Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.079694

ARTICLE

An Innovative Binary Model Framework for Cyberattack Detection and Classification in Imbalanced Domains

Óscar Mogollón-Gutiérrez*, José Carlos Sancho Núñez, Mar Ávila, MohammadHossein Homaei and Andrés Caro

Department of Computer and Telematic Systems Engineering, Universidad de Extremadura, School of Technology, Cáceres, Spain

*Corresponding Author: Óscar Mogollón-Gutiérrez. Email: oscarmg@unex.es

Received: 26 January 2026; Accepted: 31 March 2026; Published: 08 May 2026

ABSTRACT: Cyberattacks have increased in frequency and complexity in recent years, resulting in significant consequences for organizations. The negative consequences of cyberattacks force organizations to implement adequate cybersecurity measures to prevent and mitigate the impact of these attacks. Analysis of network traffic is essential for determining whether a cyberattack has been conducted. Intrusion detection systems (IDS) are used to detect malicious actions or irregularities in information systems. In conjunction with artificial intelligence (AI), they enable the development of intelligent intrusion detection systems. This paper presents an intelligent method of network traffic classification for securing systems with multiple connected devices. A proposed method combines several binary models, one for each type of cyberattack, in a two-step framework. The final system predicts whether new incoming traffic is legitimate or potentially malicious. In the first step, attacks are detected. In the second one, cyberattacks are classified into several categories. The proposal also addresses class imbalance by oversampling minority classes during binary model generation. The method was tested on four datasets related to intrusion detection in order to validate its effectiveness. Compared to classical machine learning models and state-of-the-art approaches, this model showed a significant gain in F1 scores, reaching an F1-score of 0.7213, 0.7754, 0.9340, and 0.9793 for NSL-KDD, UNSW-NB15, CSE-CICIDS2018, and ToN-IoT, respectively.

KEYWORDS: Cyberattack detection; cyberattack classification; multi-model learning; machine learning; imbalanced learning; intrusion detection

1 Introduction

In recent years, the frequency and sophistication of cyberattacks have significantly increased, resulting in a growing trend that has significant consequences for organizations. These attacks can compromise sensitive data, disrupt business operations, and damage an organization's reputation, leading to financial losses and decreased customer trust. As a result, it is imperative for organizations to implement effective cybersecurity measures to prevent and mitigate the impact of these attacks. The consequences of cyberattacks are not limited to the affected organization, as they can also have broader societal impacts such as the compromise of critical infrastructure or the spread of false information. It is crucial for both the public and private sectors to address this growing trend and take steps to protect against these types of threats.

One effective approach to preventing cyberattacks is the development of secure code. This involves designing and implementing software systems to minimize vulnerabilities and maximize the system's security [1]. However, this is not a foolproof method for securing data. There are a multitude of other factors

that can contribute to the vulnerability of an organization's systems. For this reason, studying network traffic is essential to determine whether a cyberattack is taking place or, on the contrary, whether connected devices are being used as expected and, therefore, legitimate.

To address this challenge, intrusion detection systems (IDSs) help identify malicious actions or variations in the use of information systems. According to Buczak and Guven [2], there is a first division of IDSs depending on the cyberattack prior knowledge: IDSs based on the inappropriate use of the systems and IDS as an anomaly detector. The former starts from a set of known attacks and will associate each system state with one of them. The latter is responsible for modeling the legitimate behavior of a system and detecting variations in its behavior. A second level could be established depending on the location of the IDS: network-based or host-based. A network-based IDS monitors traffic within a network, while a host-based IDS analyses activity taking place on a single host.

Using artificial intelligence (AI) as a tool for studying network traffic enables intrusion detection. In recent years, models have emerged in the scientific community that create IDSs based on AI (machine learning [3–5], or deep learning [6–8]). AI makes it possible to build models based on the behavior of a system. Based on the data acquired by the IDS, model development can proceed through either supervised or unsupervised learning methods. Supervised learning involves constructing models using a dataset where each data point is labeled. Conversely, unsupervised learning enables the creation of behavioral patterns utilizing datasets that do not contain predefined labels.

Class imbalance in network datasets is a significant challenge in the field of intrusion detection [9–11]. This occurs when one class (e.g., normal traffic) significantly outnumbers the other class (e.g., malicious traffic). This can lead to poor performance of intrusion detection algorithms, as they may be biased towards the majority class and fail to accurately identify instances of the minority class. To address this issue, various techniques have been proposed, such as oversampling the minority class or undersampling the majority class. However, these methods have limitations and there is ongoing research in the field to find more effective approaches to handling class imbalance in network datasets.

Despite notable progress in AI-driven intrusion detection, the existing literature exhibits several interrelated gaps that limit the practical effectiveness of current approaches. First, most IDS proposals treat class imbalance and classification architecture as independent problems: resampling or cost-sensitive strategies are applied to standard multiclass classifiers without exploiting the structural benefits of problem decomposition. Second, while two-stage detection schemes (detection followed by classification) and One-vs.-Rest (OvR) decomposition have each been studied separately, their systematic integration within a unified framework where OvR-generated binary models are combined with oversampling to simultaneously address both the multiclass classification challenge and the class imbalance problem remains unexplored. Third, existing works typically evaluate their proposals on one or two benchmark datasets, which limits the ability to assess robustness and generalizability across diverse network environments and attack profiles. Fourth, many studies rely on accuracy as a primary evaluation metric, which is inherently misleading in imbalanced settings and can obscure poor detection of minority attack classes such as U2R and R2L. To the best of our knowledge, no prior work has jointly addressed these four limitations within a single coherent framework. This gap motivates the present study.

To address the aforementioned research gaps, this paper presents a model for cyberattack detection and classification based on network traffic analysis using machine learning and deep learning techniques. The proposed framework consists of a two-step classification system. It first detects whether traffic is normal or anomalous and then it classifies the type of attack in the case of suspicious traffic (two-step approach). The selection of this technique provides two main benefits, not only the imbalance problem decreases but also [12] cyberattack detection is improved [13–15]. The One-vs.-Rest approach is implemented to generate

specialized binary models for each attack category, and SMOTE-based oversampling is integrated during binary dataset generation to mitigate class imbalance at the data level. Predictions from each binary model are then processed through an ensemble strategy to determine the final classification of each network trace.

The main contributions of this paper, each directly addressing one or more of the identified research gaps, can be summarized as follows:

- Development of an innovative model for addressing imbalanced classification challenges (Gap 1 and Gap 2). This approach incorporates the Synthetic Minority Oversampling Technique (SMOTE) within the One-vs.-Rest (OvR) dataset generation process, creating balanced binary datasets for each traffic category. Unlike existing approaches that apply resampling to standard multiclass classifiers, our framework structurally integrates oversampling with problem decomposition.
- Creation of a two-step binary model framework system for classification tasks (Gap 2). This design combines OvR-based binary model generation with a two-stage detection-then-classification architecture, simultaneously addressing the multiclass classification challenge and the class imbalance problem within a single unified framework.
- Comprehensive evaluation of the proposed framework across four established datasets: NSL-KDD, UNSW-NB15, CSE-CICIDS2018, and TON-IoT (Gap 3). This multi-dataset evaluation strategy assesses the robustness and generalizability of the proposal across diverse network environments and attack profiles, going beyond the single- or dual-dataset evaluations common in the literature.
- Adoption of imbalance-aware evaluation metrics (Gap 4). The experimental analysis employs Macro-F1 as the primary evaluation criterion, ensuring that both majority and minority classes contribute equally to the performance assessment. This avoids the misleading conclusions that arise from accuracy-centric evaluation in highly imbalanced scenarios.

The remainder of the article is organized as follows: Section 2 describes the state-of-the-art in network traffic detection and classification area. Section 3 provides explanations to describe the proposed classifier scheme. This section also describes data and algorithms used in the experiments and how the performance of each model is assessed. Section 4 details experimental results together with an in-depth discussion for each experiment. Finally, conclusions are presented in Section 5.

2 Related Works

In recent years, there has been a notable increase in research focused on intrusion detection systems. This increase in research activity is primarily attributed to the rising number of devices connected to networks. Additionally, the escalating diversity of cyberattacks necessitates the development of robust intrusion detection systems. Consequently, the scholarly literature contains a wide variety of these systems, each designed to meet the specific needs of the systems they are intended to safeguard.

2.1 Single Classifiers

Some researchers have suggested using traditional classifiers to differentiate between normal and malicious behavior [16]. Other studies engage in multiclass classification, categorizing non-legitimate behaviors into specific types of attacks [17]. For instance, Khammassi and Krichen [18] use genetic algorithm-based feature selection followed by decision trees (DT) for detecting intrusions. Additionally, some investigations emphasize the use of deep learning models involving artificial neural networks. In Ref. [19], a convolutional neural network is employed for both binary and multiclass intrusion classification. In Ref. [20] an innovative deep metric learning methodology for network intrusion detection problems is defined, aiming to couple autoencoders to triplet networks. However, their approach is only proposed for binary scenarios. In Ref. [21]

a hybrid system for a cloud intrusion detection system (CIDS) is proposed. It uses a stacked contractive auto-encoder (SCAE) for feature reduction and a support vector machine (SVM) classification algorithm for the detection of malicious attacks. The authors in [22] propose a method for intrusion detection using feature selection and deep reinforcement learning. In Ref. [23], authors proposed a hybrid stacked autoencoder and SVM system for fast and efficient network intrusion detection in big data frameworks. They demonstrated that removing highly correlated features from the dataset had a small effect on accuracy but reduced the time taken to train the model or predict data. More recently, Devi et al. [24] proposed a federated learning-enabled lightweight IDS using optimized deep learning models (including CNN-LSTM hybrids) for Distributed Denial of Service (DDoS) detection in Internet of Things (IoT) environments, demonstrating that lightweight architectures can achieve competitive detection accuracy under decentralized training constraints. Recent efforts have introduced complex deep learning architectures to handle multiple benchmark datasets. For instance, Ullah et al. [25] proposed a transformer-based transfer learning model for imbalanced traffic, while Arafah et al. [26] applied Wasserstein Generative Adversarial Networks (WGANs) combined with autoencoders to synthesize minority attacks. Although evaluated across multiple datasets such as NSL-KDD and CICIDS-2017, these monolithic deep learning models suffer from high computational overhead and lack the structural simplicity of problem decomposition, which makes them harder to tune for specific minority attack profiles.

2.2 Multi-Model Learning

Multi-model or ensemble learning is an effective technique that involves training multiple models and combining their outputs to improve the overall performance of the system. In the context of intrusion detection, multi-model learning has been shown to be particularly effective at improving the accuracy and robustness of the model. Specifically, in Ref. [27], a combined voting model is implemented where the final class prediction is derived from the probabilities generated by each individual classifier. Similarly, Maniriho et al. [28] employs a combined approach, but in this case, the strategy involves randomly generated classifiers. Their study demonstrates that the binary classifier achieves high accuracy due to the use of a reduced dataset, with 34% of the training set used for testing. Both studies indicate that combining predictions from individual models generally enhances the capability to detect attacks. Recent advancements in multi-model learning, like the stacking ensemble approach proposed by Talukder et al. [29], show that putting feature embedding together with ensemble techniques can catch dynamic cyber threats. But, their architecture is still a direct multiclass framework. This means the ensemble must still learn complex, overlapping boundaries for all classes at the same time instead of decomposing the problem.

