. The authors used multiple SVMs of type "one-versus-all" to detect a specific attack across three datasets ("KDD corrected", "NSL-KDD", and "Gure-KDD"). Those datasets include multiple intrusion classes in addition to normal connections. The "corrected KDD" has 38 classes, "NSL-KDD" has 23 classes, and "Gure-KDD" has 28 classes. Their proposed approach efficiently detected the rarest attack types, "User to Root (U2R)" and "Remote to Local (R2L)."

Since this paper concerns feature selection in intrusion detection, the next subsection outlines previous efforts in this area. In addition, the challenges faced by most IDSs are discussed. The authors in [19] used an SVM classifier to build an accurate IDS using augmented features. Their proposed model involved reconstructing the original features from the "NSL-KDD" dataset to provide higher-quality, more concise training data for the SVM algorithm. The SVM is then trained on the newly constructed features, and then the IDS model is built. Their proposed model demonstrated improved detection performance and reduced training time.

In another paper, the authors employed an ensemble-based Intrusion Detection model for the "Internet of Things IoT". The authors utilized "logistic regression", "naive Bayes", and "decision trees" with a voting classifier. The effectiveness of their proposed model was measured using the "CICIDS2017 dataset". Their obtained accuracy showed remarkable improvement as compared to some other models on both "binary" and "multi-class classification" problems [20].

The authors have also focused on addressing class imbalance in IDS datasets. In [21], the authors combined oversampling and undersampling within an ensemble classification framework to address class imbalance. Their experiment was conducted on two Car Hacking datasets: Attack and Defense Challenge 2020 (CHADC2020) and IoTID20. Their approach demonstrated strong performance in optimizing precision, recall, and F1-score, critical metrics for evaluating IDS effectiveness.

One recent study in "multi-class IDS" is [22], where the authors proposed an approach called the "Aegean Wi-fi Intrusion Dataset (AWID)". The authors used a dataset with four classes ("normal", "injection", "flooding", and "impersonating"). Their work showed that using only 10% of the available features achieved the same classification accuracy.

The authors have also employed "deep learning" in addressing the "multiclass intrusion detection" problem. An example of this kind of study is [23], in which the authors used a "neural network" consisting of multiple stacked fully connected layers to handle a flow-based "anomaly detection IDS". The authors used the "CICIDS2017" dataset for their experiments. The results showed that their proposed model achieved

notable performance in "multi-class classification" in terms of accuracy, detection rate, and False Alarm Rate.

The "multi-class classification" is not exclusive to intrusion detection problems. It can be used with other problems such as "text classification". In [24], the authors concluded that removing data imbalance and sparsity can lead to significant improvements in classifier efficiency.

4.1. Utilizing feature selection in intrusion detection

Feature selection has been used in intrusion detection in both scenarios ("binary classification" and "multi-class classification") to improve the performance of the "NIDS" by reducing data dimensionality during preprocessing [9]. Therefore, many studies have been conducted on applying feature selection to improve the performance of "IDS" in terms of accuracy, detection rate, and other metrics, as in [15].

Previous studies have employed diverse datasets specific to "IDS" testing. Nevertheless, in this paper, our focus is on utilizing the "ISCX 2012 dataset" (which will be further elucidated in Section 6).

In [25], the authors used the "intraclass" and "interclass" correlation coefficients to obtain a class-specific subset of features. The "interclass" and "intraclass" correlation coefficients were used to measure the validity and reliability of the features, respectively. The authors tested their model on the "ISCX 2012 dataset". They observed that combining "interclass" and "intraclass" correlation coefficients increased the detection rate and decreased both execution time and the false alarm rate.

In other studies such as [26], the authors opted to build their intrusion detection systembased on the normal traffic to detect unseen intrusions using the "ISCX 2012" dataset. The authors utilized a "one-class SVM classifier" to train a model for anomaly detection in "HTTP traffic" by learning its regular attributes. Their method included obtaining significant attributes from both "normal" and "abnormal HTTP traffic". The system underwent training using specifically "normal" traffic examples to establish a baseline for "normal behavior". Whenever a deviation from the normal traffic model was detected, the system generated an alert. The authors achieved 80% accuracy and a false alarm rate of 8.6% in detecting attacks on port 80h.

The authors in [27] used a wrapper-based feature selection method on the UNSW-NB dataset, reducing the feature set from 43 to 19. In addition, they used a decision tree classifier on the reduced dataset. To improve detection rates for DoS, Exploit, and Fuzzer attacks, the authors proposed a hierarchical multi-class classifier. They used a random forest classifier as the base classifier for their experiment.

Cloud computing environments also face various attacks that target the cloud's bandwidth, services, and resources, making the cloud unavailable to both cloud users and providers. The authors in [28] used an "ensemble-based feature selection" method that combined the outputs of multiple filter methods to obtain the best feature subset for detecting "Distributed Denial of Service (DDoS)" attacks. They applied their approach on the "NSL-KDD dataset" using a "decision tree" classifier. Their findings showed that their proposed approach reduced the number of features from 41 to 13, achieving higher detection rates and classification accuracy than other techniques.

Another approach to using "machine learning" with "intrusion detection" was [29], which used two classifiers: "Multiple Criteria Linear Programming (MCLP)" and "Support Vector Machines". Their approach incorporated "Time-Varying Chaos Particle Swarm Optimization (TVPSO)" to perform both parameter tuning and feature selection simultaneously. They adopted a weighted objective function that balanced maximizing the detection rate and minimizing the false alarm rate. Their results showed accurate intrusion detection by selecting a more discriminative feature subset. They also found that the "MCLP classifier" achieved comparable performance to the SVM and produced better detection rates for "R2L" and "U2R" attacks.

The authors in [2] proposed a filter-based feature selection algorithm named "Flexible Mutual Information Feature Selection (FMIFS)". They incorporated their method with the SVM classifier to build an intrusion detection system called "Least Square Support Vector Machines based Intrusion Detection System (LSSVM-IDS)". Their proposed method was able to process linearly and non-linearly dependent features. The performance of the proposed method was evaluated on three datasets: "KDD Cup99", "NSL-KDD" and "Kyoto 2006+". Their results showed that their feature selection algorithm identified critical features, enabling the "IDS" to achieve better accuracy and lower computational cost than other methods.

4.2. Challenges to IDS

One of the major challenges in "network-based intrusion detection" is the vast volume of data that needs to be collected from the network. To address this, raw network traffic must

be condensed into “higher-level” events, such as “connection records,” before being used by a “machine learning” algorithm. This process involves abstracting each “high-level” event by defining a set of features that describe its characteristics [30]. The result of this process will be an “intrusion detection dataset”. Nonetheless, these datasets tend to lead certain “machine learning algorithms” to overfit, which can subsequently hinder the performance of the “IDS” in terms of detection accuracy and false alarm rates. Furthermore, attacks are usually categorized differently; hence, their treatment differs. Therefore, knowing the exact type of a threat (attack) is important for taking the appropriate response to that attack.

5. METHODS

In this section, a brief overview of each classification algorithm used in this work is presented.

(i) Random Forest

Random Forests were first introduced by [31]. They use a “supervised learning algorithm” constructed from multiple “decision trees” [32]. The method is based on the concept of an “ensemble” of decision trees. It starts by calculating a decision tree on randomly selected features and data instances. The data instances are split into “training” and “testing” sets. The training set will be used to train the trees, while the test set will be used to evaluate their performance [33]. The method combines the predictions of many models into a single prediction, which is typically called the plurality or “majority vote” in classification problems [34].

(ii) J48

J48 is another supervised classifier; in fact, the “Java-built” version of “C4.5”, which is an improved version of the “ID3” algorithm [35]. A “J48 tree” differs from an “ID3” tree in determining the best target feature based on the gain ratio. In addition, “J48” can handle numerical features by setting a threshold and splitting the data into partitions based on whether their feature values are above or below it. Another advantage of the “J48” tree is that it performs pruning of the “decision tree” after building the tree, thereby reducing the tree’s size and saving time and memory [36]. “J48” implements a pruning technique called sub-tree replacement, which involves replacing nodes in a “decision tree” along with their leaves to reduce the number of tests in the current path [37].

(iii) J48graft

J48Graft is a modified version ofthe “J48” tree. The tree uses splitting criteria to split the data based on information gain. Interestingly, this tree addresses the shortcomings of the “ID3” tree with a large “decision tree” [38]. The main objective of grafting is to reclassify areas of an example space where no training data are available, or when data are misclassified, thereby decreasing prediction error [36].

(iv) Random Tree

Random Tree is another powerful “supervised classifier” from the family of “ensemble learning” algorithms. The classifier uses the “bagging technique” to generate a random subset of data to build a “decision tree”. The basic concept of the “Random Tree” is to split at each node and select the best split among all variables [39][40].

(v) Support Vector Machines

Support Vector Machines areamong the most powerful machine learning models, having proven their efficiency in classification and regression problems. An “SVM” is based on finding a separating line between two labeled classes of data (in the case of a “binary classification” problem) [41]. However, the classifier can be used for “multi-class” classification by generating hyperplanes separating the labeled classes using the “one against all” approach [42]. The “SVM” learning system employs a technique known asthe “kernel trick” to map the input data into a higher-dimensional space [41].

6. ISCX 2012 DATASET

The “ISCX dataset”, created in 2012, was generated by the “Information Security Centre of Excellence (ISCX)” located at the “University of New Brunswick” [43]. This datasetwas chosen for this paper because it contains realistic network traffic data, is labeled, comprises diverse intrusion scenarios, and is a popular benchmark dataset. The dataset contains real traces assembled to generate profiles for agents that generate real traffic for “HTTP”, “SMTP”, “SSH”, “IMAP”, “POP3”, and “FTP”. The generated dataset contains various features, including the entire packet payloads, as well as other relevant features such as the total number of bytes sent or received.

