IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

IEEE SYSTEMS, MAN AND CYBERNETICS SOCIETY SECTION

Received 6 May 2024, accepted 27 May 2024, date of publication 29 May 2024, date of current version 20 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3407029

RESEARCH ARTICLE

A Two-Level Ensemble Learning Framework for Enhancing Network Intrusion Detection Systems

OSVALDO ARRECHE[1], ISMAIL BIBERS[2], AND MUSTAFA ABDALLAH[2], (Member, IEEE)

[1]Electrical and Computer Engineering Department, Purdue School of Engineering and Technology, Indiana University–Purdue University Indianapolis (IUPUI), Indianapolis, IN 46202, USA

[2]Computer and Information Technology Department, Purdue School of Engineering and Technology, Indiana University–Purdue University Indianapolis (IUPUI), Indianapolis, IN 46202, USA

Corresponding author: Mustafa Abdallah (mabdall@iu.edu)

This work was supported in part by Lilly Endowment under Grant AnalytixIN, in part by the Enhanced Mentoring Program with Opportunities for Ways to Excel in Research (EMPOWER), and in part by the 1st Year Research Immersion Program (1RIP) grants from the Office of the Vice Chancellor for Research at Indiana University–Purdue University Indianapolis.

ABSTRACT The exponential growth of intrusions on networked systems inspires new research directions on developing artificial intelligence (AI) techniques for intrusion detection systems (IDS). In this context, several AI techniques have been leveraged for automating network intrusion detection tasks. However, each AI model has unique strengths points and weaknesses, and one may be better than the other depending on the dataset, which might aggravate which model to choose. Thus, combining these AI models can improve their use of generalization and application in network intrusion detection tasks. In this paper, we aim to fill such a gap by evaluating diverse ensemble methods for network intrusion detection systems. In particular, we build a two-level ensemble learning framework for evaluating such ensemble learning methods in network intrusion detection tasks. In the first level of our framework, we load the input dataset, train the base learners and ensemble methods, and generate the evaluation metrics. This level also produces new datasets (needed to train the second level) based on both prediction probabilities of base and ensemble models used in the first level. The second level of the framework consists of loading the datasets generated from the first level, training the ensemble methods, and generating the evaluation metrics. Our framework also considers feature selection for both levels. In particular, we perform XAI-based feature selection in the first level and Information Gain-based feature selection in the second level. We present results for several ensemble model combinations in our two-level framework (i.e., 24 methods), including different bagging, stacking, and boosting methods on several base learners (e.g., decision trees, support vector machines, deep neural networks, and others). We evaluate our framework on three network intrusion datasets with different characteristics (RoEduNet-SIMARGL2021, NSL-KDD, and CICIDS-2017). We also categorize AI models according to their performances on our evaluation metrics. Our evaluation shows that it is beneficial to perform two-level learning for most setups considered in this work. We also release our source codes for the community to access as a baseline two-level ensemble learning framework for network intrusion detection.

INDEX TERMS Intrusion detection systems, ensemble learning, network security, two-level learning, feature selection, machine learning, NSL-KDD, CICIDS-2017, RoEduNet-SIMARGL2021.

I. INTRODUCTION

Intrusion Detection Systems (IDS) aim to detect unauthorized use, misuse, and abuse of computer networks by insiders and external attackers [1], [2], [3]. Traditional IDS rely on the assumption that intruders' behavior will markedly differ from

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaojie Su.

legitimate users, making unauthorized actions detectable. The advancement of Artificial Intelligence (AI) has spurred the development of fully automated IDS [4], [5], utilizing AI techniques such as neural networks [6], [7], support vector machines [8], [9], decision trees [10], [11], naive bayes [12], [13], and random forest [14], [15].

However, most AI methods, except decision forest, are base learning models that do not combine decisions for IDS management [16], [17]. These models often suffer from high false positive rates, with large companies receiving thousands of security alerts daily [18], and high false negative rates, which pose risks in safety-critical network applications [19]. Previous studies typically focused on the classification accuracy of individual AI algorithms without harnessing the collective strength of these diverse techniques. This has highlighted the urgent need to utilize ensemble learning methods to enhance IDS effectiveness [20], [21], [22].