2.3 Two-Stage Classification

The two-step scheme for intrusion detection involves first training a classifier to identify normal behavior, and then training a second classifier to identify abnormal behavior based on the output of the first classifier. This approach has several advantages over traditional single-classifier approaches. For example, by first training a classifier to identify normal behavior, the second classifier is able to focus on learning to identify only the subtle deviations from normal behavior that are indicative of an intrusion. Meftah et al. presented a contribution based on this approach in [30]. It is unclear, however, whether their strategy is effective since it is based on a modified dataset, which removes some of the original classes. Following the two-stage approach, Ref. [14] suggests a deep learning system applying deep learning at the second level of classification. Recent studies, such as the hybrid framework by Chen et al. [31], deploy advanced two-stage classifiers to combine network and host data. However, their second stage does not address the internal imbalance among attack subtypes through structural problem decomposition. Similarly, Mahmoud et al. [32]

recently introduced an explainable two-stage system to better capture low-frequency attacks. Despite filtering normal traffic in the first step, their second stage still relies on standard multiclass algorithms, forcing the model to learn multiple overlapping minority boundaries simultaneously.

2.4 One-vs.-Rest Approach

One-vs.-rest is a technique in machine learning for classification tasks with multiple classes. It involves training a separate binary classifier for each class, where the class under consideration is treated as the positive class and all other classes are mixed into a single negative class. In [33], authors achieve high accuracy in detecting different attacks from the BoT-IoT dataset using a Hadoop-Spark strategy. They find that the Random Forest algorithm is effective for training extremely imbalanced datasets in combination with a OVR wrapper. More recently, Kil et al. [34] implemented multi-binary classifiers to improve memory efficiency and feature selection in endpoints. Yet, they isolated this binary decomposition from data-level resampling methods, missing the opportunity to use SMOTE in a lower-dimensional two-class space where synthetic noise is drastically minimized.

2.5 Class Imbalance

Given the severity of the class imbalance problem introduced in Section 1, a substantial body of work has explored different mitigation strategies in the context of intrusion detection. These strategies can be broadly organized into three categories: data-level resampling, cost-sensitive learning, and hybrid approaches. This subsection critically reviews representative contributions in each category and identifies their respective strengths and limitations.

Data-level resampling methods aim to rebalance the class distribution prior to model training. Among oversampling techniques, the Synthetic Minority Oversampling Technique (SMOTE) remains the most widely adopted in the IDS literature. In [35], SMOTE combined with Random Undersampling was applied to UNSW-NB15 and NSL-KDD for binary classification, demonstrating improved minority class detection. However, their evaluation was limited to a two-class setting without addressing multiclass scenarios. Bagui and Li [36] conducted a broader comparison of Random Undersampling, Random Oversampling, SMOTE, and Adaptive Synthetic (ADASYN) for training an artificial neural network, concluding that SMOTE consistently yielded the best trade-off between precision and recall. AlShehari and Alsowail [37] further investigated the effect of under-sampling, over-sampling, and hybrid-sampling strategies across multiple classifiers, finding that the optimal resampling method varied by algorithm: Decision Trees achieved the highest ROC-AUC with undersampling, whereas KNN performed best with oversampling and hybrid sampling. Additionally, Aziz and Ahmad [38] proposed an unsupervised undersampling approach using K-Means clustering for two-class classification on UNSW-NB15, although the generalizability of this approach to multiclass settings remains unexplored. More recently, to solve the non-stop challenge of class imbalance, Le et al. [39] checked the combination of hybrid resampling techniques with ensemble classification. While it is effective, their method does resampling directly inside a multiclass space, which naturally makes the risk of inter-class synthetic noise higher compared to our proposed binary decomposition strategy.

Cost-sensitive learning methods address the imbalance at the algorithmic level by assigning higher misclassification penalties to minority classes. Rani and Gagandeep [40] proposed an IDS based on a cost-sensitive neural network, achieving an improvement of approximately 4% over standard baselines. More recently, Telikani et al. [11] developed an evolutionary cost-sensitive deep learning model that employed an adaptive differential evolution algorithm to optimize misclassification costs during training. Their architecture combined unsupervised feature extraction through a stacked autoencoder with supervised classification via a convolutional neural network, targeting the detection of low-frequency intrusions.

Hybrid approaches combine resampling with advanced classification architectures. Zhang et al. [41] integrated SMOTE oversampling with clustering-based undersampling as a preprocessing pipeline for a convolutional neural network. While their results were promising, a notable limitation is that they did not report imbalance-aware classification metrics (e.g., Macro-F1 or per-class recall), making it difficult to assess the model's true effectiveness on minority classes. Other new works, such as the model introduced by Omer Albasheer et al. [42], use SMOTE combined with Edited Nearest Neighbors (SMOTE-ENN) to clean synthetic noise before applying normal classifiers. However, just like older models, they do not take advantage of the structural benefits of reducing the classification task to independent binary boundaries before they apply the oversampling algorithm.

A common limitation across the reviewed works is that most focus on either resampling or cost-sensitive strategies in isolation, without combining them with structural classification approaches such as One-vs.-Rest decomposition. Furthermore, the majority of these studies evaluate their proposals on one or two datasets, limiting the generalizability of their conclusions. Our work addresses these gaps by integrating SMOTE-based oversampling within an OvR binary model framework and validating the approach across four benchmark datasets.

2.6 Novelty of This Contribution

In our previous work [43], we presented a preliminary two-stage ensemble model for the UNSW-NB15 dataset; the current work extends this approach to a comprehensive binary model framework validated across four different datasets. After an analysis of the scientific literature on intrusion detection and to the best of our knowledge we have not found any previous work that incorporates the use of the OvR strategy into the construction of a binary model framework. Furthermore, our proposed model follows a two-step scheme to reap previously listed benefits. As a summary, Table 1 graphically highlights the innovation between our work against the state of the art using the following criteria:

A These works apply a technique or use a cost-sensitive algorithm to deal with class imbalance. The proposal addresses the imbalance between classes by applying the SMOTE oversampling algorithm during the generation of binary models.

B The research proposal uses several classification algorithms to improve the system's performance. K-Nearest Neighbours, Decision Trees, Support Vector Machines and Multilayer Perceptron algorithms are trained to build binary specialized models for the proposed framework.

C Researchers address intrusion detection by considering a two-step scheme. This refers to the use of a first level of attack detection and, in a second step, suspicious traffic will be classified among preset attack categories.

D The work includes imbalance classification metrics that are not biased by minority/majority classes. Our experimental analysis considers imbalanced when evaluating the performance of the proposed system.

E The number of datasets used to test the proposal. The purpose of this study is to evaluate the efficacy of the proposed system over four well-known datasets. As recently highlighted by Sarhan et al. [44], most machine learning models in intrusion detection fail to generalize when they are tested across diverse datasets because of big variations in feature sets and attack distributions. Our multi-dataset evaluation specifically answers this concern, proving that the modular OvR binary framework consistently adapts to different network topologies without needing highly complex deep learning architectures.

Table 1: Comparison of existing works in field of intrusion detection.

Work	A	B	C	D	E
[14]	No	No	Yes	F1	2
[17]	No	No	No	No	2
[20]	Yes	No	No	F1	4
[19]	No	No	No	F1	2
[21]	No	No	No	F1	2
[16]	No	No	No	F1	3
[11]	Yes	Yes	No	F1	2
Our approach	Yes	Yes	Yes	F1, Macro-F1	4

While OvR decomposition, SMOTE, two-stage classification, and multi-model selection have each been employed individually in the IDS literature, their joint integration within a single coherent framework has not been previously proposed. The novelty of this work resides in their synergistic combination, which produces emergent benefits that no single component achieves alone. Existing two-stage approaches [14,15] employ a detection-then-classification pipeline but use standard multiclass classifiers at the second stage without OvR decomposition, thus failing to exploit per-class specialization or reduce the effective imbalance ratio per task. Ensemble or multi-model approaches [27,28] combine multiple classifiers via voting but apply them to the same multiclass problem without binary decomposition, meaning each model must still learn all inter-class boundaries simultaneously in an imbalanced space. OvR-based approaches [33] employ binary decomposition but use a single algorithm without multi-model selection or two-stage filtering, and do not integrate oversampling within the OvR dataset generation process. SMOTE-based approaches [35,41] apply oversampling to standard multiclass or binary settings without leveraging OvR decomposition, where SMOTE would operate in a lower-dimensional two-class space with reduced inter-class synthetic noise. In contrast, our framework uniquely combines these four components such that: (1) OvR decomposition reduces each classification task to a binary problem, creating an optimal setting for SMOTE application; (2) multi-model selection independently identifies the best-performing algorithm per class, enabling per-class specialization; and (3) the two-stage architecture filters Normal traffic early, reducing the classification workload and further mitigating imbalance at the second stage.

3 Proposed Framework

This section describes the proposed binary model learning framework for detecting and classifying cyberattacks. The framework consists of two main phases: binary model generation and the binary model learning framework itself.

3.1 Binary Model Generation

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ denote the training dataset with N traffic categories $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$. Let $\mathcal{D}_{c_j} \subset \mathcal{D}$ denote the subset of samples belonging to category c_j, with $|\mathcal{D}_{c_j}| = n_j$. For each target category c_j, a binary dataset $\mathcal{B}_j = \mathcal{B}_j^+ \cup \mathcal{B}_j^-$ is constructed as follows. The positive class is defined as $\mathcal{B}_j^+ = \mathcal{D}_{c_j}$, containing all n_j samples of the target category. The negative class $\mathcal{B}_j^-$ aggregates samples from all remaining categories $c_k \in \mathcal{C}\backslash\{c_j\}$. To ensure balanced representation across non-target categories, the quota per non-target category is computed as $n\_samples\_c = \lfloor n_j/(N-1) \rfloor$. For each non-target category c_k: if $|\mathcal{D}_{c_k}| \geq n\_samples\_c$, then $n\_samples\_c$ samples are randomly drawn from $\mathcal{D}_{c_k}$; otherwise, all available samples are retained and

SMOTE is applied to synthesize the deficit ($n\_samples\_c - |\mathcal{D}_{c_k}|$ additional samples). The resulting binary dataset $\mathcal{B}_j$ contains n_j positive samples and $\sum_{k\neq j} n\_samples\_c$ negative samples, yielding an approximately balanced two-class dataset. Algorithm 1 formalizes this procedure.