The complete dataset was gathered over a span of seven days, from “Friday June 11th at 00:01:06” to “Friday June 18th at 00:01:06”. It consisted of over 1 million network-trace packets and encompassed 20 distinct features. Each data example was categorized as either “normal” or “attack,” representing the two classes in the dataset

[44]. Due to its realistic traffic, labeled connections, and multiple attack scenarios, the "ISCX 2012 dataset" has become a benchmark for intrusion detection research and has been recognized by the security community [43]. The primary objective of designing the dataset was to address the technical limitations observed in other intrusion detection datasets. It was specifically intended to provide a collection of network traces that encompassed both current legitimate network behaviors and intrusive patterns [45]. More details about the "ISCX 2012 dataset" are available in [43].

7. MULTI-CLASS CLASSIFICATION USING BINARY TREE STRUCTURE

To utilize the "binary Support Vector Machine (SVM)"- based feature selection method (RFA), a binary tree structure is devised to facilitate "multi-class classification". The suggested binary tree configuration is depicted in Figure 1. The advantage of cascading multiple "binary SVMs" to implement "multi-class SVM" is that it avoids the complexity of implementing the traditional "multi-class classifier" in the feature selection process. Consequently, this will give us an early indication of the class causing misclassification and degrading performance.

As explained in the Introduction, multiple binary classifiers are operated to perform multi-class classification. The classification process starts at the top of the initial sub-tree, as depicted in Figure 1, to differentiate incoming traffic into two general categories: normal and attack. Within the attack category (the right branch), there are two sub-categories: "DoS attacks" and "non-DoS attacks". The "DoS attacks" branch further branches out into two sub-classes: "HTTP DoS" and "DDoS attacks", while the "non-DoS attacks" encompass "internal attacks" and "brute-force attacks". As a result, a total of four subtrees were employed in constructing the binary tree configuration: "normal" vs. "attack", "DoS" vs. "non-DoS", "HTTP DoS" vs. "DDoS", and "internal" vs. "brute-force attacks".

8. THE RECURSIVE FEATURE ADDITION (RFA) FEATURE SELECTION

The initial RFA method [14], is a feature selection technique integrated within the model. This method follows a "greedy forward approach" and leverages the "SVM" classifier to identify and select the most relevant features. The method starts with an empty set of features that will be updated during the selection process. The RFA recursively adds one feature at a time to that set according to the calculated ranking coefficients of the remaining features.

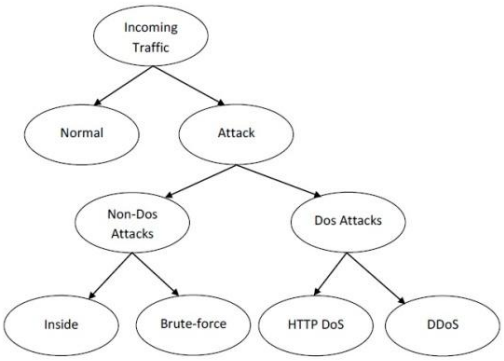

Fig. 1 Binary tree structure for multi-class classification used in this paper

The RFA is designed for two-class classification problems. In this paper, a "multi-class classification" problem is addressed. As stated earlier, during the learning process, the "RFA" employs the "SVM" classifier to conduct "feature selection". The "SVM" computes the weights wi of the decision function D(x) only for the support vectors (a small subset of the training instances).

The cost function of the "SVMs" that needs to be minimized is [46]:

$$J = \sum_{k=1}^{p} \alpha_k \left(1 - b_y k\right) - \frac{1}{2} \alpha . H . \alpha \quad (1)$$

Subject to

$$\alpha_k \geq 0, k = 1,2,3, \ldots p$$

Where H is the matrix that can be calculated as:

$$H = y_s y_t K(x_s, x_t) \quad (2)$$

and x_s and x_t are training examples. In Equation 2, K represents a kernel function used to assess the similarity between examples xs and xt, s = 1..N, t = 1..N, where N is the number of features, while y denotes a vector of class labels. The algorithm incorporates the "RBF" kernel function, which can be computed using the following equation:

$$K(x_s, x_t) = \exp\left(-\gamma \|x_s - x_t\|^2\right) \quad (3)$$

where γ represents a constant and commonly selected as $\frac{1}{number\ of\ features}$.

The "ranking coefficient" involves calculating the change in the cost function after each feature i addition. Another variable in the algorithm is the H matrix, which must be recalculated and is denoted $H(+i)$. The subscript $(+i)$ signifies the variable after adding feature i. This process entails computing $K(x_h(+i), x_k(+i))$. The final ranking coefficient, DT, is determined through the following equation:

$$DT = \left(\frac{1}{2}\right) \alpha^T H \alpha - \left(\frac{1}{2}\right) \alpha^T H(+i) \alpha \quad (4)$$

At the beginning of the algorithm, the ranking coefficient *DT* is set to the second term, since no feature has been selected yet. The algorithm proceeds by selecting the feature that shows the greatest difference, DT(i), and adding it to the list of selected features. The process is iteratively executed to perform "Recursive Feature Addition (RFA)". A comprehensive depiction of the "RFA" algorithm is discussed in [14]. Figure 2 shows the detailed flowchart of "RFA" using "SVMs".

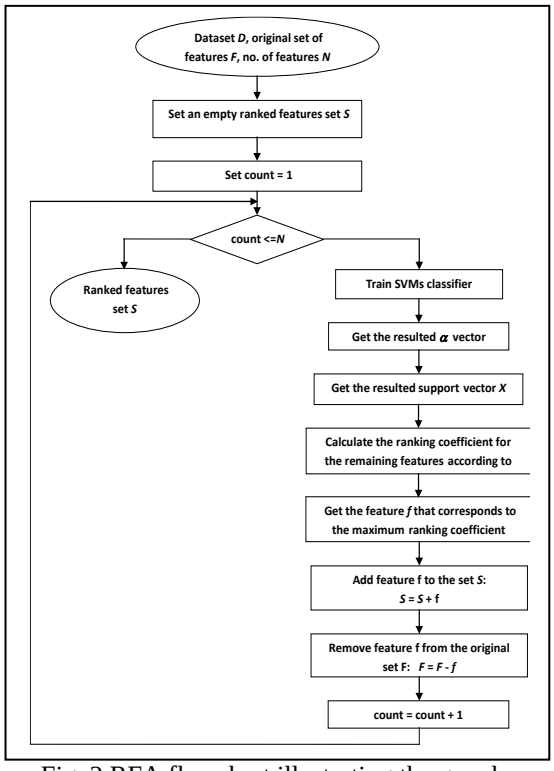

Fig. 2 RFA flowchart illustrating the greedy fashion in selecting features using SVMs

9. EXPERIMENTAL DESIGN

In this section, the dataset preparation and the evaluation metrics used to evaluate the system are discussed.

9.1. Dataset Preparation

To prepare the dataset for experimentation, two distinct approaches for dividing the data into training and test sets are used. The first approach involved assigning 10% of the data to training and 90% to testing. The second approach centered on generating equitable datasets to facilitate both training and testing. By employing these two approaches, it is aimed to analyze the performance of feature selection under different splitting conditions and determine the most effective approach. Table 1 shows the dataset distribution when employing the equitable approach. Concerning the attributes, an identical set of attributesis employed, including "*ApplicationName*", "*TotalSourceBytes*", "*TotalDestinationBytes*", "*TotalDestinationPackets*", "*DestinationPort*", "*TotalSourcePackets*", "*SourcePort*" "*Direction*", "*SourceTCPFlagsDescription*", "*ProtocolName*", "*DestinationTCPFlagsDescription*", and "*Class*". However, certain attributes, such as "TimeStart", "TimeEnd", and "Base64", were excluded from this experiment.

Table 1: The "Balanced" Approach of Datasets' Distribution

Dataset	Training	Testing
Normal vs. attacks	19107	19111
DoS vs. non-DoS attacks	10154	9501
HTTP DoS vs. DDoS attacks	5066	4409
Internal vs. brute-force attacks	5088	5092

Table 2 demonstrates the alternative approach for dataset distribution, which involves an "imbalanced" distribution with 10% of the data allocated for training and 90% for testing purposes.

Table 2: Datasets' distribution for the "imbalanced" approach

Dataset	Training	Testing
Normal vs. attacks	1910	17199
DoS vs. non-DoS attacks	1015	9501
HTTP DoS vs. DDoS attacks	4114	37035
Internal vs. brute-force attacks	2555	23006

We used two different approaches to the dataset's distribution to study the behaviour of feature selection under each distribution and observe which one performs best.

9.2. Evaluation metrics

In our experiments, the objective was to evaluate the influence of implementing feature selection on a particular (dataset) by calculating various metrics commonly used in "feature selection" and "intrusion detection" systems. Following the feature selection process, we computed the following metrics to evaluate the outcomes:

1. Accuracy

The accuracy of a classifier is determined by employing the following formula:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

Where *TP*, *TN*, *FP*, and *FN* are "True Positive", "True Negative", "False Positive", and "False Negative" values, respectively. The classifier's accuracy is the percentage of the

correctly classified examples to the total number of examples.