To address this aforementioned issue of base learners, recent studies have explored ensemble learning across various AI models for IDS [20], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35]. Particularly, some focused on anomaly detection [25], [26], [27], [29], [30], [32], [33], classifying normal and anomalous traffic, while others develop multiclass frameworks for various network intrusions [23], [24], [28], [31], [34], [35]. These works utilize methods such as Boosting, Stacking, and Bagging with base models including Decision Trees, SVMs, and Neural Networks, while using accuracy, precision, recall, F1, and false positive rates to assess its quality. Furthermore, some works used benchmark datasets (e.g., NSL-KDD) [25], [29], [30], [34], and others test on real networks (e.g., Palo Alto) [32] and in real-time [26]. However, there is a lack of comprehensive evaluation across a broad range of AI methods and datasets, limiting the generalizability of these studies.

This paper aims to fill such a gap in evaluating diverse ensemble methods for network intrusion detection systems. We build a two-level (Level 00 and Level 01) ensemble learning framework for evaluating such ensemble learning methods in network intrusion detection based on the prior works [20], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35] that delineate several ensemble learning approaches. Our work distinguishes itself by integrating prediction probabilities and predicted classes into the ensemble learning methods at Level 01 of our framework, inspired by previous research [24]. We also conduct feature selection at both levels: XAI-based at Level 00 [36] and using Information Gain at Level 01 [37], assessing the efficacy compared to using all features.

We evaluate our framework on three network intrusion datasets with different characteristics. The first dataset is the recent RoEduNet-SIMARGL2021 dataset [38] collected from the SIMARGL project (supported by the European Union).[1] It contains realistic network traffic with features from live traffic, which makes the dataset highly usable for network intrusion detection systems. The second dataset is CICIDS-2017 [39], a benchmark intrusion detection dataset created by the Canadian Institute for Cybersecurity at the University of Brunswick in 2017 with different attack profiles. The final one is the widely recognized NSL-KDD [40] benchmark dataset.

Our evaluation indicates that two-level learning is advantageous for the NSL-KDD dataset, as the top results from Level 01 outperform those from Level 00. Specifically, eight models at Level 01 (Table 8) show higher F1 scores than the best model at Level 00 (Table 6). Conversely, for the RoEduNet-SIMARGL2021 dataset, it is preferable to remain at Level 00 due to negligible performance gains at Level 01, although seven weaker base learner models do show potential improvements. For the CICIDS-2017 dataset, 17 out of 21 models demonstrate better outcomes when comparing the top results of Level 01 (Table 15) with those of Level 00 (Table 14). We also conduct a False Positive Rate (FPR) experiment to compare Levels 01 and 00. The best results were achieved using Level 01 with All Features and Probabilities. Notably, the CAT, LGBM, Bag_LGBM, and XGB models excelled in FPR across all datasets, with NSL-KDD showing the most significant improvement–reducing FPR from 5% to 1% between the best base learner in Level 00 and the top method in Level 01. The other datasets also showed improvements, albeit to a lesser extent. Additionally, we performed a statistical analysis using the Wilcoxon signed-rank test on various models across the three datasets, confirming the statistical significance of most of our models (Section VI-E).

This research moves towards bridging the gap in applying ensemble learning methods to network IDS by extensively evaluating and comparing different metrics crucial for network security in AI models, such as accuracy, precision, recall, F1, FPR, and runtime. Our framework enhances the use of ensemble learning in network intrusion detection systems, contributing significantly to advancements in this crucial network security research area.

Summary of Contributions: We summarize below our main contributions.

- We propose a two-level framework for ensemble learning for network intrusion detection tasks. Our framework uses two levels of learning for evaluating different metrics for network intrusion detection.
- We evaluate our framework on three popular network intrusion detection datasets, which are the CICIDS-2017 and NSL-KDD benchmark datasets, and the real-world RoEduNet-SIMARGL2021 dataset.
- We evaluate our framework under 21 AI models (base learners and ensemble methods) for Level 00, and 24 AI models (ensemble methods) for Level 01.
- We compare the performance of different ensemble learning methods and different base learners under feature selection.

[1] To the best of our knowledge, no prior work applied ensemble learning methods for this recent dataset, as discussed in the related work section.