Algorithm 1: Binary dataset generation

Require: Training dataset $\mathcal{D}$, categories $\mathcal{C} = \{c_1, \ldots, c_N\}$, algorithms $\mathcal{A} = \{a_1, \ldots, a_M\}$
Ensure: Set of best binary models $\mathcal{M}^* = \{m_1^*, \ldots, m_N^*\}$

for each category $c_j \in \mathcal{C}$ **do**
  $\mathcal{B}_j^+ \leftarrow \mathcal{D}_{c_j}$ {Positive class: all samples of target category}
  $n_j \leftarrow |\mathcal{B}_j^+|$
  $n\_samples\_c \leftarrow \lfloor n_j/(N-1) \rfloor$
  $\mathcal{B}_j^- \leftarrow \varnothing$
  for each category $c_k \in \mathcal{C}\backslash\{c_j\}$ **do**
    if $|\mathcal{D}_{c_k}| \geq n\_samples\_c$ **then**
      $\mathcal{S}_k \leftarrow \text{RANDOMSAMPLE}\ (\mathcal{D}_{c_k},\ n\_samples\_c)$
    else
      $\mathcal{S}_k \leftarrow \mathcal{D}_{c_k} \cup\ \text{SMOTE}\ (\mathcal{D}_{c_k},\ n\_samples\_c - |\mathcal{D}_{c_k}|)$
    end if
    $\mathcal{B}_j^- \leftarrow \mathcal{B}_j^- \cup \mathcal{S}_k$
  end for
  Relabel: $y_i \leftarrow +1\ \forall(\mathbf{x}_i, y_i) \in \mathcal{B}_j^+;\quad y_i \leftarrow -1\ \forall(\mathbf{x}_i, y_i) \in \mathcal{B}_j^-$
  $\mathcal{B}_j \leftarrow \mathcal{B}_j^+ \cup \mathcal{B}_j^-$
  for each algorithm $a_m \in \mathcal{A}$ **do**
    $m_{j,m} \leftarrow \text{GRIDSEARCHCV}(a_m,\ \mathcal{B}_j,\ K\text{-fold},\ \text{Macro-F1})$
  end for
  $m_j^* \leftarrow \arg\max_{m_{j,m}} \text{Macro-F1}(m_{j,m})$ {Select best algorithm for c_j}
end for
return $\mathcal{M}^* = \{m_1^*, \ldots, m_N^*\}$

Oversampling of underrepresented categories was performed using SMOTE as implemented in the Imbalanced-learn library (version 0.12). The default SMOTE configuration was used: $k\_neighbors = 5$ (i.e., each synthetic sample is generated by interpolating between a minority instance and one of its five nearest neighbors in feature space), and the `sampling_strategy` was set to equalize the number of samples between the target class and the aggregated non-target class in each binary dataset, as showed in Fig. 1. SMOTE was applied exclusively to the training data, the held-out test set was never exposed to oversampling. This protocol eliminates the risk of data leakage and guarantees that all evaluation metrics reflect the model's ability to generalize to unseen, unmodified data.

The choice of SMOTE as the oversampling component is grounded in the following theoretical and empirical considerations. First, SMOTE generates synthetic minority samples by interpolating between existing instances in feature space, rather than simply duplicating them. This produces more diverse training data that helps classifiers learn more generalizable decision boundaries, reducing the overfitting risk inherent to naive random oversampling [36]. Second, SMOTE has been independently validated across a wide range of classifiers including KNN, SVM, DT, and neural networks and consistently demonstrates a favorable trade-off between precision and recall [36,37]. This cross-classifier robustness is critical in our multi-algorithm framework, where the oversampled binary datasets must be compatible with all four candidate learners. Third, SMOTE operates as a model-agnostic, data-level strategy that requires no modifications

to the classifier's internal loss function or training procedure. This is an important design consideration given that our framework already introduces substantial structural complexity through OvR decomposition, multi-model selection, and two-step classification. Adding algorithm-level modifications (e.g., cost-sensitive tuning per classifier and per class) would increase the hyperparameter search space considerably and reduce the generalizability of the framework. Fourth, in the specific context of our OvR framework, SMOTE is applied to binary datasets containing only two classes (target category vs. rest), which substantially reduces the risk of inter-class synthetic noise that SMOTE can introduce when applied in high-dimensional multiclass settings with overlapping minority classes [35].

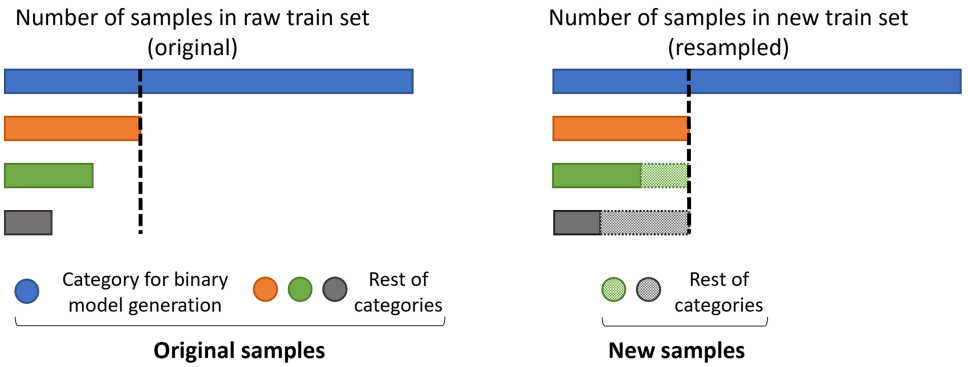

Figure 1: Detailed dataset distribution for binary model generation.

We acknowledge that alternative imbalance-handling strategies exist, including ADASYN (Adaptive Synthetic Sampling), Borderline-SMOTE, cost-sensitive learning, and focal loss. Each of these methods offers potential complementary benefits. A systematic comparative evaluation of these alternatives within the proposed OvR binary framework constitutes an important direction for future work, as discussed in Section 5.

Once binary datasets are generated, generation of the binary model is performed. To accomplish this, hyperparameter tuning together with preprocessing is performed to generate the model that best fits the dataset. This process is repeated for four different classification algorithms (K-Nearest Neighbours (KNN), Support Vector Machine (SVM), Decision Trees (DT) and Multilayer Perceptron (MLP)). This optimization uses Grid Search and 5-Fold Cross-Validation methodology, taking the Macro-F1 metric as an evaluation criterion during the hyperparameter fitting process. The reason for considering Macro-F1 lies in the fact that the new dataset has two main classes and, therefore, both classes are considered equally important.

The decision to utilize the OvR strategy in our binary model framework, as opposed to a standard multiclass classifier or a hierarchical classification architecture, was driven by several theoretical and practical considerations. Table 2 summarizes the comparative analysis of three classification paradigms commonly employed in intrusion detection: direct multiclass, hierarchical, and OvR binary decomposition.

Table 2: Comparative analysis of classification paradigms for intrusion detection in imbalanced domains.

Criterion	Direct Multiclass	Hierarchical	OvR Binary (Ours)
Imbalance handling	Difficult; minority classes are overshadowed by the joint majority	Partially mitigated at each hierarchical level	Naturally reduced; each binary problem has only two classes

(Continued)

Table 2 (continued)

Criterion	Direct Multiclass	Hierarchical	OvR Binary (Ours)
Model specialization	Single model must learn all decision boundaries simultaneously	Specialized at each hierarchical level, but dependent on upper-level accuracy	Each binary model is independently specialized for one class
Resampling compatibility	SMOTE operates in a high-dimensional multiclass space, risking inter-class noise	Applied per hierarchical level with moderate complexity	SMOTE is applied per binary dataset, reducing synthetic noise between similar minority classes
Error propagation	No cascading errors, but global confusion among all classes	Errors at upper levels cascade irreversibly to lower levels	No cascading; each model predicts independently
Interpretability	Low; difficult to identify which class causes misclassification	Moderate; tree structure aids analysis	High; per-class model performance is directly observable
Scalability to new attacks	Requires full retraining	Requires restructuring the hierarchy	Only a new binary model needs to be added
Training cost	Single model training	Multiple levels of training	N binary models (parallelizable)

Regarding advantages of binary decomposition over direct multiclass classification, in standard multiclass classification, a single model must simultaneously learn the decision boundaries among all classes. In intrusion detection datasets, where the class distribution is severely skewed (e.g., Normal traffic may constitute over 80% of samples while U2R attacks represent less than 0.1%), the multiclass learner tends to be dominated by the majority class, leading to poor recall on minority attack categories [12]. By decomposing the problem into N independent binary tasks via OvR, each binary model focuses exclusively on distinguishing one specific class from all others. This decomposition inherently reduces the effective imbalance ratio per task, since each binary dataset only contains two classes. Furthermore, oversampling techniques such as SMOTE can be applied more effectively in a binary setting, where synthetic samples are generated in a lower-dimensional class space with reduced risk of inter-class noise.

In the context of hierarchical approaches (e.g., first detecting attacks, then classifying subtypes) introduce a dependency between levels: if the first-level detector misclassifies an attack as legitimate traffic, the second level never has the opportunity to classify it correctly. This error propagation problem has been documented in prior IDS works [30]. Our framework mitigates this risk through a two-step design where the first step acts as a filter, but each second-step binary model operates independently.

The OvR approach is not without limitations. First, it requires training N binary models (one per class) with M candidate algorithms each, resulting in $N \times M$ models during the generation phase. However, as discussed in Section 3.2, this cost is substantially mitigated by the inherent parallelizability of the training process, as each binary model is independent. Second, the final classification relies on combining the probability outputs of independently trained models, which may introduce calibration inconsistencies when the individual models exhibit different confidence scales. We address this by selecting the class with

the highest probability across all binary models, a strategy that has been shown to be effective in OvR settings [12]. Third, in datasets with many categories (e.g., UNSW-NB15 with 10 traffic types), the OvR approach may generate binary datasets where the "rest" class is highly heterogeneous, potentially making the binary boundary harder to learn for certain models. Our experimental results (Section 4) demonstrate that despite this limitation, the framework achieves competitive or superior performance across all four datasets evaluated. The framework for binary model generation is illustrated in Fig. 2.

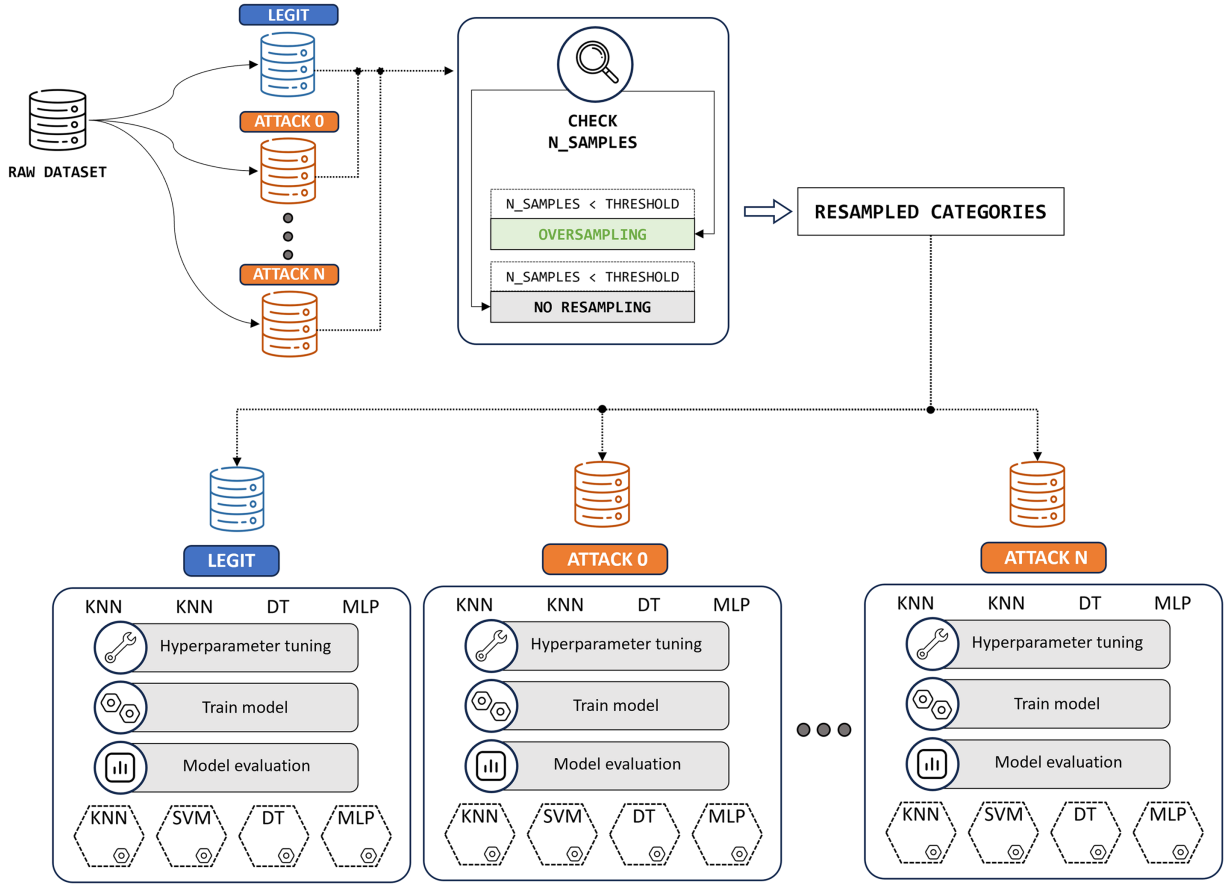

Figure 2: Binary models generation for a dataset with N different categories.