2. F1 − score

The *F1-score*, which represents the "methodical mean" of "precision" and "recall", can be computed using the following equation:

$$F1 - score = \frac{2*TP}{(2*TP)+FP+FN} \quad (6)$$

3. Detection rate or Recall

The "detection rate (*DR*)" or "*Recall*" of any "IDS" denotes the proportion of accurately identified attacks to the overall number of attacks and can be measured using the following equation:

$$Detection\ Rate(DR) = \frac{TP}{TP+FN} \quad (7)$$

4. Precision

The precision indicates the ratio of accurately categorized attacks *TP* to the overall number of classified instances as attacks (including both *TP* and *FP*) according to the following equation:

$$Precision = \frac{TP}{TP+FP} \quad (8)$$

5. False Alarm Rate (FAR)

The "False Alarm Rate (*FAR*)" signifies the ratio of normal instances erroneously identified as attacks to the total number of normal instances according to the following equation:

$$FAR = \frac{FP}{FP+TN} \quad (9)$$

10. EXPERIMENTAL RESULTS

In this section, we present the experimental results for three categories: the performance metrics for the "balanced" datasets tree, the performance metrics for the "imbalanced" datasets tree, and the final confusion matrix for each dataset.

10.1. The results gained from implementing feature selection on the "balanced" Datasets.

In this part, we present the outcomes obtained by using the "RFA feature selection" method on the "ISCX datasets". After applying feature selection to the "ISCX 2012 dataset", we evaluated performance using five metrics. The performance metrics are shown for all five algorithms: "SVM", "Random Forests", "J48", "Random Tree" and "J48 Graft". We are going to show the results of each algorithm on the entire tree as follows:

i. SVMs

Table 3 shows the performance metrics achieved after employing feature selection on the balanced "ISCX datasets" using "SVMs". Each column in Table 3 represents the performance metrics of one sub-tree in the overall tree structure.

From Table 3, we can see that the system classified normal vs. attack connections (the second column) with high accuracy, F1−score, detection rate, and precision, and a low "False Alarm Rate" (only0.4%). With the "DoS" attacks sub-tree (the third column), the systemachieved very high scores across all metrics and a very low "FalseAlarm Rate". With the "DoS" vs. "non-DoS" attacks sub-tree (the fourthcolumn), the system was not able to achieve the same as with the previous twosub-trees, as the accuracy was only 72%, *F1-score*was 80%, detectionrate was 57%, "False Alarm Rate" was 0.08%, and precision was 99%. Inthe last sub-tree, "non-Dos" attacks (which involve "internal" and "brute-force" attacks), the system achieved perfect results across all metrics, as shown in the last column of Table 3.

Table 3: Results obtained from implementing feature selection on the balanced dataset usingSVMs

Metric	Normal vs. attack	DoS Attacks	DoS vs. Non-Dos	Non-DoS attacks
Accuracy	99%	99%	72%	100%
F1-score	99%	99%	80%	100%
D. rate	99%	100%	57%	100%
Precision	99%	99%	99%	100%
FAR	0.4%	0.04%	0.08%	0%

ii. Random Forest

Table 4 illustrates the performance metrics for the "balanced ISCX" dataset tree using the "Random Forest" algorithm after feature selection. Each column in Table 4 represents the performance metrics of one sub-tree in the overall tree structure.

Again, the system has achieved almost a perfect score across all sub-trees, except for the "DoS" vs. "Non-DoS" sub-tree. We observe that the system achieved 83% accuracy and F1−score, and 92% detection rate. The "FAR" was 25%, which is quite high compared to the other values in the same row. The recorded precision was only 76% for the "DoS" vs. "Non-DoS" sub-tree.

Table 4: Results obtained from implementing feature selection on the balanced dataset usingRandom Forest

Metric	Normal vs. attack	DoS Attacks	DoS vs. Non-Dos	Non-DoS attacks
Accuracy	99%	99%	83%	100%
F1-score	99%	99%	83%	100%
D. rate	99%	99%	92%	100%
Precision	99%	99%	76%	100%
FAR	0.1%	0.08%	25 %	0%

iii. J48

In Table 5, the performance metrics after applying feature selection to the "balanced ISCX" datasets using the "J48" tree are shown. Each column in Table 5 represents the performance metrics of one sub-tree in the overall tree structure.

Similarly to the previous tables, Table 5 shows comparable results for accuracy, F1−score, and detection rate across all sub-trees, except for "DoS" vs. "Non-DoS".

Table 5: Results obtained from implementing feature selection on the balanced dataset using J48

Metric	Normal vs. attack	DoS Attacks	DoS vs. Non-Dos	Non-DoS attacks
Accuracy	99%	99%	73%	100%
F1-score	99%	99%	77%	100%
D. rate	99%	99%	98%	100%
Precision	99%	99%	64%	100%
FAR	0.1%	0.08%	47 %	0%

However, the FAR metric shows poor performance (47%), and the precision is only 64% for the "DoS" vs. "Non-DoS" sub-tree.

iv. Random Tree

Table 6 presents the performance metrics obtained by applying feature selection to the "balanced ISCX" datasets using the "Random Tree" classifier. Each column in Table 6 represents the performance metrics of one sub-tree in the overall tree structure.

Table 6: Results obtained from implementing feature selection on the balanced dataset using Random Tree

Metric	Normal vs. attack	DoS Attacks	DoS vs. Non-Dos	Non-DoS attacks
Accuracy	99%	99%	90%	100%
F1-score	99%	99%	89%	100%
D. rate	99%	99%	80%	100%
Precision	99%	99%	99%	100%
FAR	0.2%	0.04%	0.1%	0%

Interestingly, the "Random Tree" classifier has achieved better performance than "SVMs", "Random Forest", and "J48" and that can be seen by comparing the fourth column ("DoS" vs. "Non-DoS") in all the tables, as the other columns show comparable performance in all the previous tables.

v. J48Graft

In Table 7, the performance metrics after applying feature selection to the "balanced ISCX" datasets using the "J48Graft" tree are shown. Each column in Table 7 represents the performance metrics of one sub-tree in the overall tree structure.

Again, "J48graft" shows similar performance to the previous algorithms (except the "Random Tree") as shown in Table 7. Same as with "SVMs", "Random Forest", and "J48", the recorded results of "J48graft" show poor performance with the "DoS" vs. "Non-DoS" sub-tree, as the accuracy was 77%, the *F1-score* was 73%, the FAR was 47%, and the precision was 64%.

Table 7: Results obtained from implementing feature selection on the balanced dataset using J48Graft

Metric	Normal vs. attack	DoS Attacks	DoS vs. Non-Dos	Non-DoS attacks
Accuracy	99%	99%	77%	100%
F1-score	99%	99%	73%	100%
D. rate	99%	99%	98%	100%
Precision	99%	99%	64%	100%
FAR	0.1%	0.1%	47%	0%

10.2. The results obtained from implementing feature selection on the imbalanced datasets

Same as with the "balanced" datasets, we measured the same metrics on the "imbalanced datasets". We are presenting the results for each sub-tree in a single column in the table. We are going to show the results of each algorithm on the entire tree as follows:

i. SVMs

Table 8 shows the performance metrics after applying feature selection to the "imbalanced ISCX 2012" datasets with "SVMs".

It can be seen in Table 8 that the system achieved 99% across four metrics and 0.7% "False Alarm Rate" for normal vs. attack connections (the second column). With the "DoS" attacks subtree (the third column), the system achieved 99% across all metrics and only 0.01% "False Alarm Rate". However, the situation was different with the "DoS" vs. "non-DoS" attacks sub-tree (the fourth column), as the accuracy was 73%, the F1-Score was 77%, the detection rate was 98%, the precision was 64%, and the "False Alarm Rate" was 47%. For the last sub-tree ("non-DoS" attacks), the system achieved a perfect score across all metrics, as shown in the last column of Table 8.

Table 8: Results obtained from implementing feature selection on the imbalanced dataset using SVMs

Metric	Normal vs. attack	DoS Attacks	DoS vs. Non-Dos	Non-DoS attacks
Accuracy	99%	99%	73%	100%
F1-score	99%	99%	77%	100%
D. rate	99%	99%	98%	100%
Precision	99%	99%	64%	100%
FAR	0.7%	0.01%	47%	0%

ii. Random Forest

Table 9 shows the performance metrics after applying feature selection to the "imbalanced ISCX 2012" datasets using "Random Forest". Each column in Table 9 represents the performance metrics of one sub-tree in the overall tree structure.

As shown in Table 9, all the sub-trees have achieved perfect or near-perfect scores.

Table 9: Results obtained from implementing feature selection on the imbalanced dataset using Random Forest

Metric	Normal vs. attack	DoS Attacks	DoS vs. Non-Dos	Non-DoS attacks
Accuracy	99%	99%	81%	100%
F1-score	99%	99%	81%	100%
D. rate	99%	99%	90%	100%
Precision	99%	99%	74%	100%
FAR	0.5%	0.02%	27%	0%

However, the situation is different with "DoS" vs. "non-DoS" attacks (the fourth column), where the obtained accuracy and *F1-score* were both 81%, the detection rate was 90%, "FAR" was 27%, and the precision was only 74%. Despite these results, overall, "Random Forest" has achieved better results than "SVMs".

iii. J48

Table 10 shows the performance metrics after applying feature selection to the "imbalanced ISCX 2012 datasets" with "J48".