- We release our source codes for the community to access it as a baseline two-level ensemble learning framework for network intrusion detection and to build on it with new datasets and models.[2]

II. RELATED WORK

A. EXISTING EFFORTS IN LEVERAGING ENSEMBLE LEARNING FOR IDS

The survey [20] provides an overview of IDS from 2009 to 2020, emphasizing the development of ensemble systems like Stacking, Bagging, Boosting, and voting. It reviews the state of the art in ensemble models, examining a variety of datasets such as CICIDS-2017, KDD'99, NSL-KDD, Kyoto 2006+, and AWID, and models like NN, SVM, DT, Fuzzy Clustering, and RBF. The study's main contribution is a detailed analysis that encourages exploring new ensemble methods, offering valuable insights for future IDS research. Hence, we divided this section into two groups: Binary and Multiclassification.

1) ENSEMBLE LEARNING FOR BINARY CLASSIFICATION ANOMALY DETECTION APPROACHES

The article [25] introduces an anomaly detection framework using datasets like CICIDS-2017, UNSW-NB15, and KDD'99, applying feature selection via the Chi-square method, and leveraging base models such as Gaussian Naive Bayes, Logistic Regression, and Decision Tree. Predictions are generated using a Stochastic Gradient Descent ensemble model. This study highlights the use of stacking to enhance IDS performance across various datasets, though it notes limitations such as data imbalance, suggesting that data augmentation could help in solving this imbalance limitation. Additionally, the work [29] presents an ensemble framework for binary anomaly classification in IDS, utilizing NSL-KDD and UNSW-NB15 datasets with models like Random Forest, AdaBoost, XGBoost, and Gradient Boosting Decision Tree, using soft voting for results integration. This method aims to increase cyber-attack detection accuracy and reduce false alarms. Similarly, the work [30] applies ensemble techniques like Majority Voting alongside LR (Logistic Regression), DT (Decision Tree), NB (Naive Bayes), NN (Neural Network), and SVM to NSL-KDD, UNSW-NB15, and CICIDS2017, achieving superior performance. It suggests the need for new, especially real-world, datasets and unsupervised learning methods. Another study [32] implements its framework on the Palo Alto network log and NSL-KDD, UNSW-NB15 datasets, using weighted voting with SVM, Autoencoder, and Random Forest for anomaly detection, highlighting real-world application benefits but noting scalability and limited voting method efficiency.

In the IoT context, the article [33] introduces an anomaly detection framework using the TON-IoT dataset with base models including Random Forests, Decision Trees, Logistic Regression, and K-Nearest Neighbors. These are integrated using ensemble methods like stacking and voting to improve attack detection. However, its limitations include not testing the framework on other datasets and not exploring other ensemble methods like bagging and averaging. Another study, [26], presents an online anomaly detection system for network intrusion using ensemble Autoencoders, focusing on real-time applications. Additionally, the article [27] addresses overfitting in ensemble learning for small binary classification datasets, using models like Random Forest, Naive Bayes, and Logistic Regressor. It features a model selection procedure to find the best model for specific instances, with the main drawback being the high computational cost from cross-validation. This approach also explores pruning to combat overfitting, though its effectiveness varies across different datasets and models.

2) ENSEMBLE LEARNING FOR MULTICLASS CLASSIFICATION APPROACHES

Several studies have applied ensemble learning to multiclass IDS. The work [23] introduces a novel approach using stacking on TensorFlow models (CNN, DNN, RNN, LSTM) across datasets like CICIDS-2017 and ToN_IoT, using class predictions to train a subsequent DNN, marking its innovation from using class probabilities. The main limitations include no real IoT testing, reliance on a single ensemble method, and high resource use on IoT devices. Another significant study, [24], introduces the GTCS (Game Theory and Cyber Security) dataset to address benchmark dataset flaws and uses the Weka toolkit for adaptive ensemble learning with feature selection via Information Gain, applying J48, MLP, and IBK models with majority voting. Challenges here include the lack of real-world testing and a limited range of AI models.

Additionally, [28] combines Linear Genetic Programming (LGP), Adaptive Neural Fuzzy Inference System (ANFIS), and RF for higher accuracy and lower false alarms but faces challenges in optimal weight assignment for weighted voting and lacks broad testing across datasets. Meanwhile, [34] employs bagging with Naïve Bayes, PART, and Adaptive Boosting on the KDD'99 dataset, using voting and bootstrapping but remains untested on newer datasets.