Regarding hyperparameter tuning, Grid Search and Cross-Validation were applied. Table 3 shows the hyperparameter search space for each of the applied algorithms.

The hyperparameter search ranges reported in Table 3 were defined based on two complementary sources of evidence. First, the selected ranges reflect the most commonly evaluated hyperparameter spaces in the IDS literature for each algorithm [36,45]: for KNN, $k \in \{1, \ldots, 30\}$ covers the range empirically shown to balance the bias–variance trade-off in network traffic datasets; for SVM, $C \in \{1, \ldots, 1000\}$ and $\gamma \in \{10^{-5}, \ldots, 1\}$ span the standard logarithmic grid used for RBF kernels; for DT, $\texttt{max\_depth} \in \{1, \ldots, 35\}$ prevents overfitting while allowing sufficient tree complexity for multi-feature traffic patterns; and for MLP, the learning rate and batch size ranges follow standard practice for shallow neural networks on tabular data [36]. Regarding, the Grid Search with 5-Fold Cross-Validation methodology serves as an implicit sensitivity analysis: all combinations within the defined search space are evaluated and the Macro-F1 score

is computed across five independent validation folds for each configuration, systematically revealing the sensitivity of model performance to hyperparameter variations before the best configuration is selected.

Table 3: Hyperparameters search space per algorithm.

Algorithms	Hyperparameters	Values
KNN	n_neighbors	{1 . . .30}
	Weights	{uniform, distance}
	Algorithm	{auto, ball_tree, kd_tree}
SVM	C	{1 . . .1000}
	Gamma	$\{1 \times 10^{-5} \ldots .1\}$
DT	Criterion	{gini, entropy}
	max_depth	{1 . . .35}
MLP	Activation	{relu, tanh}
	learning_rate_init	$\{1 \times 10^{-5} \ldots .1\}$
	max_iter	{20, 40, 60, . . ., 200}
	batch_size	{32, 64, 128, 256, auto}

3.2 Binary Model Learning Framework

Fig. 3 illustrates the proposed binary model learning framework. On the left side, the top-performing binary model for each category is selected. In detail, Fig. 3A shows how binary classifiers are independently trained and evaluated, as previously explained. This process is executed separately and in parallel for every category. Upon completion of the training phase, each candidate model undergoes evaluation using a classification metric. The resulting scores are compared across all classifiers within the same category to objectively rank their relative performance. The model yielding the highest evaluation score is identified as the top-performing candidate for that category, as visually indicated by the highlighted score in Fig. 3B. In the final phase, the best-performing model is selected as the sole representative binary classifier for its respective category, as depicted in Fig. 3C. The selected models are subsequently integrated into the learning framework, illustrated on the right side of Fig. 3.

The first phase of the proposed system, focused on attack detection, consists of analyzing all traffic and determining whether it is a legitimate or a suspected attack. In order to accomplish this task, a binary model has previously been generated. Specifically, the model will output two probabilities: the first will indicate whether the sample of traffic is legitimate and the second will indicate whether it may be a potential attack. Based on the output with the highest value, the nature of the network trace will be determined.

The initial task was to classify behavior as either Normal or an Attack (detection level or attack detection), utilizing a specifically trained binary model. The second level involves classifying a potential threat as one of the possible attacks collected in any intrusion dataset (classification level).

Traffic classification is performed using a multi-model framework composed of the binary classifiers of several attack types. Each classifier generates probabilities for class membership for all classes. The class with the highest probability is selected as the final prediction. Subsequently, evaluation metrics are computed based on the resulting confusion matrix. Algorithm 2 formalizes the two-step inference procedure.

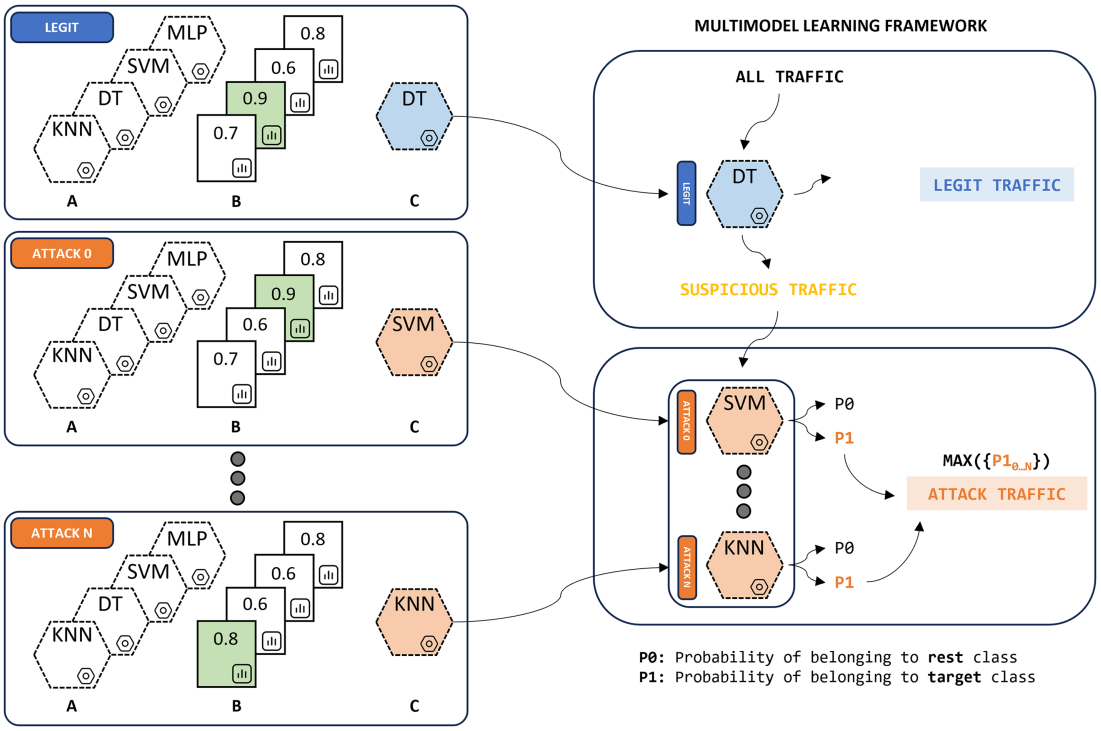

Figure 3: Binary model learning framework.

Algorithm 2: Two-step inference

Require: Test sample $\mathbf{x}$, binary models $\mathcal{M}^* = \{m_1^*, \ldots, m_N^*\}$ where m_1^* corresponds to Normal traffic
Ensure: Predicted class $\hat{y}$

Step 1—Attack Detection:
$(p_{\text{Normal}}, p_{\text{Attack}}) \leftarrow m_1^*(\mathbf{x})$ {Binary probabilities from Normal model}
if $p_{\text{Normal}} \geq p_{\text{Attack}}$ **then**
  return $\hat{y} \leftarrow$ Normal
end if
Step 2—Attack Classification:
for each attack model m_j^*, $j = 2, \ldots, N$ **do**
  {Parallelizable}
  $p_j^+ \leftarrow P(y = c_j \mid \mathbf{x}, m_j^*)$ {Positive-class probability}
end for
return $\hat{y} \leftarrow c_{\arg\max_{j \in \{2,\ldots,N\}} p_j^+}$

Traffic classification is performed using a multi-model framework composed of the binary classifiers of several attack types. Each classifier generates probabilities for class membership for all classes. The class with the highest probability is selected as the final prediction. Subsequently, evaluation metrics are computed based on the resulting confusion matrix.

In the two-step multimodel classification system described, the training and inference times play distinct roles in the overall performance and efficiency of the model. During the training phase, the system employs a binary classifier for each category, which significantly influences the training duration. This approach, while methodical and thorough, necessitates a substantial amount of computational resources and

time, as each classifier must be individually trained on its respective subset of data. This process ensures that each binary classifier is finely tuned to its specific category, leading to improved accuracy and robustness in the classification task. Despite the required training time, it should be considered that the architecture of having separate models for each category allows for parallel processing during training, further enhancing the speed.

Contrastingly, the inference time in this system is notably faster, despite the multimodel approach. The speed of inference primarily depends on the number of categories present, as each category corresponds to a binary classifier that makes an independent prediction. This streamlined process is significantly more efficient than the training phase, as it does not require the intensive computational resources necessary for learning. Similarly to the inference, parallelization can be performed in the second step of classification when outputs from several models are combined to classify the network trace.

Our approach introduces novel variations in the combination and integration of classifiers. Our study focuses on the application of the OvR strategy within this framework, which allows each classifier to distinguish a specific attack class from the rest. This combination of the OvR strategy with multiple classifiers has not been extensively explored in the context of attack classification, and its effectiveness is demonstrated through several experimental evaluations. Furthermore, our work contributes by proposing and evaluating a specific oversampling method tailored to our multi-attack classification environment. We carefully analyze the impact of oversampling and demonstrate the effectiveness of our chosen method in mitigating the class imbalance problem, leading to improved classification performance.

Computational Complexity

Let N denote the number of traffic categories, M the number of candidate algorithms, n the number of training samples per binary dataset, d the number of features, and K the number of cross-validation folds. The computational cost of the proposed framework can be analyzed in two phases.

Training phase. For each of the N categories, M binary models are trained via K-fold cross-validation with Grid Search over a hyperparameter grid of size G. The total number of model fits is $\mathcal{O}(N \times M \times K \times G)$. The cost per individual fit depends on the algorithm: $\mathcal{O}(n^2 \cdot d)$ for KNN (distance computation), $\mathcal{O}(n^2 \cdot d)$ to $\mathcal{O}(n^3)$ for SVM with RBF kernel, $\mathcal{O}(n \cdot d \cdot \log n)$ for DT, and $\mathcal{O}(n \cdot d \cdot h \cdot e)$ for MLP where h and e are the hidden layer size and number of epochs, respectively. Crucially, since binary models are independent, all $N \times M$ training tasks are parallelizable, reducing the wall-clock time to that of the slowest single model when sufficient computational resources are available.