Table 10: Results obtained from implementing feature selection on the imbalanced dataset using J48

Metric	Normal vs. attack	DoS Attacks	DoS vs. Non-Dos	Non-DoS attacks
Accuracy	98%	99%	73%	100%
F1-score	98%	99%	77%	100%
D. rate	98%	99%	98%	100%
Precision	99%	99%	64%	100%
FAR	0.9%	0.02%	47%	0%

Each column in Table 10 represents the performance metrics of one sub-tree in theoverall tree structure. A quick look at Table 10 shows that "J48" is very close to "SVMs," especially when comparing the fourth column ("DoS" vs. "non-DoS") of the above table with the same column in Table 8.

iv. Random Tree

In Table 11, the performance metrics after applying feature selection on the "imbalanced ISCX 2012 datasets" with the "Random Tree" are shown.

Table 11: Results obtained from implementing feature selection on the imbalanced dataset using Random Tree

Metric	Normal vs. attack	DoS Attacks	DoS vs. Non-Dos	Non-DoS attacks
Accuracy	99%	99%	82%	100%
F1-score	99%	99%	82%	100%
D. rate	99%	99%	90%	100%
Precision	99%	99%	76%	100%
FAR	0.6%	0.02%	24%	0%

Each column in Table 11 represents the performance metrics of one subtree in the overall tree structure.

As shown in Table 11, the "Random Tree" has achieved the best classification performance compared to the other algorithms and that can be noticed clearly from the fourth column ("DoS" vs. "non-DoS").

v. J48Graft

In Table 12, the performance metrics after applying "RFA" on the "imbalanced" ISCX 2012 datasets" using "J48Graft". Each column in Table 12 represents the performance metrics of one sub-tree in the overall tree structure. Again, the classification performance obtained from "J48Graft" is close to the classification performance of "SVMs", "J48" and "Random Forest". Therefore, the "Random Tree" algorithm has achieved the best classification performance over all the other algorithms.

Table 12: Results obtained from implementing feature selection on the imbalanced dataset using J48Graft

Metric	Normal vs. attack	DoS Attacks	DoS vs. Non-Dos	Non-DoS attacks
Accuracy	99%	99%	73%	100%
F1-score	99%	99%	77%	100%
D. rate	99%	99%	98%	100%
Precision	99%	99%	64%	100%
FAR	1%	0.02%	47%	0%

Now, to summarize the results obtained by applying the aforementioned classifiers to both "balanced" and "imbalanced" datasets, we will investigate which algorithm performs best across most metrics. Since all algorithms recorded in proximity result in all sub-trees except the "DoS" vs. "Non Dos" subtree, we will determine which algorithm performs best on this subtree.

For accuracy, the "Random Tree" classifier was superior to the other classifiers, achieving 90% and 82% on the "balanced" and "imbalanced" datasets, respectively. Similarly, for the *F1−score*, the "Random Tree" also recorded the best performance by scoring 89% and 82% on the "balanced" and "imbalanced" datasets, respectively. However, the situation was different for the detection rate, as both "J48" and "J48Graft" achieved 98% on the "balanced" datasets. On the "imbalanced" datasets, all of "SVM", "J48", and "J48Graft" achieved 98% detection rate.

The fourth metric was the precision. With the "balanced" dataset, both the "SVM" and "Random Tree" recorded the best score 99%. With the "imbalanced" dataset, the "Random Tree" outperformed the other algorithms, scoring 76%. Lastly, with the "FAR" metric, the "SVM" recorded the lowest score at 0.01%, and the next-closest was the "Random Tree," which recorded 0.1%. With the "imbalanced" dataset, the "Random Tree" achieved the lowest score 24% as compared to the other classifiers.

10.3. Constructing the ultimate confusion matrix for multi-class IDS detection

From the aforementioned tables, a wide range of results is obtained, which needs to be analyzed. Therefore, to gain better insight into the results and identify which machine learning algorithm performs best for each dataset, it is decided to calculate the final confusion matrix for each algorithm and both datasets ("balanced" and "imbalanced").

As the binary tree methodology is adopted to assess the effectiveness of "feature

selection", numerous confusion matrices are derived.Each confusion matrix corresponds to a specific sub-tree and comprises 2x2 cells. As a result, some computations are conducted to form the ultimate confusion matrix, which comprises cells in a 5x5 table. As a result, two final confusion matrices are generated; one for each dataset distribution.

As mentioned previously, the "binary tree structure" employed in our approach comprises four binary sub-trees. Consequently, applying machine learning to these subtrees yields four binary confusion matrices. The final confusion matrix is then constructed by combining these individual matrices. Presented below is a detailed methodology outlining the steps involved in computing the final confusion matrix for "multi-class IDS".

Table 13 presents the first confusion matrix for the "normal" vs. "attack" sub-tree when employing "SVMs" on the "balanced ISCX datasets".

Initially, the proportion of each cell relative to its corresponding class must be determined. For instance, dividing 9516 (the value at the top left of Table 13 representing the count of correctly classified "normal" examples) by the total number of the "normal" class (9555) yields 99.59%.

Table 13: The first confusion matrix depicting the "normal" vs. "attack" sub-tree when utilizing "SVMs" on the "balanced ISCX datasets"

	Predicted Normal	Predicted Attack
Actual Normal	9516	39
Actual Attack	70	9486

Similarly, when 39 is divided by 9555, the result is 0.41%.

Equivalently, for the "attack" class, dividing 70 by the total number of samples in this class (9556) yields approximately 0.73%. Dividing 9486—the value at the bottom right of Table 13, representing the number of correctly classified "attack" instances—by 9556 yields 99.27%.

Table 14 presents the computed percentages for the "normal" and "attack" classes.

Table 14: The percentages of the "normal" and "attack" classes, derived from the corresponding confusion matrix

	Percentage of predicted Normal	Percentage of predicted Attack
Actual Normal	99.59%	0.41%
Actual Attack	0.73%	99.27%

Likewise, the confusion matrix for the "non-DoS" vs. "DoS attacks" subtree is shown in Table 15.

Table 15: The confusion matrix depicting the "Non-DoS" vs. "DoS" sub-tree when utilizing "SVMs" on the "balanced ISCX" datasets

	Predicted "Non-DoS"	Predicted "DoS"
Actual "Non-DoS"	5088	4
Actual "DoS"	1886	2523

Additionally, the percentages of each class have been computed and are presented in Table 16.

Table 16: The percentages of "Non DoS" and "DoS" attacks computed based on the corresponding confusion matrix

	Percentage of Predicted "Non-DoS"	Percentage of Predicted "DoS"
Actual "Non-DoS"	99.92%	0.08%
Actual "DoS"	42.78%	57.22%
Average:	71.35%	28.65%

In this table, the column averages are calculated, as they will be necessary for determining the ultimate confusion matrix for "multi-class IDS".

Table 17 presents the intermediate confusion matrix for the "internal" vs. "brute-force" attacks (both categorized as "non-DoS attacks") of the third subtree.

Table 17: The intermediate confusion matrix for the sub-tree comparing "Internal" and "Brute-force" attacks, using "SVMs" on the "balanced ISCX" datasets

	Predicted "Internal"	Predicted "Brute-force"
Actual "Internal"	2547	0
Actual "Brute-force"	0	2545

From the derived percentages in the preceding confusion matrix, it can be observed that the results are perfect, as illustrated in Table 18.

Table 18: The percentages of "Internal" and "Brute-force" attacks calculated based on the corresponding confusion matrix

	Percentage of Predicted "Internal"	Percentage of Predicted "Brute-force"
Actual "Internal"	100%	0%
Actual "Brute-force"	0%	100%
Average:	50%	50%

Lastly, the fourth sub-tree is devoted to "DoS" attacks, particularly "HTTP" and "DDoS", and the consequent confusion matrix for this sub-tree is shown in Table 19.

Table 19: The confusion matrix for the sub-tree distinguishing between "HTTP DoS" and "DDoS" attacks, obtained using "SVMs" on the "balanced ISCX" datasets

	Predicted "HTTP DoS"	Predicted "DDoS"
Actual "HTTP DoS"	2522	1
Actual "DDoS"	0	1886

In a manner akin to the prior sub-tree, the percentages are calculated using the confusion matrix for "DoS" attacks, as depicted in Table 20.

Table 20: The percentages of "HTTP DoS" and "DDoS" attacks calculated based on the corresponding confusion matrix

	Percentage of Predicted "HTTP DoS"	Percentage of Predicted "DDoS"
Actual "HTTP DoS"	99.96%	0.04%
Actual "DDoS"	0%	100%
Average:	49.98%	50.02%

Now, to generate the ultimate confusion matrix, the procedure starts from the top of the tree structure depicted in Figure 1. Beginning with the first row, the first cell symbolizes the accurately classified "normal" connections. The value for the first cell in the first row is directly extracted from the top left cell of Table 14.

Nonetheless, for the other cells in the first row, approximations should be used since precise values are unavailable. These cells depict the "normal" connections that have been "misclassified," including those classified as "attacks" and "internal" attacks. In accordance with the binary tree structure, the proportion of "normal" connections that are "misclassified" as attacks (in general) is utilized as an initial reference point. This value is derived from the top-right cell of Table 14 and, in this instance, amounts to 0.41%.

Subsequently, the process continues to evaluate misclassified "normal" connections as "internal" and "brute-force" attacks, adhering to the binary tree structure. Initially, the percentage of "normal" connections misclassified as "attacks" (0.41%) is multiplied by the mean percentage of classifying any "attack" as a "non-DoS" attack (71.35%), derived from the lower left cell of Table 16. Consequently, this results in a value of 0.15% for "normal" connections that are inaccurately classified as "internal" attacks. Equivalently, for "normal" connections erroneously classified as "brute-force" attacks, the process proceeds along the right branch of the "non-DoS" sub-tree. Multiplying the preceding two values (0.41% and 71.35%) with the mean percentage of classifying any "non-DoS" attack as a "brute-force" attack (50%), obtained from the lower right cell of Table 18, results in a value of 0.15%.