Moreover, the prior work [31] develops the MFFSEM (Multi-Dimensional Feature Fusion and Stacking Ensemble Mechanism) method, blending feature fusion with stacking ensemble learning for better accuracy and applying it to multiple datasets using DT and RF as base models, yet struggles with real-world deployment and limited ensemble methods. Lastly, [35] leverages CVM (Core Vector Machine), an SVM variant, achieving high accuracies and faster performance with a focus on weighted voting, yet it lacks validation on modern datasets and traditional ensemble methods comparison.

B. CONTRIBUTION OF THIS WORK

We list our specific contributions in four main points.

[2]The URL for our source codes is: https://github.com/ogarreche/Ensemble_Learning_2_Levels_IDS

TABLE 1. A comparison between different aspects of our own work and prior relevant works on ensemble learning for network intrusion detection (including methods for ensemble learning, datasets, and AI models).

Paper	Dataset	Base Model	Ensemble Model	Extensive Evaluation
Our Work	CICIDS-2017, NSL-KDD, RoEduNet-SIMARGL2021	LR, DNN, MLP, DT, SVM, KNN	ADA, LGBM, XGB, CAT, Bagging, Stacking, Boosting	Yes
A stacking ensemble of deep learning models for IoT intrusion detection [23]	CICIDS-2017, ToN_IoT	DNN,CNN,RNN,LSTM	DNN	No
Ensemble Classifiers for Network Intrusion Detection Using a Novel Network Attack Dataset [24]	GTCS	J48, MLP, and IBK	Majority voting	No
A novel ensemble learning-based model for network intrusion detection [25]	KDD'99, CICIDS-2017, UNSW-NB15	Gaussian Naive Bayes, LR, DT	SGD	No
Kitsune [26]	Kitsune (OS Scan, Fuzzing, and Mirai)	Kitnet, GMM, SVM, DNN, Autoencoders	Kitsune	No
Classification and Clustering Based Ensemble Techniques for Intrusion Detection Systems: A Survey [20]	KDD'99, NSL-KDD, Kyoto 2006+,AWID Dataset, CICIDS-2017	DNN, SVM	Voting-based, Weighted majority voting	No
Ensemble Selection from Libraries of Models [27]	ADULT, BACT, COD, CALHOUS, COVTYPE, HS, LETTER.P1, LETTER.P2, MEDIS, MG, SLAC	RF, NB, LR	Ensemble selection procedure	No
Ensemble Classifiers for Network Intrusion Detection System [28]	KDD'99	LGP, ANFIS, RF	Weighed Voting	No
A Network Intrusion Detection System Using Ensemble Machine Learning [29]	NSL-KDD, UNSW-NB15	Random Forest, AdaBoost, XGBoost,Gradient boosting decision tree	Soft voting scheme	No
Network Intrusion Detection and Comparative Analysis Using Ensemble Machine Learning and Feature Selection [30]	NSL-KDD, UNSW-NB15, and CICIDS2017	LR, DT, NB, NN, SVM	Majority Voting, DT, NB, LR, NN, SVM	No
Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection [31]	KDD'99, NSL-KDD, UNSW-NB15, CICIDS-2017	DT, RF	CAT,LGBM, ADA,ExTra Trees, Voting,MFFSEM	No
Toward an Online Network Intrusion Detection System Based on Ensemble Learning [32]	NSL-KDD, UNSW-NB15, Palo Alto network log	Autoencoder, SVM, RF	Weighed Voting	No
Ensemble-Learning Framework for Intrusion Detection to Enhance Internet of Things Devices Security [33]	TON-IoT	DT, RF, LR, KNN	Stacking, Voting	No
An Ensemble Approach for Intrusion Detection System Using Machine Learning Algorithms [34]	KDD'99	NB, ADA, Part	Voting, Bagging, Bootstrapping	No
A Network Intrusion Detection System Based On Ensemble CVM Using Efficient Feature Selection Approach [35]	KDD'99	CVM	Weighed Voting	No