Inference phase. For a new sample, the first step applies a single binary model (Normal vs. Attack) at cost $\mathcal{O}(C_1)$. If classified as an attack, the second step evaluates $N-1$ binary classifiers in parallel, each at cost $\mathcal{O}(C_i)$, and selects the class with the highest probability. The total sequential inference cost is $\mathcal{O}(C_1 + \sum_{i=2}^{N} C_i)$, or $\mathcal{O}(C_1 + \max_i\{C_i\})$ under parallelization, where C_i is the prediction cost of the i-th binary model. In practice, this is comparable to a single multiclass model, since the individual binary classifiers operate on the same feature dimensionality with simpler two-class decision boundaries.

3.3 Datasets

One of the main concerns when dealing with intrusion datasets is class imbalance. As the prevalence of cyberattacks continues to increase in digital society, it is imperative to have access to reliable and accurate datasets for developing effective cybersecurity systems. Four widely used cybersecurity datasets are used in this work to evaluate the effectiveness of the system: NSL-KDD [46], UNSW-NB15 [47], CIC-IDS2018 [48], and TON-IoT [49]. These datasets have been utilized in various academic research and industry studies to

evaluate the efficiency of cybersecurity techniques such as intrusion detection systems, malware analysis, and network anomaly detection.

Table 4 presents the per-class sample counts in the training partition of each dataset, together with the percentage share (%) and the Imbalance Ratio (IR). The IR is defined as the ratio between the number of samples in the largest class and the number of samples in the class under consideration, i.e., $\mathrm{IR}_c = N_{\max}/N_c$. An IR value greater than 1.5 is conventionally considered to indicate a meaningful class imbalance [12].

Table 4: Per-class sample distribution and Imbalance Ratio (IR) for each dataset. Bold values highlight the extreme minority classes (highest IR) for each dataset, indicating the specific attack types where the model faces the greatest data scarcity.

Dataset	Class	Samples	%	IR
NSL-KDD	Normal	67,343	53.46	1.00
	DoS	45,927	36.46	1.47
	Probe	11,656	9.25	5.78
	R2L	995	0.79	67.68
	U2R	52	0.04	**1295.06**
UNSW-NB15	Normal	56,000	37.00	1.00
	Generic	40,000	26.43	1.40
	Exploits	33,393	22.07	1.68
	Fuzzers	18,184	12.02	3.08
	DoS	12,264	8.10	4.57
	Reconnaissance	10,491	6.93	5.34
	Analysis	2000	1.32	28.00
	Backdoor	1746	1.15	32.07
	Shellcode	1133	0.75	49.43
	Worms	130	0.09	**430.77**
CSE-CICIDS2018	Benign	869,858	83.07	1.00
	DDoS	576,191	55.01	1.51
	DoS	328,364	31.35	2.65
	BruteForce	190,654	18.21	4.56
	Bot	143,097	13.67	6.08
	Infilteration	93,063	8.89	9.35
	Web	596	0.06	**1459.49**
ToN-IoT	Normal	300,000	47.11	1.00
	Scanning	20,000	3.14	15.00
	DDoS	20,000	3.14	15.00
	DoS	20,000	3.14	15.00
	Injection	20,000	3.14	15.00
	XSS	20,000	3.14	15.00
	Password	20,000	3.14	15.00
	Backdoor	20,000	3.14	15.00
	Ransomware	20,000	3.14	15.00
	MITM	1043	0.16	**287.63**

As Table 4 shows, all four datasets exhibit severe class imbalance. The most extreme cases are U2R in NSL-KDD (IR = 1295), Web attacks in CSE-CICIDS2018 (IR = 1459), Worms in UNSW-NB15 (IR = 431), and MITM in ToN-IoT (IR = 288). These extreme imbalance ratios confirm that standard multiclass classification without explicit imbalance handling will inevitably result in poor detection of minority attack categories, further motivating the use of oversampling within our OvR binary decomposition framework.

3.4 Data Preprocessing

Prior to model training, a uniform preprocessing pipeline was applied across all four datasets. First, categorical features (e.g., protocol type, service, and flag fields in NSL-KDD; protocol and state in UNSW-NB15) were transformed into numerical representations using one-hot encoding or label encoding, depending on the cardinality of each feature. Second, all numerical features were standardized to zero mean and unit variance using standard scaling ($z = (x - \mu)/\sigma$), which is essential for distance-based algorithms such as KNN and SVM and for ensuring stable gradient-based optimization in MLP. Third, non-informative features (e.g., source/destination IP addresses and timestamps that serve as identifiers rather than discriminative attributes) were removed from all datasets, as these would introduce noise rather than meaningful classification signals.

No explicit feature selection or dimensionality reduction technique (e.g., PCA, mutual information ranking) was applied in this study. The rationale for this is that the proposed framework relies on the OvR binary decomposition, where each binary model independently learns the most relevant feature interactions for distinguishing its target class from the rest. This per-class specialization inherently allows each model to weight features differently during training. Furthermore, the hyperparameter tuning process via Grid Search with 5-Fold Cross-Validation implicitly optimizes the use of available features for each algorithm.

3.5 Algorithms

For the construction of the proposed multi-model classifier, we integrated intelligent classification algorithms that have demonstrated the best performance in both our experience and in previous scientific contributions [45]. These four algorithms were selected because they represent fundamentally different learning paradigms: instance-based (KNN), maximum-margin (SVM), rule-based (DT), and gradient-based neural (MLP) ensuring that the multi-model selection mechanism can exploit complementary inductive biases.

- K Nearest Neighbors Classifier (KNN). The KNN algorithm is an instance-based learning method, where classification is based on the training data [50]. It classifies a sample by calculating its distance from other samples and determines its class through a majority vote of its nearest k neighbors.
- Support Vector Machine (SVM). The SVM algorithm's core concept is to identify the hyperplane that optimally separates the dataset into distinct sets [51].
- Decision Trees (DT). Models based on DTs predict the class of a sample by applying straightforward decision rules learned during training [52].
- Multilayer Perceptron (MLP). An MLP is a feedforward artificial neural network consisting of an input layer, one or more hidden layers, and an output layer. In this study, we used the `MLPClassifier` implementation from Scikit-Learn with the default architecture of a single hidden layer containing 100 neurons. The Adam optimizer was employed with default momentum parameters ($\beta_1 = 0.9, \beta_2 = 0.999$), L2 regularization ($\alpha = 0.0001$), and a convergence tolerance of 10^{-4}. The activation function and learning rate were subject to hyperparameter tuning (Table 3).

Main software tools used for these experiments are Python version 3.12, with the assistance of libraries such as Pandas, Numpy, Scikit-Learn, and Imbalanced-learn, which are widely used in this research domain [36,53]. Hyperparameter tuning for each algorithm was performed using Grid Search and 5-Fold Cross-Validation methodologies [54], allowing the testing of various value combinations within a predefined search space. In terms of hardware, main specifications are Intel Core i9-10900 @5.20 GHz processor and 64 GB of RAM.

3.6 Evaluation Metrics

Evaluating classification performance under class imbalance requires careful metric selection, since standard accuracy can be misleading when the class distribution is heavily skewed [55,56]. In imbalanced intrusion detection datasets, a classifier that predicts the majority class for every sample can still achieve high accuracy while completely failing to detect minority attack categories. For this reason, our evaluation strategy prioritizes imbalance-aware metrics and treats accuracy only as a secondary, comparative reference.

3.6.1 Confusion-Matrix-Based Metrics

All per-class metrics are derived from the confusion matrix. For a given class c, the confusion matrix yields true positives (TP_c), true negatives (TN_c), false positives (FP_c), and false negatives (FN_c) [57]. Based on these quantities, the following metrics are computed:

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c} \tag{1}$$

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c} \tag{2}$$

$$\text{F1}_c = \frac{2 \cdot \text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c} \tag{3}$$

3.6.2 Justification of Metric Selection

The evaluation framework in this study is designed around two distinct levels:

- Binary model selection (Section 3.1). Macro-F1 is used as the optimization criterion during hyperparameter tuning via Grid Search with 5-Fold Cross-Validation. Since each binary dataset contains only two classes (target vs. rest), Macro-F1 ensures that both classes contribute equally to model selection, preventing the optimizer from favoring the majority class.
- Framework-level evaluation (Section 4). The final multiclass performance is assessed using Precision, Recall, F1-score, and Accuracy. Accuracy is reported solely for comparability with prior studies that use it as their primary metric; however, it is not the basis for any conclusion regarding the effectiveness of the proposed framework. Instead, the weighted F1-score is used as the principal indicator of overall classification quality, as it balances per-class performance against class prevalence.

We acknowledge that threshold-independent metrics such as ROC-AUC and PR-AUC [57,58] provide valuable complementary insights, particularly for assessing the discriminative ability of a classifier across all operating points. However, our framework produces hard class predictions through a maximum-probability decision rule applied over the outputs of multiple binary classifiers, rather than a single score that can be thresholded. Computing meaningful per-class ROC or PR curves would require access to the raw probability outputs of each binary model independently, which does not directly correspond to the final

multiclass decision. Furthermore, the comparative works in the IDS literature that we benchmark against predominantly report threshold-dependent metrics (Precision, Recall, F1, Accuracy), making these the most appropriate choice for a fair and direct comparison [45].

4 Experimental Results

Performance comparison of the proposed model is presented in this section using four widely used cybersecurity datasets: NSL-KDD, UNSW-NB15, CIC-IDS2018, and TON-IoT. The proposal is evaluated using well-known classification metrics and compared with other research proposed in the scientific literature.

The decision to test our binary model framework on multiple datasets was driven by several reasons. Firstly, it allows us to assess the robustness and generalizability of our framework across diverse attack scenarios and network environments. Secondly, it ensures a more realistic evaluation, considering variations in attack profiles and network conditions associated with different datasets. Thirdly, comparative analysis across multiple datasets helps identify our framework's strengths and weaknesses and enables us to compare its performance against existing methods. Lastly, the selection of widely recognized datasets aligns with community standards, facilitates better comparability with existing literature, and contributes to the advancement of intrusion detection research.

As a first step in testing the proposed model on each cybersecurity dataset, binary models were developed for each type of traffic. As explained in section 3.1, balanced datasets were generated for each type of cyberattack.

Each model is evaluated based on Macro-F1. Four binary classification models were generated using well-known algorithms for each type of traffic: three from Machine Learning (SVM, DT, KNN) and one from Deep Learning (MLP). Macro-F1 is not affected by the imbalance between classes. This metric was then used to select the most suitable algorithms for the proposed learning framework. Thus, both classes are considered equally relevant when calculating the binary model's performance.

The second step developed a multi-model classifier by combining the optimal models. Previously described test sets are used to evaluate the performance of the model. A confusion matrix is then generated, in which the rows correspond to the actual class labels, and the columns correspond to the predicted class labels.

As a final step toward validating the effectiveness of the proposal, an evaluation metric of accuracy, precision, recall, and F1-Score is used to compare our approach to four multiclass classification models based on a single algorithm and other current state-of-the-art works.