Likewise, the third cell of the first row can be computed comparably, indicating the proportion of "normal" connections inaccurately classified as "HTTP DoS" attacks. Following the binary tree structure, the process navigates the "DoS" subtree and takes the left branch.

To calculate this cell, the top-right cell of Table 14 (0.41%) is multiplied by the percentage of attacks classified as "DoS," which is the bottom-right cell of Table 16 (28.65%). Then, this product is multiplied by the percentage of classifying any "DoS" attack as an "HTTP DoS" attack, found in the bottom-left cell of Table 20 (49.98%). The resulting value is 0.06%.

The last cell in the first row represents the percentage of misclassifying normal connections as "DDoS" attacks. Following the binary tree structure, the procedure starts from misclassifying normal connections as attacks. Then, the "DoS" sub-tree is traversed,and the right branch is taken. To calculate this cell, the top-right cell of Table 14 (0.41%) is multiplied by the percentage of classifying any attack as a "DoS" attack, which is the bottom-right cell of Table 16 (28.65%). Then, this product is multiplied by the percentage of classifying any "DoS" attack as a "DDoS" attack, found in the bottom-right cell of Table 20 (50.02%). The resulting value is 0.06%.

The remaining rows are calculated similarly as the Normal row. The ultimate confusion matrix is shown in Table 21. It is worth noting that each row totals 100%, since the percentage for that row might be distributed across all columns in the case of "misclassification". In all subsequent tables, the shaded cells indicate the "estimated" values for the corresponding class.

Since there are two datasets' distributions ("balanced" and "imbalanced"), there are two confusion matrices. The experimental results are shown after applying feature selection and machine learning algorithms on both datasets. The results are depicted in a 5x5 confusion matrix for the "multi-class intrusion detection" problem. Commonly, the most favorable results achieved by an algorithm are positioned along the "main diagonal" of the confusion matrix. In the following, the results of applying each machine learning algorithm are shown:

a) SVMs

Table 21 shows the final confusion matrix after applying feature selection and the "SVMs" classifier on the "balanced ISCX datasets". Note: each row adds up to 100%.

Table 21: The final confusion matrix after applying feature selection with "SVMs" on the"balanced ISCX datasets".

Actual class		Predicted Class				
		Normal	Intern.	B.Force	HTTP	DDoS
	Normal	99.59%	0.15%	0.15%	0.06%	0.06%

	Intern.	0.73%	99.19%	0.00%	0.04%	0.04%
	B.Forc e	0.73%	0.00%	99.19%	0.04%	0.04%
	HTTP	0.73%	21.23%	21.23%	56.78%	0.02%
	DDoS	0.73%	21.23%	21.23%	0.00%	56.80%

It can be noticed from Table 21 that normal connections are 99.59% correctly classified, which eventually means fewer false positives, as the remainder (0.41%) is distributed on the other classes. In addition, both "Internal" and "brute-force" attacks are 99.19% correctly classified. However, the "SVM" with feature selection achieved 56.78% and 56.80% against the "HTTP DoS" and "DDoS" attacks, respectively, which is lower than the percentages achieved against the "internal" and "brute-force" attacks.

The same procedure is applied to construct the ultimate confusion matrix for the "imbalanced ISCX datasets" as for the "balanced datasets". However, only the final confusion matrices for each machine learning algorithm are shown, without showing the intermediate steps, as they followed the same process as the "balanced datasets".

Table 22: The final confusion matrix after applying feature selection with "SVMs" on the "imbalanced ISCX datasets".

		Predicted Class				
		Normal	Intern.	B.Forc e	HTTP	DDoS
Actual class	Normal	99.29%	0.10%	0.10%	0.26%	0.26%
	Intern.	0.87%	51.74%	0.00%	23.74%	23.64%
	B.Forc e	0.87%	0.00%	51.74%	23.74%	23.64%
	HTTP	0.87%	0.74%	0.74%	97.63%	0.02%
	DDoS	0.87%	0.74%	0.74%	0.23%	97.41%

Table 22 shows the ultimate confusion matrix for feature selection with "SVMs" on the "imbalanced ISCX dataset".

From Table 22, it can be observed that the normal connections were correctly classified at 99.29%, indicating a false-positive rate of 0.71% using feature selection with "SVMs". It can also be observed that the "non-DoS" attacks ("internal" and "brute-force") were not correctly classified at more than 60%. However, the "DoS" attacks ("HTTP" and "DDoS") were correctly classified by 97.63% and 97.41%, respectively.

b) Random Forest

Table 23 shows the final confusion matrix after applying feature selection and a "Random Forest" classifier to the "balanced ISCX datasets". With the "Random Forest" shown in Table 23, the normal connections were correctly classified at 99.88%, meaning only 0.12% false positives. However, the situation is different with the "Internal" and "Brute-force" attacks, where both were correctly classified at 74.62%. In classifying those two attacks, the "SVM" was superior to the "Random Forest" as the "SVMs" correctly classified them by 99.19%. For the "HTTP" and "DDoS" attacks, "Random Forest" achieved 92.02% and 91.72% classification accuracy, respectively.

Table 23: The final confusion matrix after applying feature selection with "Random Forest"on the "balanced ISCX datasets".

		Predicted Class				
		Normal	Intern.	B.Forc e	HTTP	DDoS
Actual class	Normal	99.88%	0.02%	0.02%	0.03%	0.03%
	Intern.	0.40%	74.62%	0.00%	12.53%	12.45%
	B.Forc e	0.40%	0.00%	74.62%	12.53%	12.45%
	HTTP	0.40%	3.77%	3.77%	92.02%	0.04%
	DDoS	0.40%	3.77%	3.77%	0.34%	91.72%

For the "imbalanced ISCX dataset", Table 24 shows the ultimate confusion matrix after applying feature selection with the "Random Forest".

Table 24: The ultimate confusion matrix after applying feature selection with "Random Forest"on the "imbalanced ISCX datasets"

		Predicted Class				
		Normal	Intern.	B.Forc e	HTTP	DDoS
Actual class	Normal	99.47%	0.11%	0.11%	0.16%	0.16%
	Intern.	0.76%	72.41%	0.00%	13.47%	13.37%
	B.Forc e	0.76%	0.00%	72.41%	13.47%	13.37%
	HTTP	0.76%	4.55%	4.55%	90.13%	0.02%
	DDoS	0.76%	4.55%	4.55%	0.37%	89.78%

It can be noticed that the "Random Forest" has achieved very close performance in classifying the "imbalanced" and "balanced" datasets.

c) J48

Table 25 shows the ultimate confusion matrix after applying feature selection and "J48" on the "balanced ISCX datasets".

Table 25: The final confusion matrix after applying feature selection with "J48" on the "balanced ISCX datasets"

		Predicted Class				
		Normal	Intern.	B.Forc e	HTTP	DDoS
Actual class	Normal	99.88%	0.02%	0.02%	0.04%	0.04%
	Intern.	0.53%	51.90%	0.00%	23.85%	23.71%
	B.Forc e	0.53%	0.00%	51.90%	23.85%	23.71%
	HTTP	0.53%	0.72%	0.72%	97.94%	0.08%
	DDoS	0.53%	0.72%	0.72%	0.36%	97.66%

From Table 25, it can be observed that normal connections were again correctly classified at 99.88%. However, the "J48" classifier performed poorly at distinguishing between "internal" and "brute-force" attacks, achieving only 51.90% accuracy. Moreover, the algorithm achieved 97.94% and 97.66% in classifying "HTTP" and "DDoS" attacks, respectively.

For the "imbalanced ISCX dataset", Table 26 shows the ultimate confusion matrix after applying feature selection with "J48". It can be observed that the performance of the "J48" classifier with the "imbalanced dataset" was almost identical to its performance with the "balanced dataset". Therefore, the results are not repeated here.

Table 26: The ultimate confusion matrix after applying feature selection with "J48" on the"imbalanced ISCX datasets"

Actual class		Predicted Class				
		Normal	Intern.	B.Force	HTTP	DDoS
	Normal	99.04%	0.13%	0.13%	0.35%	0.35%
	Intern.	1.54%	51.38%	0.00%	23.67%	23.42%
	B.Force	1.54%	0.00%	51.38%	23.67%	23.42%
	HTTP	1.54%	0.74%	0.74%	96.97%	0.02%
	DDoS	1.54%	0.74%	0.74%	0.54%	96.45%

d) Random Tree

In Table 27, the ultimate confusion matrix after applying feature selection with the "Random Tree" on the "balanced ISCX dataset" is shown.

Interestingly, the "Random Tree" classifier performed very well at classifying all attacks, as well as normal connections, in the "balanced dataset". The algorithm has correctly classified normal connections, "Internal" attacks, and "Brute-force" attacks with more than 99% accuracy.

In classifying "HTTP" and "DDoS" attacks, the "Random Tree" classifier recorded 80.31% and 80.08%, respectively.

Table 27: The ultimate confusion matrix after applying feature selection with the "RandomTree" on the "balanced ISCX datasets"

Actual class		Predicted Class				
		Normal	Intern.	B.Force	HTTP	DDoS
	Normal	99.78%	0.07%	0.07%	0.04%	0.04%
	Intern.	0.42%	99.46%	0.00%	0.06%	0.06%
	B.Force	0.42%	0.00%	99.46%	0.05%	0.06%
	HTTP	0.42%	9.62%	9.62%	80.31%	0.03%
	DDoS	0.42%	9.62%	9.62%	0.26%	80.08%

For the "imbalanced ISCX dataset", Table 28 shows the final confusion matrix after applying feature selection with the "Random Tree" method. Again, the "Random Tree" classifier has demonstrated its superiority in classifying all classes, as shown in the main diagonal of the confusion matrix in Table 28.