1) **Generalizability:** In Level 00, we apply 21 Machine learning algorithms (e.g., Table 6) within the scenarios of using feature selection or not. In Level 01, we use the datasets generated in Level 00 and apply 21 ML algorithms plus three other methods (i.e., Voting, Averaging, and Weighted Averaging) (as shown in Table 8). The process is done considering all features and feature selection and considering Level 00's predicted classes (e.g., Table 16) and Level 00's predicted probabilities (e.g., Table 15). Moreover, this process is applied to three datasets (i.e., RoEduNet-SIMARGL2021, CICIDS-2017, and NSL-KDD). These extensive experiments improve the generalizability of our framework.
2) **Extensive Evaluation:** The extensiveness of our work is seen when we compared the number of experiments performed in the works of Table 1 and the number of experiments performed in this work. In numbers, our results' tables show 126 evaluations for Level 00 (Tables 14, 10, 6), 288 evaluations for Level 01 (Tables 16, 15, 11, 12, 7, 8), and 252 evaluations for FPR (Tables 18, and 19).
3) **Expanded Insights:** In this paper, we compare our experiments in many layers, Level 00 versus Level 01, Feature Selection against no Feature Selection, and using predicted probabilities against class predictions. Such comparisons allowed us to extract different behaviors for FPR, accuracy, precision, recall, F1, and runtime performance. Such results are displayed in Table 21, which shows the best setup and models in different scenarios.
4) **Framework Flexibility:** The framework's conception has its core in modularity and flexibility. It allows users to adapt it to other datasets. Also, it has options

for feature selection and class prediction or predicted probabilities. Also, differently from other works, the differentiation of Level 00 and Level 01 does not translate directly to Base Learners and Ensemble Learning because we also apply ensemble learning methods to Level 00. A more accurate definition is that we use the ML methods directly to the three datasets in Level 00. Meanwhile, in Level 01, we apply the same models, plus voting and averaging, to the newly generated datasets, which is an innovation compared to other related works (shown in Table 1). Moreover, we open-sourced the codes to the community.

III. THE PROBLEM STATEMENT

We now provide the main preliminaries for network intrusion detection, the challenges of AI, the need for ensemble learning, and the challenges in evaluating such methods when applied to network intrusion detection.

A. NETWORK INTRUSION TYPES

There are several common network intrusion types. In our work, we consider the main network attacks in the common MITRE ATT&CK framework [41]. Thus, the network traffic can be divided into the following categories:

Normal traffic: is the regular traffic collected from the network.

PortScan (PS) / Network Service Discovery [MITRE ATT&CK ID: T1046]: It is an intrusion in which the attacker aims to make recognizance of the victim's computer. It is often used as the first step of an attack to search for vulnerable points and possible entrance ports. The functionality is to send a connection solicitation to the victim without ever finalizing the connection. However, for this attack, the solicitations are sent to various ports and the ones that send a message back are mapped as possible entrance points [42].

Denial of Service (DoS) / Network Denial of Service [MITRE ATT&CK ID: T1498]: It is a type of attack where the objective is to render the target unavailable for the network. One popular example of such an attack is where the attacker keeps sending solicitations to connect with a server. However, when the server accepts the solicitation and sends an acknowledgment to the origin expecting its response it never receives a response. As a result, the memory for these open communications is left open and fully consumed until the server becomes unavailable. We refer to [37] and [38] for detailed classes of DoS attacks.

Brute Force [MITRE ATT&CK ID: T1110]: It is an attack in which all password possibilities are attempted by the intruder to break into the victim's network. This attack is often paired with the knowledge of the most commonly used passwords. It can become effective in cases in which a user has a weak or a common easy-guessing password [37].

Web Attack / Initial Access [MITRE ATT&CK ID: TA0001, T1659, T1189]: It is a class of attacks performed via the web in which the attacker exploits web vulnerabilities. For example, an attacker may gain access to the application's underlying instance or container by exploiting a public-facing application, using a software bug, misconfiguration, or glitch. The web attack can also include attacks such as the Drive-by Compromise [43], however, web attacks (e.g., SQLi, XSS) typically do not provide initial access to a remote server [41].

Infiltration / Initial Access [MITRE ATT&CK ID: TA0001]: This attack occurs when someone tries to get initial access to a system or application. Such a class of attacks consists of a myriad of techniques as targeted spearphishing and exploiting weaknesses on public-facing web servers. The foothold gained by this attack can range from a simple change of password to continued access through valid accounts and external remote services.