4.1 Experimentation and Discussion on NSL-KDD

First, the proposed model was evaluated using the NSL-KDD dataset. To accomplish this, our approach combines the best binary models for each of the five types of traffic collected: DoS, Probe, R2L, U2R, and Normal.

Table 5 depicts the Macro-F1 metric for the NSL-KDD dataset. Results are promising in DoS, Probe, R2L, and Normal categories, with Macro-F1 values of 0.9997, 0.9967, 0.9841, and 0.9581, respectively. A more difficult task is to distinguish network traces generated by attempts to gain access to a remote machine (U2R) from other traffic, resulting in a Macro-F1 of 0.8291. In NSL-KDD, the DT algorithm reached the best results for Probe, R2L, and U2R attacks. The highest Macro-F1 is achieved using SVM for DoS attacks, while the deep learning algorithm, MLP, produced the best detection model for distinguishing legitimate from attack traffic. The DT algorithm obtained the best results in three of the five types of traffic. This may be due to the fact that the dataset is not sufficiently sufficiently complex, and the features contain enough information to distinguish

between different types of traffic. In addition, this algorithm can achieve high classification metrics for both DoS and Normal traffic. This experiment revealed that the binary models generated with SVM for DoS, DT for Probe, R2L and U2R, and MLP for Normal traffic constitute the proposed two-step model.

Table 5: Binary models evaluation for NSL-KDD in terms of Macro-F1. For each traffic category, the bold value denotes the highest Macro-F1 score obtained among all compared models.

	DoS	Probe	R2L	U2R	Normal
SVM	**0.9997**	0.9951	0.9628	0.7719	0.9168
DT	0.9993	**0.9967**	**0.9841**	**0.8291**	0.9539
MLP	0.9995	0.9946	0.9639	0.7979	**0.9581**
KNN	0.9986	0.9930	0.9566	0.7163	0.9361

The confusion matrix was obtained after evaluating the test set to assess the performance of the proposed framework after combining the best binary algorithms for each category.

Table 6 presents the classification results in a confusion matrix to identify five types of network traffic. The proposed model performed well for DoS (80.1% recall), Probe (68.6%), and Normal (96.7%); however, R2L and U2R exhibited substantially lower detection rates. Specifically, 79.7% of R2L samples were misclassified as Normal, and only 3 out of 200 U2R samples were correctly identified (66.5% classified as Normal, 27.0% as DoS). These errors stem from two interrelated causes: (1) feature-space overlap: in R2L attacks mimic legitimate user behavior (e.g., password guessing, FTP exploits) producing flow-level features nearly indistinguishable from Normal traffic, while U2R attacks exploit host-level privilege escalation with minimal network signatures and (2) severe class imbalance, as U2R comprises only 52 training samples (IR = 1295) and R2L 995 samples (IR = 67), providing the binary models with extremely limited positive examples to learn discriminative boundaries. These categories are widely recognized as the most challenging in the NSL-KDD dataset across the IDS literature, including deep learning-based approaches [14,20].

Table 6: Confusion matrix for binary model learning framework on NSL-KDD dataset.

	DoS	Probe	R2L	U2R	Normal
DoS	5971	13	227	0	1247
Probe	360	1660	38	0	363
R2L	399	2	155	2	2196
U2R	54	3	7	3	133
Normal	121	194	6	1	9389

Table 7 provides a performance comparison among several state-of-the-art research. Our proposal achieves an F1 of 0.7213, higher than the best F1 score of 0.7126 obtained by another scientific work in intrusion detection using the NSL-KDD dataset. This suggests that our model can achieve better performance in terms of the balance between precision and recall compared to the other works. This is an important finding, as it indicates that the proposed work can identify intrusions while also minimizing the number of false positives.

Table 7: Comparison of performance against state-of-the-art works for NSL-KDD. For each evaluation metric (Accuracy, Precision, Recall, and F1-score), the bold value denotes the highest score obtained among all compared methods.

Work	Accuracy	Precision	Recall	F1
[20]	**0.7660**	–	–	–
[21]	0.7608	0.7360	0.7608	0.7126
SVM	0.7658	0.7973	**0.7658**	0.7157
DT	0.7615	0.7957	0.7615	0.7196
MLP	0.7507	**0.8006**	0.7507	0.7070
KNN	0.7499	0.7955	0.7499	0.7062
Proposed	0.7620	0.7329	0.7620	**0.7213**

4.2 Experimentation and Discussion on UNSW-NB15

The UNSW-NB15 set was used to validate the proposal for the second group of experiments. Consequently, the model should identify ten types of traffic: Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Normal, Reconnaissance, Shellcode, and Worms.

In accordance with the previously described procedure, the first step was to generate classification models for each of the nine types of attacks collected. A model for detecting legitimate traffic was developed as well, resulting in ten models. Table 8 contains Macro-F1 values obtained for each of the four algorithms for each type of traffic. Generic was the best-detected attack traffic with a value greater than 0.9835. The performance of this model is likely due to the extensive dataset utilized for its training. The recall rate for identifying Reconnaissance and Shellcode attacks surpassed 0.9. For overall attack detection, irrespective of the attack type, the proposed classifier achieved a slightly higher score than 0.91. DoS attacks, Exploits, and Fuzzers presented the lowest Macro-F1 scores, each around 0.8. Nevertheless, the traffic detection and classification system demonstrated high accuracy, often exceeding 0.85 in most categories, occasionally reaching 0.9, and achieving outstanding results in certain instances.

Table 8: Binary models evaluation for UNSW-NB15 in terms of Macro-F1. For each traffic category, the bold value denotes the highest Macro-F1 score obtained among all compared models.

	Analys.	Backdr.	DoS	Expl.	Fuzz.	Generic	Normal	Recon.	Shell.	Worms
SVM	0.7934	0.8026	0.7723	0.7699	**0.8128**	0.9792	0.8304	0.8845	0.9297	0.8764
DT	**0.8298**	**0.8530**	0.7457	**0.7962**	0.7967	**0.9835**	**0.9112**	**0.9183**	**0.9643**	**0.8876**
MLP	0.8228	0.7902	**0.7909**	0.7963	0.8002	0.9813	0.8714	0.9136	0.9324	0.8876
KNN	0.7901	0.7891	0.7007	0.7414	0.8004	0.9781	0.7999	0.8377	0.8949	0.8764

Table 9 displays the confusion matrix generated by the two-phase binary classifier model. Analyzing the main diagonal of the matrix, the model demonstrates high performance for categories such as Exploits, Generic, Normal, and Reconnaissance, with a substantial number of samples accurately classified.

Finally, it is valuable to compare the performance of the proposed system with other methods discussed earlier in the Section 2. Table 10 presents a comparative evaluation of our system against various approaches. The proposed framework achieves the highest F1-score (0.7754), outperforming both Ref. [19] (F1 = 0.7400) and all four individual classifiers. While Ref. [19] reports a higher precision (0.8400), this comes at the cost of substantially lower recall (0.6900), indicating a conservative model that misses a significant proportion of attacks. Our framework achieves a more balanced trade-off between precision (0.8271) and recall (0.7604),

which is preferable in IDS scenarios where failing to detect attacks carries higher risk than occasional false alarms. Ref. [17] reports only accuracy (0.7583) without other metrics, limiting a comprehensive comparison; nonetheless, the proposed framework surpasses it in accuracy (0.7605).

Table 9: Confusion matrix for binary model learning framework on UNSW-NB15 dataset.

	Analys.	Backdr.	DoS	Expl.	Fuzz.	Generic	Normal	Recon.	Shell.	Worms
Analysis	350	230	27	35	5	0	30	0	0	0
Backdoor	351	184	6	3	2	0	32	1	2	2
DoS	1533	1075	327	538	74	15	271	41	125	90
Exploits	2173	1278	299	5243	217	52	791	187	421	471
Fuzzers	855	437	30	192	1115	19	2693	5	658	58
Generic	80	27	26	223	62	18250	77	3	106	17
Normal	132	40	225	475	827	6	34384	7	810	94
Recon.	234	122	34	42	9	1	97	2392	356	209
Shellcode	6	1	0	2	3	0	17	0	348	1
Worms	1	0	1	10	0	3	2	1	2	24

Table 10: Comparison of performance against state-of-the-art works for UNSW-NB15. For each evaluation metric (Accuracy, Precision, Recall, and F1-score), the bold value denotes the highest score obtained among all compared methods.

Work	Accuracy	Precision	Recall	F1
[19]	0.6946	**0.8400**	0.6900	0.7400
[17]	0.7583	–	–	–
SVM	0.7051	0.8157	0.7051	0.7130
DT	0.7345	0.8087	0.7345	0.7626
MLP	0.7558	0.8109	0.7558	0.7581
KNN	0.7067	0.7900	0.7067	0.7374
Proposed	**0.7605**	0.8271	**0.7604**	**0.7754**

4.3 Experimentation and Discussion on CIC-IDS2018

The third set of experiments was carried out on the CIC-IDS2018 dataset. The two-phase model developed here was designed to distinguish between legitimate traffic and the following attacks: Bot, Brute-force, DDoS, DoS, Infilteration, and Web-based attacks. Table 11 shows the Macro-F1 metrics obtained for the binary models of each algorithm, for each type of traffic. The highest classification results are marked in bold for each attack. It can be seen that the binary models selected for this set present good performance, obtaining Macro-F1 values above 0.90 for six of the seven traffic categories.

After selecting the best-performing binary model, the proposed system was tested against an unseen test set. The confusion matrix was used to evaluate the performance of the developed multi-model classifier. Table 12 shows the number of predicted samples for each model in seven categories.

To demonstrate the effectiveness of the proposal, Table 13 shows a comparison among our work and other research works reviewed in the state of the art that have used CSE-CICIDS2018. It is found that the research in Ref. [22] produces slightly lower results than our proposal achieving 0.9310 and 0.9222 in accuracy and F1, respectively. Ref. [16] obtains a higher accuracy and recall than the proposed one, although the F1 is lower than that obtained. These results may be due to the use of a dataset with only NetFlow-based features.

Table 11: Binary models evaluation for CIC-IDS2018 in terms of Macro-F1. For each traffic category, the bold value denotes the highest Macro-F1 score obtained among all compared models.

	Benign	Bot	BruteF.	DDoS	DoS	Infil.	Web
SVM	0.9256	**0.9998**	**0.9387**	0.9989	0.9550	0.6832	0.8810
DT	**0.9410**	0.9998	0.9383	0.9992	**0.9587**	**0.7151**	**0.9012**
MLP	0.9266	0.9996	0.9383	0.9993	0.9573	0.6711	0.7704
KNN	0.9170	0.9998	0.9376	**0.9995**	0.9577	0.6953	0.8301

Table 12: Confusion matrix for binary model learning framework on CSE-CICIDS2018 dataset.

	Benign	Bot	BruteF.	DDoS	DoS	Infil.	Web
Benign	260,536	3965	9	3	3	5120	2
Bot	10	57,341	0	3	0	38	2
BruteF.	0	4	71,861	0	4404	0	0
DDoS	20	246	0	252,690	0	0	0
DoS	17	47	13,659	0	11,6712	0	0
Infil.	21,261	937	2	4	1	10,248	1
Web	9	51	0	1	0	1	133

Table 13: Comparison of performance against state-of-the-art works for CIC-IDS2018. For each evaluation metric (Accuracy, Precision, Recall, and F1-score), the bold value denotes the highest score obtained among all compared methods.