Table 28: The ultimate confusion matrix after applying feature selection with the "RandomTree" on the "imbalanced ISCX datasets"

Actual class		Predicted Class				
		Normal	Intern.	B.Force	HTTP	DDoS
	Normal	99.33%	0.14%	0.14%	0.20%	0.19%
	Intern.	0.75%	74.73%	0.00%	12.31%	12.21%
	B.Force	0.75%	0.00%	74.73%	12.31%	12.21%
	HTTP	0.75%	4.58%	4.58%	90.07%	0.02%
	DDoS	0.75%	4.58%	4.58%	0.37%	89.72%

e) J48Graft

In Table 29, the ultimate confusion matrix after applying feature selection with "J48Graft" on the "balanced ISCX dataset" is shown.

Table 29: The ultimate confusion matrix after applying feature selection with "J48Graft" onthe "balanced ISCX datasets"

Actual class		Predicted Class				
		Normal	Intern.	B.Force	HTTP	DDoS
	Normal	99.90%	0.01%	0.01%	0.04%	0.04%
	Intern.	0.55%	51.89%	0.00%	23.91%	23.64%
	B.Force	0.55%	0.00%	51.89%	23.91%	23.64%
	HTTP	0.55%	0.72%	0.72%	97.89%	0.12%
	DDoS	0.55%	0.72%	0.72%	0.68%	97.33%

It can be observed that "J48Graft" performed poorly at classifying "Internal" and "Brute-force" attacks, achieving only 51.89% accuracy, leading to frequent misclassification as "HTTP" and "DDoS" attacks. However, the classifier performed very well at classifying "HTTP" and "DDoS" attacks, achieving 97.89% and 97.33%, respectively.

For the "imbalanced ISCX dataset", Table 30 shows the ultimate confusion matrix after applying feature selection with "J48Graft". Again, "J48Graft" performed poorly at classifying "Internal" and "Brute-force" attacks, recording 51.74% for both. However, "HTTP" and "DDoS" attacks were correctly classified by 97.66% and 97.13%, respectively.

Table 30: The ultimate confusion matrix after applying feature selection with "J48Graft" onthe "imbalanced ISCX datasets"

Actual class		Predicted Class				
		Normal	Intern.	B.Force	HTTP	DDoS
	Normal	98.95%	0.14%	0.14%	0.38%	0.38%
	Intern.	0.84%	51.74%	0.00%	23.84%	23.58%

	B.Force	0.84%	0.00%	51.74%	23.84%	23.58%
	HTTP	0.84%	0.74%	0.74%	97.66%	0.02%
	DDoS	0.84%	0.74%	0.74%	0.55%	97.13%

10.4. Calculating the overall performance of multi-class intrusion detection for all the classifiers

Now, to find the best classifier(s) from all of the above tables and calculations, the average of every metric is calculated for every classifier from its corresponding tree. The average of all the performance metrics (Accuracy, *F1−score*, detection rate, False Alarm Rate, and Precision) has been calculated for every classifier. These calculations have been made for both the "balanced" and "imbalanced" datasets.

For the "balanced ISCX dataset", the average is calculated across all metrics and algorithms, as shown in Table 31. The best recorded value of each metric is shaded in gray.

Table 31: The overall performance of the "multi-class intrusion detection" after applying feature selection on the "balanced ISCX datasets" for all the classifiers

Performance Metrics		Algorithms				
		SVM	R. Forest.	J48	R. Tree	J48 graft
	Accuracy	92.5 %	95.25 %	92.75 %	97%	93.75 %
	F1-score	94.5 %	95.25 %	93.75 %	96.75 %	92.75 %
	D. rate	89%	97.5%	99%	94.5%	99%
	FAR	0.13 %	6.3%	11.8%	0.08%	11.8%
	Precision	99.25 %	93.5%	90.5%	99.25 %	90.5%

As shown in Table 31, the "Random Tree" algorithm has demonstrated superiority over the other algorithms across all metrics except "Detection Rate". For the Detection Rate, both "J48" and "J48graft" outperformed the other algorithms, achieving 99%.

For the "imbalanced ISCX dataset", again the average of all the metrics for all the algorithms has been calculated as shown in Table 32. The best recorded value of each metric is shaded in gray.

Table 32: The overall performance of the "multi-class intrusion detection" after applying feature selection on the "imbalanced ISCX datasets" for all the classifiers

Performance Metrics		Algorithms				
		SVM	R. Forest.	J48	R. Tree	J48 graft
	Accuracy	92.75 %	94.75 %	92.5%	95%	92.75 %
	F1-score	93.75 %	94.75 %	93.5%	95%	93.75 %
	D. rate	99%	97%	98.75 %	97%	99%
	FAR	11.93 %	6.88%	11.98 %	6.16%	12%
	Precision	90.5%	93%	90.5%	93.5%	90.25 %

As shown in Table 32, the "Random Tree" algorithm again outperformed all other classifiers across all metrics except Detection Rate. Just for the Detection Rate, both the "SVM" and the "J48graft" classifiers outperformed the other algorithms.

11. COMPARISON WITH BASELINES

To ensure a fair and rigorous evaluation, our proposed method is benchmarked against several state-of-the-art approaches. The comparison is conducted based on key metrics and computation time.

11.1. Key Metrics Comparison

To demonstrate the effectiveness of the proposed approach, a baseline comparison based on key metrics (accuracy, precision, recall, *F1-score*, and FAR) with six recent papers has been provided. In Table 33 below, the obtained metrics have been compared with those of the six papers, indicating that this paper employed two data distribution scenarios (balanced and imbalanced). However, the proposed work has an advantage over previous work in terms of data distribution. Table 33 lists the metrics typically measured by NIDS.

Table 33 provides a strong comparison for multi-class intrusion detection on ISCX-2012, showcasing the proposed method's advantages in precision, low false alarms, and balance handling. It positions the work as state-of-the-art while transparently noting data gaps in baselines.

Table 33:The Performance Comparison of Multi-Class Intrusion Detection on ISCX-2012 Dataset (Balanced and Imbalanced Splits)

Ref.	Acc.	Precision	Recall	F1-Score	FAR
[48]	99.78 %	98.79%	99.90 %	99.34 %	NA
[49]	90.91 %	90.85%	NA	NA	NA
[50]	84.83 %	NA	74.06 %	83.02 %	NA
[51]	91.5 %	92.40%	92.60 %	92.10 %	NA
[52]	71.00 %	NA	62.30 %	67.85 %	NA

[53]	96.06 %	NA	NA	NA	2.44 %
Ours (Balanced)	97%	99.25%	99%	96.75 %	0.08 %
Ours (Imbalanc ed)	95%	93.5%	99%	95%	6.15 %

The results show that the imbalanced 99.25% precision and 0.08% FAR dominate [53]'s 2.44% FAR (over 30x better) and edge [48] 98.79% precision, critical for real-world IDS where false positives cost dearly.

In terms of balanced robustness, the 97% accuracy holds firm across balanced/imbalanced splits, outperforming mid-tier baselines ([49]-[52]: 71-91.5%) and nearing elite ones without overfitting. Unlike most baselines that lack FAR or full metrics, the proposed work's dual reporting demonstrates practical superiority on ISCX-2012's skewed classes.

Table 33 compares the predictive performance of our proposed method against several recent state-of-the-art baselines. It is important to note that the experimental conditions across these baseline studies are not strictly standardized. For instance, regarding the ISCX 2012 dataset, reference [49] adopts a 5-shot cross-dataset evaluation protocol, while reference [50] partitions a specific subset into a standard 6:2:2 train-validation-test split. Furthermore, the preprocessing protocols vary significantly; while some methods rely on statistical features, others [49][50] convert varying numbers of raw packets into grayscale images. Because the underlying data pipelines, subset selections, and test splits differ across these studies, a perfectly standardized direct comparison is challenging. However, we present these results alongside ours to provide a comprehensive benchmark of current capabilities across different modeling paradigms on the ISCX 2012 data categories.

11.2. Computation Time Comparison

In intrusion detection systems (IDS), computational efficiency is as critical as classification accuracy, particularly for real-time deployment in high-volume network environments where latency directly impacts threat response times. While our two-stage approach (Feature Selection followed by Hierarchical Classification) achieves state-of-the-art precision (99.25%) and a minimal false alarm rate (0.08%) on the ISCX-2012 dataset, it has been emphasized that its practical viability be evaluated through time-related metrics. To address this, the total processing times for Feature Selection + Hierarchical Classification have been measured on comparable hardware (Intel i5-8250U @1.60GHz, 16GB RAM, no GPU), benchmarked against recent baselines as shown in Table 34.

Table 34 provides a clear runtime comparison of our two-stage IDS (Feature Selection + Hierarchical Classification), directly addressing concerns about computational performance. It benchmarks total processing times against six baselines on ISCX-2012, showcasing the proposed method's superior efficiency—14.7-18.3 seconds total, far below [48]'s 55.7 minutes.