Botnet / Compromise Infrastructure [MITRE ATT&CK ID: T1584.005, T1059, T1036, T1070]: It is an automated class of attacks which is performed by hijacked devices executed remotely by the attacker in which scripts (bots) mimic human behavior and duplicate it [44]. This scripted technique allows scalability and easy deployment making it an ideal tool to touch multiple attack points simultaneously. Thus, Botnet is a very common network attack type.

Probe Attack/ Network Scanning or Surveillance [MITRE ATT&CK ID: T1595]: Probe attacks are often the preliminary stage of a more comprehensive attack. These attacks involve scanning a network to gather information or find known vulnerabilities [45]. An attacker with a map of machines and services that are available on a network can use this information to look for exploits. We emphasize that while port scanning is a form of probe attack, not all probe attacks are port scans. Some might target specific vulnerabilities or use different methods. Examples include ping sweeps [46], and DNS zone transfers [47].

Remote to Local Attack (R2L) [MITRE ATT&CK ID: TA0001, T1110, T1078]: R2L is a class of attacks in which an attacker (who can send packets to a machine over a network but does not have an account on that machine) gains unprivileged access as a user of that machine (i.e., initial access). Once the attacker has user-level access, they may attempt to escalate privileges to gain control over the entire system (i.e., U2R attack).

User to Root Attack (U2R) [MITRE ATT&CK ID: TA0004, T1078]: User to Root attack is a type of exploit in which the attacker starts by accessing an unprivileged user account on the system (via sniffing passwords, a dictionary attack, or social engineering) to exploit some vulnerability to gain root access to the system. This can be extremely detrimental as it can allow the attacker to manipulate the system as a root user or an administrator. Note that R2L is a necessary preliminary for Privilege Escalation (i.e., U2R).

B. INTRUSION DETECTION SYSTEMS

The increasing sophistication of network attacks poses a significant threat to critical infrastructure across various sectors [48], [49]. Consequently, IDS plays a crucial role in safeguarding computer network systems against malicious

activities, whether initiated by internal users or external infiltrators [50]. Traditional IDS designs typically rely on the assumption that an intruder's behavior will deviate noticeably from that of a legitimate user, making many unauthorized actions detectable [51]. With recent advancements in AI over the past decade, this design paradigm has paved the way for the development of AI models capable of automatically detecting network intrusions [52].

C. SHORTCOMINGS OF BASE LEARNER MODELS

Although AI models have significantly automated intrusion detection, their inherent complexity poses limitations due to the intricate nature of their learning and decision-making processes. This complexity makes it challenging for a single model to fully comprehend the nuances of datasets, leading to difficulties in learning specific subsets and achieving satisfactory metrics for certain results. This challenge is prevalent across various AI models, including Decision Trees (DT), K-nearest neighbors (KNN), Support Vector Machines (SVM), Deep Neural Networks (DNN), and others. Despite their high predictive accuracy in Intrusion Detection Systems (IDS), there remains a gap in achieving better accuracy, precision, recall, and F1 scores, especially in the event of errors or attacks (including high false positive rate for some of these AI models [18] and high false negative rate for some other AI models [19]). This issue is particularly critical in safety-sensitive applications such as network security through IDS. Consequently, there is a growing motivation to improve performance and expand the usage of AI models in the context of IDS. This has motivated the pressing need to leverage different ensemble learning methods for enhancing IDS via combining different base learner models [20], [21], [22].

D. MAIN BENEFITS OF POPULAR ENSEMBLE METHODS

Recall that base learners or individual models have unique strengths points and weaknesses, and depending on the application or task, one may be better than the other, which might aggravate which model to choose. Machine learning algorithms have unique underlying mechanics. For instance, KNN, which relies on clustering similar data around centroids, is sensitive to the number of clusters (i.e., K), class outliers, and irrelevant features. Moreover, it is computationally expensive. Another example is Deep Neural Networks (DNN). Usually, they need large volumes of data and might be time-resource intensive, besides being sensitive to perturbations in the input data. Regression techniques (e.g., Logistic Regression) are simple to apply and explain. However, it often fails to capture complex relationships (i.e., higher-order polynomials). As a last example, Decision Trees are fast to train, though they can oversimplify problems and lead to overfitting. Hence, combining these AI models can improve their use generalization, and application in network intrusion detection tasks by combining their strengths and diluting their weaknesses.