Work	Accuracy	Precision	Recall	F1
[16]	**0.9705**	–	**0.9467**	0.9000
[22]	0.9310	–	–	0.9222
SVM	0.9392	0.9386	0.9392	0.9304
DT	0.9303	0.9314	0.9303	0.9304
MLP	0.9416	**0.9404**	0.9416	0.9331
KNN	0.9365	0.9333	0.9365	0.9298
Proposed	0.9392	0.9353	0.9392	**0.9340**

4.4 Experimentation and Discussion on ToN-IoT

The final set of experiments involved testing the proposed model on the ToN-IoT set. It was necessary for the proposal to distinguish different categories of traffic in a network of IoT devices: Backdoor, DDoS, DoS, Injection, MITM, Normal, Password, Ransomware, Scanning, and XSS.

As in the previous experiments, the first step was the generation of binary models. Table 14 shows the Macro-F1 score associated with each category for the four algorithms considered in this experiment. There was an evaluation of greater than 0.9 for all models, except traffic from MITM attacks, which achieved an evaluation of 0.8329.

Based on the analysis of the performance of each algorithm individually, SVM was the algorithm that produced the best results when it came to identifying DDoS attacks. The rest of the cyberattacks were easier to detect from the rest using the DT algorithm, achieving 0.9990, 0.9915, 0.9819, 0.8329, 0.9894, 0.9398, 0.9945,

0.9581 for Backdoor, DoS, Injection, MITM, Password, Ransomware, Scanning, XSS, respectively. The binary model selected for discerning between legitimate and suspicious traffic was obtained with MLP, achieving a Macro-F1 of 0.9326. The MLP algorithm was not used in the model generation for any traffic since it would imply a notable decrease in several categories in Macro-F1.

Table 14: Binary models evaluation for ToN-IoT in terms of Macro-F1. For each traffic category, the bold value denotes the highest Macro-F1 score obtained among all compared models.

	Backdr.	DDoS	DoS	Inject.	MITM	Normal	Passwd.	Ransom.	Scan.	XSS
SVM	0.7278	**0.9863**	0.8195	0.7924	0.7391	0.9037	0.6733	0.9144	0.8914	0.8967
DT	**0.9990**	0.9754	**0.9915**	**0.9819**	**0.8329**	0.8903	**0.9894**	**0.9398**	**0.9945**	**0.9581**
MLP	0.6022	0.9708	0.6380	0.9624	0.5515	**0.9326**	0.8413	0.8990	0.8654	0.8712
KNN	0.6165	0.9687	0.7204	0.9422	0.7328	0.8767	0.7961	0.8912	0.8532	0.8278

There was an interesting observation about the performance of the DT algorithm compared to SVM, MLP, and KNN for a significant portion of the traffic in this dataset. A well-performing DT algorithm indicates that the data is not very complex. Complexity will be determined by the methods used to collect network traffic, the tools used to generate features, or the methodologies used to generate attacks.

The classifier was constructed in two steps after evaluation of the individual models and selection of the most effective ones. In order to validate the proposal, we used the test set, and in order to evaluate it, we calculated classification metrics based on the confusion matrix.

Table 15 displays the confusion matrix for evaluation on the ToN-IoT dataset. According to the results, the performance of the classification is excellent. As a result, most of the classes were able to detect the traffic they were trained for and the number of classification failures was low. Backdoor is a good example of this. There were 3948 test samples in this category, of which 3946 were correctly classified, and two were incorrectly classified as Normal.

Table 15: Confusion matrix for binary model learning framework on TON-IoT dataset.

	Backdr.	DDoS	DoS	Inject.	MITM	Normal	Passwd.	Ransom.	Scan.	XSS
Backdoor	3946	0	0	0	0	2	0	0	0	0
DDoS	0	3885	4	25	2	28	7	44	3	17
DoS	1	0	3944	3	24	15	9	1	38	0
Injection	0	27	5	3842	7	20	2	6	21	101
MITM	0	1	14	6	162	22	8	0	4	2
Normal	1	29	10	17	17	59,097	56	668	8	70
Password	0	4	7	1	16	7	3814	34	1	18
Ransom.	0	0	0	0	0	55	0	3931	0	81
Scanning	0	0	9	4	9	7	3	2	4051	0
XSS	0	21	1	45	1	16	5	261	5	3579

A comparison is made between the results obtained in this research and previous contributions to the detection of intrusions on ToN-IoT in Table 16. The proposed framework achieves the highest scores across all four metrics (accuracy: 0.9788, precision: 0.9807, recall: 0.9788, F1: 0.9793), outperforming both Ref. [16] (F1 = 0.9400) and Ref. [11] (F1 = 0.9660), which employ NetFlow-based features and evolutionary cost-sensitive deep learning, respectively. The improvement over the best individual classifier (DT, F1 = 0.9714)

confirms that the two-step binary framework provides a consistent gain beyond what any single algorithm achieves. Notably, the margin over [11] is particularly significant given that their approach uses a more complex deep learning architecture, supporting the thesis that the structural design of the proposed OvR framework is the primary driver of performance.

Table 16: Comparison of performance against state-of-the-art works for TON-IoT. For each evaluation metric (Accuracy, Precision, Recall, and F1-score), the bold value denotes the highest score obtained among all compared methods.

Work	Accuracy	Precision	Recall	F1
[16]	0.9380	–	0.9229	0.9400
[11]	–	0.9730	0.9610	0.9660
SVM	0.7604	0.7747	0.7604	0.7013
DT	0.9711	0.9727	0.9711	0.9714
MLP	0.8922	0.8880	0.8922	0.8878
KNN	0.8649	0.8871	0.8649	0.8624
Proposed	**0.9788**	**0.9807**	**0.9788**	**0.9793**

4.5 Limitations

The proposed framework is evaluated on static, offline benchmark datasets and does not address concept drift the phenomenon whereby the statistical properties of network traffic and attack patterns change over time, causing a trained model's decision boundaries to become progressively misaligned with the evolving data distribution [59]. In real-world deployment, new attack variants, shifts in legitimate usage patterns, and adversarial adaptation can all degrade the performance of a fixed model. However, the modular architecture of the proposed framework offers a structurally favorable basis for drift adaptation: since each binary classifier operates independently within the OvR decomposition, individual models can be selectively retrained or replaced when drift is detected in a specific attack category, without requiring a complete retraining of the entire system. Integrating drift detection mechanisms together with incremental or online learning strategies for the underlying classifiers, constitutes an important direction for future research to enable the deployment of the proposed framework in non-stationary network environments.

Closely related to concept drift is the challenge of real-time, high-throughput deployment. The current evaluation is performed offline on stored benchmark datasets and does not measure end-to-end latency under live traffic conditions. Nevertheless, the framework exhibits several properties that are favorable for real-time operation. First, as shown in the complexity analysis (Computational Complexity Section), the inference cost for a single sample is bounded by $\mathcal{O}(C_1 + \max_i\{C_i\})$ when the second-step binary classifiers are executed in parallel, which is comparable to a single multiclass model. Second, the classifiers employed (KNN, SVM, DT, MLP) are lightweight in comparison with deep learning architectures, yielding low per-sample prediction latencies that are well within the requirements of network-level monitoring. Third, the two-step architecture provides an inherent early-exit mechanism: traffic classified as Normal in the first step bypasses the second-step classification entirely, reducing the average inference workload in networks where legitimate traffic constitutes the vast majority of flows. Formal benchmarking of throughput and latency under realistic network conditions, including integration with packet capture and flow extraction pipelines, is identified as an important direction for future work.

5 Conclusions

This paper proposes a novel binary model learning framework that ensures cybersecurity in interconnected environments by detecting and classifying cyberattacks through the study of network traffic traces. This proposal also deals with class imbalance by oversampling minority classes during binary model generation.

This method has been tested on four well-known intrusion detection datasets: NSL-KDD, UNSW-NB15, CSE-CIC-IDS2018 and TON-IoT. In all scenarios, the proposed model shows a gain in F1, when compared to other proposals in the scientific literature, resulting in a combination of precision and recall. Thus, from experiments, our model effectively classifies network traffic in imbalance scenarios.

As future work, investigating the use of new oversampling methods in combination with proposed binary model framework could be an interesting avenue. Furthermore, replacing or augmenting the current classical classifiers (KNN, SVM, DT, MLP) with more complex algorithms such as XGBoost, LightGBM or deep learning architectures such as CNN-LSTM hybrids, attention-based models, or transformer networks as internal components of the proposed OvR binary framework represents a particularly promising direction.

Acknowledgement: The authors would like to acknowledge the support provided by University of Extremadura, which greatly facilitated this research

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Óscar Mogollón-Gutiérrez, José Carlos Sancho Núñez and MohammadHossein Homaei; methodology, Óscar Mogollón-Gutiérrez; software, Óscar Mogollón-Gutiérrez; validation, Óscar Mogollón-Gutiérrez, José Carlos Sancho Núúñez and Mar Ávila; formal analysis, Óscar Mogollón-Gutiérrez; investigation, Óscar Mogollón-Gutiérrez and MohammadHossein Homaei; resources, Andrés Caro; data curation, Óscar Mogollón-Gutiérrez; writing original draft preparation, Óscar Mogollón-Gutiérrez; writing review and editing, José Carlos Sancho Núñez, Mar Ávila, MohammadHossein Homaei and Andrés Caro; visualization, Óscar Mogollón-Gutiérrez; supervision, Andrés Caro; project administration, Andrés Caro; funding acquisition, Andrés Caro. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The source code of the proposed binary model framework is publicly available at https://github.com/Grupo-de-Ingenieria-de-Medios-GIM/BinaryModelFramework. The four datasets used in this study are publicly available from their original sources: NSL-KDD [46], UNSW-NB15 [47], CSE-CICIDS2018 [48], and ToN-IoT [49].

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Sancho Nuñez JC, Caro Lindo A, Garcia Rodriguez P. A preventive secure software development model for a software factory: a case study. IEEE Access. 2020;8:77653–65. doi:10.1109/ACCESS.2020.2989113.
2. Buczak AL, Guven E. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun Surv Tutor. 2016;18(2):1153–76. doi:10.1109/COMST.2015.2494502.
3. Farooq MS, Abbas S, ur Rahman A, Sultan K, Khan MA, Mosavi A. A fused machine learning approach for intrusion detection system. Comput Mater Contin. 2023;74(2):2607–23. doi:10.32604/cmc.2023.032617.
4. Gebrye H, Wang Y, Li F. Traffic data extraction and labeling for machine learning based attack detection in IoT networks. Int J Mach Learn Cybern. 2023;14(7):2317–32. doi:10.1007/s13042-022-01765-7.
5. Horchulhack P, Viegas EK, Santin AO. Toward feasible machine learning model updates in network-based intrusion detection. Comput Netw. 2022;202(4):108618. doi:10.1016/j.comnet.2021.108618.