Table 34: Total Runtime Comparison(s) on ISCX-2012 dataset

No.	Reference	Feature Selection Time+ Hierarchical Classification Time
1.	[48]	55.7 min.
2.	[49]	NA
3.	[50]	163 sec.
4.	[51]	NA
5.	[52]	NA
6.	[53]	40.76 sec.
7.	Ours	18.3 sec. (Balanced) 14.7 sec. (Imbalanced)

Table 34 shows that the proposed method is ~3x faster than [53] (40.76s), ~9x faster than [50] (163s), and ~180x faster than [48] (55.7 min = 3342s)—ideal for real-time IDS on modest hardware.

When evaluating the computation times presented in Table 34, it is critical to consider the hardware environments used by the baseline methods. Therefore, the hardware specifications for the baselines used to measure computation time are illustrated in Table 35.

The processing times for our proposed approach were achieved on a highly constrained, low-power mobile processor (Intel Core i5-8250U at 1.6 GHz) with 16 GB of RAM and no dedicated GPU. In stark contrast, several baseline methods benefited from high-performance computing architectures. For example, reference [50] utilized an enterprise-grade setup featuring a 16-core Intel Xeon processor, 64 GB of RAM, and an NVIDIA GeForce RTX 3090 (24 GB) GPU.

Table 35: Hardware specifications for baselines and our framework

Ref.	CPU	RAM	GPU	CUDA	OS
[48]	Intel(R) Core(TM) i5-12400	16 GB	NVIDIA GeForce GTX 4060 8G	11.4.2	Win. 10
[50]	Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz	64 GB	GeForce RTX 3090 (24 GB)	11.1	Ubuntu 20.04 LTS

[53]	Intel Core(TM) processor 3.5 GHz	16 GB	NA	NA	Win. 10
Ours	Intel core i5-8250U, 1.6 GHz	16 GB	NA	NA	Win. 11

Similarly, references [48] and [53] employed modern desktop processors (Intel Core i5-12400 and a 3.5 GHz desktop CPU, respectively) paired with RTX 4060 GPUs. Consequently, our method's computational efficiency is highly competitive, achieving these processing speeds despite operating under severe hardware constraints compared to the baseline environments.

12. CONCLUSIONSANDFUTURE WORK

In this paper, a "multi-class intrusion detection system" using feature selection and five machine learning classifiers has been designed and implemented. The embedded feature selection "RFA" [14] has been employed to select the best subset of features using one of the most well-known intrusion detection datasets, "ISCX 2012". The "RFA" is originally designed to work with a "binary SVM" as a core classifier, and it has been utilized here to build a "multi-class intrusion detection system". The objective of the "multi-class intrusion detection" has been achieved by adopting a multi-level binary tree hierarchy to classify the incoming traffic into one of five classes (one normal and four attack classes). Due to the success of this technique, it is recommended to use it for any "multi-class classification problem" by splitting it into multiple binary classification problems.

To introduce different levels of complexity, two ways of preparing the datasets for the experiments, in terms of the distribution of training and test examples ("balanced" and "imbalanced"), have been adopted. The "balanced" distribution involved giving both the training and the testing datasets the same number of examples (or almost the same). While the "imbalanced" distribution involved constructing a testing dataset nine times the size of the training dataset.

In terms of performance evaluation, the performance of the system using five performance metrics: accuracy, *F1−score*, Detection rate, False Alarm Rate ("FAR"), and precision has been measured. These metrics have been measured at each binary sub-tree of the whole binary tree structure and for both "balanced" and "imbalanced" datasets. Furthermore, a new method for constructing the final confusion matrix for the system from the multiple confusion matricesobtained from each sub-tree is proposed. The ultimate confusion matrix has been built for both datasets ("balanced" and "imbalanced").

In addition, to demonstrate the effectiveness of the proposed approach, a baseline comparison with the state-of-the-art methods has been provided. The comparison involved two parts: the key metrics and the time taken for feature selection and hierarchical classification.

When the final confusion matrix is calculated after applying each machine learning algorithm, the experimental results showed the following:

1. For the "balanced" datasets, it has been observed that using feature selection with machine learning classifiers has improved system performance. Among the five machine learning algorithms, the "Random Tree" classifier has proven superior to all other classifiers in detecting all attacks, especially "DoS" attacks ("HTTP" and "DDoS"), achieving 80.31% and 80.08%, respectively.
2. For the "imbalanced" datasets, the "Random Tree" classifier also achieved the best performance, outperforming all other classifiers. The "Random Tree" was predominant in detecting all attacks, especially the "non-DoS" attacks ("Internal" and "Brute-force"), achieving 74.73% for both, which is better than all the classifiers in detecting those two attacks.
3. None of the classifiers (except the "Random Tree") performed well at detecting all classes, as observed: when a classifier performs well on some classes, it performs poorly on others.
4. By comparing the obtained results with the state of the art, the proposed method showed superior performance across key metrics (Accuracy, Precision, Recall, F1-Score, and FAR) and time for feature selection and hierarchical classification.

However, in terms of the feature set selected, the proposed model ranked the features of the dataset from the most important to the least important as follows:

1. Direction
2. appName
3. destinationTCPFlagsDescription
4. protocolName
5. totalSourceBytes
6. totalDestinationBytes

7. totalDestinationPackets
8. totalSourcePackets
9. sourcePort
10. destinationPort

In the future, we are planning to explore how to apply other feature selection methods on highly class "imbalanced datasets" (i.e., where there are many examples of one class over others), such as the work done in [47], since this "imbalance" could result in a deterioration of classification accuracy [12]. In addition, we intend to apply feature selection to intrusion detection datasets for mobile devices, as attackers increasingly target these devices.

REFERENCES

[1] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Communications Surveys Tutorials, vol. 18, no. 2, pp. 1153–1176, Secondquarter 2016, doi: 10.1109/COMST.2015.2494502

[2] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," IEEE Transactions on Computers, vol. 65, no. 10, pp. 2986–2998, Oct 2016, doi: 10.1109/TC.2016.2519914

[3] B. Asiegbu, C. Ikerionwu, and N. Chima Ajanwachuku, "An intrusion detection system using support vector machine and infinite latent feature selection approach," African Journal of Computing and ICT, vol. 12, pp. 74–84, 09 2019.

[4] V. Bolon-Canedo, N. Sanchez-Marono, and A. Alonso-Betanzos, "Feature selection and classification in multiple class datasets: An application to KDD cup 99 dataset," Expert Systems with Applications, vol. 38, no. 5, pp. 5947 – 5957, 2011, doi: https://doi.org/10.1016/j.eswa.2010.11.028

[5] G. Wang, J. Hao, J. Ma, and L. Huang, "A new approach to intrusion detection using artificial neural networks and fuzzy clustering," Expert Systems with Applications, vol. 37, no. 9, pp. 6225 – 6232, 2010, doi: https://doi.org/10.1016/j.eswa.2010.02.102

[6] Y. Yi, J. Wu, and W. Xu, "Incremental {SVM} based on reserved set for network intrusion detection," Expert Systems with Applications, vol. 38, no. 6, pp. 7698 – 7707, 2011, doi: https://doi.org/10.1016/j.eswa.2010.12.141

[7] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," Expert Systems with Applications, vol. 41, no. 4, Part 2, pp. 1690-1700, 2014, doi: https://doi.org/10.1016/j.eswa.2013.08.066

[8] R. Singh, H. Kumar, and R. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," Expert Systems with Applications, vol. 42, no. 22, pp. 8609-8624, 2015, doi: https://doi.org/10.1016/j.eswa.2015.07.015

[9] C.-R. Wang, R.-F. Xu, S.-J. Lee, and C.-H. Lee, "Network intrusion detection using equality constrained-optimization-based extreme learning machines," Knowledge-Based Systems, vol. 147, pp. 68 – 80, 2018, doi: https://doi.org/10.1016/j.knosys.2018.02.015

[10] M. G. Raman, N. Somu, K. Kirthivasan, R. Liscano, and V. S. Sriram, "An efficient intrusion detection system based on hypergraph-genetic algorithm for parameter optimization and feature selection in support vector machine," Knowledge-Based Systems, vol. 134, pp. 1–12, 2017, doi: https://doi.org/10.1016/j.knosys.2017.07.005

[11] A. Karami and M. Guerrero-Zapata, "A fuzzy anomaly detection system based on hybrid PSO-Kmeans algorithm in content-centric networks," Neurocomputing, vol. 149, Part C, pp. 1253–1269, 2015, doi: https://doi.org/10.1016/j.neucom.2014.08.070

[12] Y. Zhu, J. Liang, J. Chen, and Z. Ming, "An improved NSGA-III algorithm for feature selection used in intrusion detection," Knowledge-Based Systems, vol. 116, pp. 74–85, 2017, doi: https://doi.org/10.1016/j.knosys.2016.10.030

[13] H. Debar, "An introduction to intrusion-detection systems," in Proceedings of Connect, 2000, pp. 1–18.