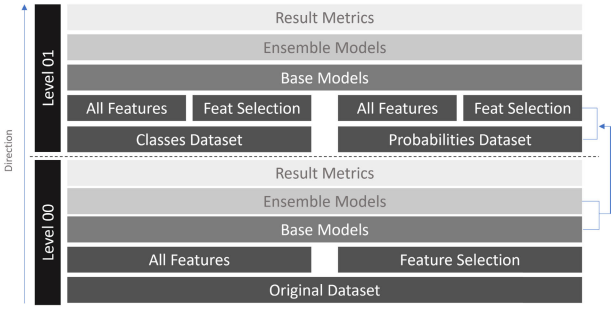


FIGURE 1. A high-level overview of our two-level ensemble learning framework for network intrusion detection.

Ensemble Learning is the field that explores this concept of combining the strengths of different base learners. Famous ensemble techniques are Bagging, Boosting, and Stacking. Bagging is a technique that creates a subset of the datasets through bootstrapping (i.e., data sampling with replacement) and inserts them into different instances of a machine learning model, training them in parallel (i.e., independently) and combining their outputs. The goal of Bagging is to reduce overfitting and improve generalization. In a different light, the Boosting technique considers training different instances of the same model sequentially, aiming to correct mistakes from the previous one by assigning higher weights (i.e., emphasis) to misclassified data points, avoiding the same mistakes. Meanwhile, the Stacking method usually refers to training base learners (i.e., part of our Level 00 in our framework) and using their predictions to train a meta-model (i.e., our Level 01 in our framework). It allows the usage of a myriad of base learners, capitalizing on their uniqueness to achieve the complex behavior between features and prediction.

In our work, we explore these different types of ensemble learning models in our two-level learning framework and compare the performances of these methods (on both levels in our framework) with just using the base models for network intrusion detection tasks. We perform such comparisons for three different datasets with different characteristics to get a deeper understanding of our proposed framework.

IV. FRAMEWORK

The main goal of this work is to provide an ensemble learning pipeline that derives improved result metrics for each dataset. Our framework can help in choosing efficient methods that help security analysts better identify invasions and classify attacks on the network traffic to prevent intrusions under their management. The components of our framework are detailed below in a high-level overview in Figure 1, and a low-level overview in Figure 2.

A. HIGH-LEVEL ENSEMBLE LEARNING PIPELINE COMPONENTS

Figure 1 shows two major areas (i.e., Level 00 and Level 01) divided by a horizontal line, indicated by the blocks on the left-handed side in a vertical position. The diagram is

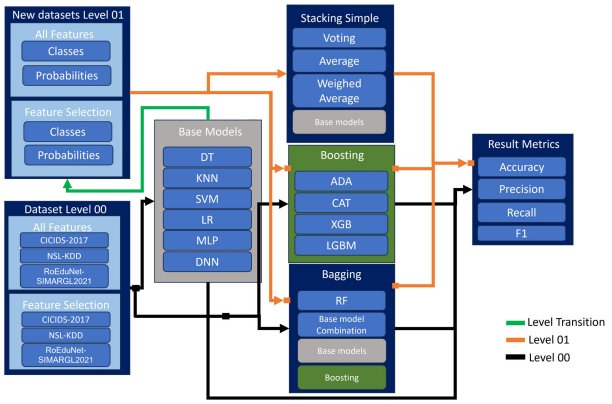


FIGURE 2. A low-level overview of our Ensemble Learning framework for network intrusion detection. It considers a diverse set of AI models and network intrusion datasets.

read from bottom to top, analogous to a pyramid that starts by building a strong foundation and then grows vertically with stacking, and its vertical layout is due to the ensemble learning methods applied in this work.