6. Jain DK, Ding W, Kotecha K. Training fuzzy deep neural network with honey badger algorithm for intrusion detection in cloud environment. Int J Mach Learn Cybern. 2023;14(6):2221–37. doi:10.1007/s13042-022-01758-6.
7. Kamalakkannan D, Menaga D, Shobana S, Sagar KVD, Rajagopal R, Tiwari M. A detection of intrusions based on deep learning. Cybern Syst. 2023;56(5):511–25. doi:10.1080/01969722.2023.2175134.
8. Jeon S, Kim HK. AutoVAS: an automated vulnerability analysis system with a deep learning approach. Comput Secur. 2021;106(4):102308. doi:10.1016/j.cose.2021.102308.
9. Sharma B, Sharma L, Lal C, Roy S. Anomaly based network intrusion detection for IoT attacks using deep learning technique. Comput Electr Eng. 2023;107(1):108626. doi:10.1016/j.compeleceng.2023.108626.
10. Ding H, Chen L, Dong L, Fu Z, Cui X. Imbalanced data classification: a KNN and generative adversarial networks-based hybrid approach for intrusion detection. Future Gener Comput Syst. 2022;131(7):240–54. doi:10.1016/j.future.2022.01.026.
11. Telikani A, Shen J, Yang J, Wang P. Industrial IoT intrusion detection via evolutionary cost-sensitive learning and fog computing. IEEE Internet Things J. 2022;9(22):23260–71. doi:10.1109/jiot.2022.3188224.
12. Galar M, Fernandez A, Barrenechea E, Bustince H, Herrera F. A Review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans Syst Man Cybern Part C. 2012;42(4):463–84. doi:10.1109/tsmcc.2011.2161285.
13. Tama BA, Comuzzi M, Rhee KH. TSE-IDS: a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. IEEE Access. 2019;7:94497–507. doi:10.1109/access.2019.2928048.
14. Khan FA, Gumaei A, Derhab A, Hussain A. TSDL: a two-stage deep learning model for efficient network intrusion detection. IEEE Access. 2019;7(c):30373–85. doi:10.1109/ACCESS.2019.2899721.
15. de Souza CA, Westphall CB, Machado RB, Sobral JBM, dos Santos Vieira G. Hybrid approach to intrusion detection in fog-based IoT environments. Comput Netw. 2020;180(7):107417. doi:10.1016/j.comnet.2020.107417.
16. Sarhan M, Layeghy S, Portmann M. Evaluating standard feature sets towards increased generalisability and explainability of ML-based network intrusion detection. Big Data Res. 2022;30(3):100359. doi:10.1016/j.bdr.2022.100359.
17. Kasongo SM, Sun Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. Comput Secur. 2020;92(1):101752. doi:10.1016/j.cose.2020.101752.
18. Khammassi C, Krichen S. A GA-LR wrapper approach for feature selection in network intrusion detection. Comput Secur. 2017;70(2):255–77. doi:10.1016/j.cose.2017.06.005.
19. Al-Turaiki I, Altwaijry N. A convolutional neural network for improved anomaly-based network intrusion detection. Big Data. 2021;9(3):233–52. doi:10.1089/big.2020.0263.
20. Andresini G, Appice A, Malerba D. Autoencoder-based deep metric learning for network intrusion detection. Inf Sci. 2021;569:706–27. doi:10.1016/j.ins.2021.05.016.
21. Wang W, Du X, Shan D, Qin R, Wang N. Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine. IEEE Trans Cloud Comput. 2022;10(3):1634–46. doi:10.1109/tcc.2020.3001017.
22. Ren K, Zeng Y, Cao Z, Zhang Y. ID-RDRL: a deep reinforcement learning-based feature selection intrusion detection model. Sci Rep. 2022;12(1):15370. doi:10.1038/s41598-022-19366-3.
23. Mighan SN, Kahani M. A novel scalable intrusion detection system based on deep learning. Int J Inf Secur. 2020;20(3):387–403. doi:10.1007/s10207-020-00508-5.
24. Devi M, Nandal P, Sehrawat H. Federated learning-enabled lightweight intrusion detection system for wireless sensor networks: a cybersecurity approach against DDoS attacks in smart city environments. Intell Syst Appl. 2025;27(5):200553. doi:10.1016/j.iswa.2025.200553.
25. Ullah F, Ullah S, Srivastava G, Lin JCW. IDS-INT: intrusion detection system using transformer-based transfer learning for imbalanced network traffic. Digit Commun Netw. 2024;10(1):190–204.
26. Arafah M, Phillips I, Adnane A, Hadi W, Alauthman M, Al-Banna AK. Anomaly-based network intrusion detection using denoising autoencoder and Wasserstein GAN synthetic attacks. Appl Soft Comput. 2025;168(8):112443. doi:10.1016/j.asoc.2024.112455.

27. Elijah AV, Abdullah A, JhanJhi NZ, Supramaniam M, Balogun Abdullateef O. Ensemble and deep-learning methods for two-class and multi-attack anomaly intrusion detection: an empirical study. Int J Adv Comput Sci Appl. 2019;10(9):520–8. doi:10.14569/ijacsa.2019.0100969.
28. Maniriho P, Mahoro LJ, Niyigaba E, Bizimana Z, Ahmad T. Detecting intrusions in computer network traffic with machine learning approaches. Int J Intell Eng Syst. 2020;13(3):433–45. doi:10.22266/IJIES2020.0630.39.
29. Talukder MA, Islam MM, Uddin MA, Hasan KF, Sharmin S, Alyami SA, et al. Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. J King Saud Univ Comput Inf Sci. 2024;36(1):101880.
30. Meftah S, Rachidi T, Assem N. Network based intrusion detection using the UNSW-NB15 dataset. Int J Comput Digit Syst. 2019;8(5):477–87. doi:10.12785/ijcds/080505.
31. Chen Z, Simsek M, Kantarci B, Bagheri M, Djukic P. Machine learning-enabled hybrid intrusion detection system with host data transformation and an advanced two-stage classifier. Comput Netw. 2024;250(1):110549. doi:10.1016/j.comnet.2024.110576.
32. Mahmoud MM, Youssef YO, Abdel-Hamid AA. XI2S-IDS: an explainable intelligent 2-Stage intrusion detection system. Future Internet. 2025;17(1):14.
33. Sanchez RAM, Zaman M, Goel N, Naik K, Joshi R. Towards developing a robust intrusion detection model using hadoop spark and data augmentation for IoT networks. Sensors. 2022;22(20):7726. doi:10.3390/s22207726.
34. Kil YS, Jeon YR, Lee SJ, Lee IG. Multi-binary classifiers using optimal feature selection for memory-saving intrusion detection systems. Comput Model Eng Sci. 2024;141(2):1591–611. doi:10.32604/cmes.2024.052637.
35. Divekar A, Parekh M, Savla V, Mishra R, Shirole M. Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives. In: Proceedings of the 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS); 2018 Oct 25–27; Kathmandu, Nepal. doi:10.1109/cccs.2018.8586840.
36. Bagui S, Li K. Resampling imbalanced data for network intrusion detection datasets. J Big Data. 2021;8(1):6. doi:10.1186/s40537-020-00390-x.
37. Al-Shehari T, Alsowail RA. Random resampling algorithms for addressing the imbalanced dataset classes in insider threat detection. Int J Inf Secur. 2022;22(3):611–29. doi:10.1007/s10207-022-00651-1.
38. Aziz MN, Ahmad T. Clustering under-sampling data for improving the performance of intrusion detection system. J Eng Sci Technol. 2021;16(2):1342–55. doi:10.21533/pen.v8.i2.1096.
39. Le TTH, Shin Y, Kim M, Kim H. Towards unbalanced multiclass intrusion detection with hybrid sampling methods and ensemble classification. Comput Secur. 2024;138(2):103649. doi:10.1016/j.asoc.2024.111517.
40. Rani M, Gagandeep. Effective network intrusion detection by addressing class imbalance with deep neural networks multimedia tools and applications. Multimed Tools Appl. 2022;81(6):8499–518. doi:10.1007/s11042-021-11747-6.
41. Zhang H, Huang L, Wu CQ, Li Z. An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset. Comput Netw. 2020;177:107315. doi:10.1016/j.comnet.2020.107315.
42. Omer Albasheer F, Ramesh Haibatti R, Agarwal M, Yeob Nam S. A novel IDS based on jaya optimizer and smote-ENN for cyberattacks detection. IEEE Access. 2024;12(2):15342–58. doi:10.1109/access.2024.3431534.
43. Mogollon-Gutierrez O, Sancho Nuñez JC, Avila Vegas M, Caro Lindo A. A novel ensemble learning system for cyberattack classification. Intell Autom Soft Comput. 2023;37(2):1691–709.
44. Sarhan M, Layeghy S, Moustafa N, Gallagher M, Portmann M. Feature extraction for machine learning-based intrusion detection in IoT networks. Digit Commun Netw. 2024;10(1):205–16. doi:10.1016/j.dcan.2022.08.012.
45. Thakkar A, Lohiya R. A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges. Arch Comput Methods Eng. 2020;28(4):3211–43. doi:10.1007/s11831-020-09496-0.
46. Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA); 2009 Jul 8–10; Ottawa, ON, Canada. p. 1–6.

47. Moustafa N, Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Proceedings of the 2015 Military Communications and Information Systems Conference, MilCIS 2015; 2015 Nov 10–12; Canberra, ACT, Australia. doi:10.1109/MilCIS.2015.7348942.
48. Sharafaldin I, Lashkari AH, Ghorbani AA. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy; 2018 Jan 22–24; Funchal, Portugal. doi:10.5220/0006639801080116.
49. Moustafa N. A new distributed architecture for evaluating AI-based security systems at the edge: network TON_IoT datasets. Sustain Cities Soc. 2021;72:102994. doi:10.1016/j.scs.2021.102994.
50. Guo G, Wang H, Bell D, Bi Y, Greer K. KNN model-based approach in classification. Lect Notes Comput Sci. 2003;2888:986–96. doi:10.1007/978-3-540-39964-3.
51. Hearst MA. SVMs—a practical consequence of learning theory. IEEE Intell Syst Their Appl. 1998;13(4):18–21. doi:10.1109/5254.708428.
52. Quinlan JR. Induction of decision trees. Mach Learn. 1986;1(1):81–106. doi:10.1007/bf00116251.
53. Yang Y, Zheng K, Wu C, Yang Y. Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. Sensors. 2019;19(11):2528. doi:10.3390/s19112528.
54. Baig MM, Awais MM, El-Alfy ESM. A multiclass cascade of artificial neural network for network intrusion detection. J Intell Fuzzy Syst. 2017;32(4):2875–83. doi:10.3233/JIFS-169230.
55. He H, Garcia EA. Learning from imbalanced data. IEEE Trans Knowl Data Eng. 2009;21(9):1263–84. doi:10.1109/tkde.2008.239.
56. Sokolova M, Lapalme G. A systematic analysis of performance measures for classification tasks. Inf Process Manag. 2009;45(4):427–37. doi:10.1016/j.ipm.2009.03.002.
57. Fawcett T. An introduction to ROC analysis. Pattern Recognit Lett. 2006;27(8):861–74. doi:10.1016/j.patrec.2005.10.010.
58. Davis J, Goadrich M. The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning (ICML); 2006 Jun 25–29; Pittsburgh, PA, USA. p. 233–40.
59. Gama J, Žliobaitė I, Bifet A, Pechenizkiy M, Bouchachia A. A survey on concept drift adaptation. ACM Comput Surv. 2014;46(4):44:1–37. doi:10.1145/2523813.