[14] T. Hamed, R. Dara, and S. Kremer, "An accurate, fast embedded feature selection for SVMs," in Machine Learning and Applications (ICMLA), 2014 13th International Conference on, Dec 2014, pp. 135–140, doi: 10.1109/ICMLA.2014.104

[15] T. Hamed, R. Dara, and S. C. Kremer, "Network intrusion detection system based on recursive feature addition and bigram technique," Computers & Security, vol. 73, pp. 137 – 155, 2018, doi: https://doi.org/10.1016/j.cose.2017.10.011

[16] Y. Wahba, E. ElSalamouny, and G. ElTaweel, "Improving the performance of multi-class intrusion detection systems using feature reduction," arXiv preprint, vol. 13, no. 3, pp. 256-262, 2015, doi: https://doi.org/10.48550/arXiv.1507.06692

[17] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class svm," Journal of King Saud University - Computer and Information Sciences, pp. 462-472, 2016, doi: https://doi.org/10.1016/j.jksuci.2015.12.004

[18] S. K. Sahu and S. K. Jena, "A multiclass SVM classification approach for intrusion detection," in Distributed Computing and Internet Technology. Springer, 2016, pp. 175–181, doi: 10.1007/978-3-319-28034-9_23

[19] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on svm with feature augmentation," Knowledge-Based Systems, vol. 136, pp. 130–139, 2017, doi: https://doi.org/10.1016/j.knosys.2017.09.014

[20] A. Abbas, M. A. Khan, S. Latif, M. Ajaz, A. A. Shah, and J. Ahmad, "A new ensemble-based intrusion detection system for internet of things," Arabian Journal for Science and Engineering, vol.

47, no. 2, pp. 1805–1819, 2022, doi: https://doi.org/10.1007/s13369-021-06086-5

[21] Y. Shin, M. Kim, H. Kim et al., "Towards unbalanced multiclass intrusion detection with hybrid sampling methods and ensemble classification," Applied Soft Computing, vol. 157, p. 111517, 2024, doi: https://doi.org/10.1016/j.asoc.2024.111517

[22] S. Bhandari, A. K. Kukreja, A. Lazar, A. Sim, and K. Wu, "Feature selection improves tree-based classification for wireless intrusion detection," in Proceedings of the 3rd International Workshop on Systems and Network Telemetry and Analytics, ser. SNTA20. New York, NY, USA: Association for Computing Machinery, pp. 19–26, 2020, doi: 10.1145/3391812.3396274

[23] P. Toupas, D. Chamou, K. M. Giannoutakis, A. Drosou, and D. Tzovaras, "An intrusion detection system for multi-class classification based on deep neural networks," in 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019, pp. 1253–1258, doi: 10.1109/ICMLA.2019.00206

[24] M. Damaschk, T. D¨onicke, and F. Lux, "Multiclass text classification on unbalanced, sparse and noisy data," in Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing. Turku, Finland: Link¨oping University Electronic Press, pp. 58–65, Sep. 2019.

[25] A. R. Vasudevan and S. Selvakumar, "Intraclass and interclass correlation coefficient-based feature selection in NIDS dataset," Security and Communication Networks, vol. 8, no. 18, pp. 3441–3458, 2015, doi: 10.1002/sec.1269

[26] Z. Heidarian, N. Movahedinia, N. Moghim, and P. Mahdinia, "Intrusion detection based on normal traffic specifications," International Journal of Computer Network and Information Security, vol. 7, no. 9, p. 32-38, 2015, doi: 10.5815/ijcnis.2015.09.04

[27] A. Sarikaya and B. G. Kilic, "A class-specific intrusion detection model: Hierarchical multi-class IDS model," SN Computer Science, vol. 1, no. 202, pp. 1 – 11, 2020, doi: https://doi.org/10.1007/s42979-020-00213-z

[28] O. Osanaiye, H. Cai, K.-K. R. Choo, A. Dehghantanha, Z. Xu, and M. Dlodlo, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing," EURASIP Journal on Wireless Communications and Networking, vol. 2016, no. 1, p. 130, 2016, doi: 10.1186/s13638-016-0623-3

[29] S. M. H. Bamakan, H. Wang, T. Yingjie, and Y. Shi, "An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization," Neurocomputing, vol. 199, pp. 90–102, 2016, doi: https://doi.org/10.1016/j.neucom.2016.03.031

[30] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets," in Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST-2005), pp. 85–89, Octobe 2005.

[31] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, 2001, doi: https://doi.org/10.1023/A:1010933404324

[32] S. Rukmawan, F. Aszhari, Z. Rustam, and J. Pandelaki, "Classification of infarction using random forest," in Journal of Physics: Conference Series, vol. 1752, no. 1, IOP Publishing, p. 012044 2021, doi: 10.1088/1742-6596/1752/1/012044.

[33] K. Gawthorpe, "Random forest as a model for Czech forecasting," Prague Economic Papers, vol. 0, pp. 1–22, 2021, doi: 10.1088/1742-6596/1752/1/012044

[34] C. C. Aggarwal, Data mining: the textbook. Springer, 2015, doi: https://doi.org/10.1007/978-3-319-14142-8

[35] M. Al Sadig and K. N. A. Sattar, "Developing a prediction model using j48 algorithm to predict symptoms of COVID-19 causing death," IJCSNS, vol. 20, no. 8, p. 80, 2020.

[36] Y. Hayashi and S. Yukita, "Rule extraction using recursive-rule extraction algorithm with j48graft combined with sampling selection techniques for the diagnosis of type 2 diabetes mellitus in the pima indian dataset," Informatics in Medicine Unlocked, vol. 2, pp. 92–104, 2016, doi: https://doi.org/10.1016/j.imu.2016.02.001

[37] R. Jehad and S. A. Yousif, "Fake news classification using random forest and decision tree (j48)," Al-Nahrain Journal of Science, vol. 23, no. 4, pp. 49–55, 2020.

[38] T. N. Shah, M. Z. Khan, M. Ali, B. Khan, and N. Idress, "Cart, j-48graft, j48, id3, decision stump and random forest: A comparative study," University of Swabi Journal(USJ), vol. 2, no. 1, pp. 1–6, 2018.

[39] S. Kalmegh, "Analysis of WEKA data mining algorithm reptree, simple cart and randomtree for classification of indian news," International Journal of Innovative Science, Engineering & Technology, vol. 2, no. 2, pp. 438–446, 2015.

[40] B. Pfahringer, "Random model trees: an effective and scalable regression method," Working Paper, Dept. Comput. Sci., Univ. Waikato, Hamilton, New Zealand, pp. 1–12, 2010.

[41] M. Wadkar, F. Di Troia, and M. Stamp, "Detecting malware evolution using support vector machines," Expert Systems with Applications, vol. 143, p. 113022, 2020, doi: https://doi.org/10.1016/j.eswa.2019.113022

[42] G.-Z. Li, J. Yang, G.-P. Liu, and L. Xue, "Feature selection for multiclass problems using support vector machines," in PRICAI 2004: Trends in Artificial Intelligence, C. Zhang, H. W. Guesgen, and W.-K. Yeap, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 292–300, doi: https://doi.org/10.1007/978-3-540-28633-2_32

[43] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," Computers & Security, vol. 31, no. 3, pp. 357–374, 2012, doi: https://doi.org/10.1016/j.cose.2011.12.012

[44] W. Yassin, N. I. Udzir, A. Abdullah, M. T. Abdullah, H. Zulzalil, and Z. Muda, "Signature-based anomaly intrusion detection using integrated

data mining classifiers," in Biometrics and Security Technologies, 2014 International Symposium on, pp. 232–237, Aug 2014, doi: 10.1109/ISBAST.2014.7013127

[45] Z. Tan, A. Jamdagni, X. He, P. Nanda, R. P. Liu, and J. Hu, "Detection of denial-of-service attacks based on computer vision techniques," IEEE Transactions on Computers, vol. 64, no. 9, pp. 2519–2533, Sept 2015, doi: 10.1109/TC.2014.2375218

[46] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in Proceedings of the fifth annual workshop on Computational learning theory. ACM, 1992, pp. 144–152, doi: 10.1145/130385.130401

[47] P. Zhou, X. Hu, P. Li, and X. Wu, "Online feature selection for high-dimensional class-imbalanced data," Knowledge-Based Systems, vol. 136, pp. 187 – 199, 2017, doi: https://doi.org/10.1016/j.knosys.2017.09.006

[48] J. Mao, X. Yang, B. Hu, Y. Lu, and G. Yin, "Intrusion Detection System Based on Multi-Level Feature Extraction and Inductive Network," Electronics, vol. 14, no. 1, pp. 189–189, Jan. 2025, doi: https://doi.org/10.3390/electronics14010189.

[49] J. Bo, K. Chen, S. Li, and P. Gao, "Boosting Few-Shot Network Intrusion Detection with Adaptive Feature Fusion Mechanism," Electronics, vol. 13, no. 22, pp. 4560–4560, Nov. 2024, doi: https://doi.org/10.3390/electronics13224560.

[50] C. Xu, F. Zhang, Z. Yang, Z. Zhou, and Y. Zheng, "A few-shot network intrusion detection method based on mutual centralized learning," Scientific Reports, vol. 15, no. 1, Mar. 2025, doi: https://doi.org/10.1038/s41598-025-93185-0.

[51] M. Wang, N. Yang, Y. Guo, and N. Weng, "Learn-IDS: Bridging Gaps between Datasets and Learning-Based Network Intrusion Detection," Electronics, vol. 13, no. 6, p. 1072, Mar. 2024, doi: https://doi.org/10.3390/electronics13061072.

[52] M. A. Shyaa, N. Farizah Ibrahim, Z. Binti Zainol, R. Abdullah, and M. Anbar, "Reinforcement Learning-Based Voting for Feature Drift-Aware Intrusion Detection: An Incremental Learning Framework," IEEE Access, vol. 13, no. 1, pp. 37872–37903, Feb. 2025, doi: https://doi.org/10.1109/ACCESS.2025.3544221.

[53] B. Yang, G. Zhang, and K. Wang, "A Federated Deep Transfer Learning Algorithm for Intrusion Detection," International Journal of Information Security and Privacy, vol. 19, no. 1, pp. 1–27, Aug. 2025, doi: https://doi.org/10.4018/ijisp.387079.