1) BLOCKS OF LEVEL 00

Level 00 contains all the blocks to its right (i.e., original dataset, all features, feature selection, base models, ensemble models, and result metrics). The elements' description is as follows. The "original dataset" block is the input dataset we are experimenting on (i.e., in our context, it can be CICIDS-2017, NSL-KDD, and RoEduNet-SIMARGL2021). Next, the framework forks into two scope possibilities (i.e., using all features available in the dataset or a selected subset of the features), which are the blocks named "All features" and "Feature Selection." After collecting the network traffic data, our framework feeds such data into different black-box AI models, where these models detect intrusions and normal traffic via constructing a multi-class classification problem. In this classification problem, each class represents one possible network intrusion (e.g., denial of service, port-scanning, and brute force attacks). In this context, the blocks "Base models" and "Ensemble models" refer to the models used (i.e., ADA, Bagging-ADA, RF, Bagging-RF, DT, Bagging-DT, KNN, Bagging-KNN, MLP, Bagging-MLP, CAT, Bagging-CAT, LR, Bagging-LR, LGBM, Bagging-LGBM, Bagging-Combination, DNN, and XGB). See Section V and Appendix A for more details. Finally, the block "Result metrics" is the evaluation method consisting of main evaluation metrics which are Accuracy, Recall, Precision, F1, and runtime.

2) BLOCKS OF LEVEL 01

Level 01 contains all the blocks to its right (i.e., Classes Dataset, Probabilities Dataset, All features, Feature Selection, Base models, Ensemble models, and Result metrics). The elements are the same as Level 00, but the datasets are different (those generated from Level 00). Note that there is an arrow connecting Level 00 to Level 01. Such an arrow represents the creation of new datasets based on Level 00. These datasets use the base models (i.e., ADA, LGBM, CAT, XGB, DNN, MLP, RF, LR, and DT) and their predicted classes in the "Classes Dataset" case or predictions probabilities in the "Probabilities Dataset" case. Then, there is a second split, creating four new datasets. This split is due to the framework using or not using feature selection in Level 00 (e.g., see Table 22 for the top features used in the case of feature selection for each dataset). As a side note, the idea of using the actual classes instead of the prediction probabilities is inspired by the paper [23] in which they achieved better results using the method. Another difference in decision-making in Level 01 is that feature selection in Level 01 (e.g., stacking) is based on choosing which predictions or classes the user wants to use from the available models (i.e., ADA, LGBM, CAT, XGB, DNN, MLP, RF, LR, and DT) instead of the original features from the original dataset in Level 00.

B. LOW-LEVEL ENSEMBLE LEARNING PIPELINE COMPONENTS

We now explain the Low-Level Ensemble Learning Pipeline Components (shown in Figure 2). The reader may follow the arrows in the block diagram to facilitate its readability. Each arrow type has a color code: black refers to processes in Level 00, orange relates to processes in Level 01, and the green arrow marks the transition from Level 00 to Level 01. Each block group has sub-groups that indicate that they are a set or included in the context of the outer block. Also, note that some blocks have different colors for easy identification (e.g., the base models are gray). These blocks seen before may appear in another context (i.e., inside other blocks). For example, note that base models (gray) appear as a small gray block inside stacking. First, we explain the components in Level 00 and then those of Level 01.

1) NETWORK INTRUSION DATASETS - LEVEL 00

The first component in our pipeline is loading a network intrusion dataset from the database as a starting point. In our work, we use three popular network intrusion detection datasets, which are RoEduNet-SIMARGL2021 [38], CICIDS-2017 [39], and NSL-KDD [40] datasets. For more information on the datasets, see Section V and Table 5b.

2) NETWORK INTRUSION DATASETS - LEVEL 01

The datasets in Level 01 have their origin from the models results in Level 00 (i.e., ADA, LGBM, CAT, XGB, DNN, MLP, RF, LR, and DT), and their size is the test size from Level 00 (i.e., 30% of the size used in Level 00). There are four categories for such datasets in Level 01, as explained below.

- *Probabilities and All Features:* AI models generate prediction results probabilities before deciding the outcome, which indicates how certain it is on a given result. We extract the class probability corresponding to

its correct label from the original test dataset to create a new one. Hence, each column of this new dataset (i.e., features) corresponds to the model used in Level 00, and it has a value range from 0 to 1 corresponding to the prediction probability of that model. Also, all features are used in Level 00 to generate this dataset.

- *Probabilities and Feature Selection:* It is similar to the above first dataset, but with one distinction where the models in Level 00 use feature selection (i.e., top features instead of all of the features).
- *Classes and All Features:* In this scenario, the information used to create the dataset is the prediction itself (e.g., if there are three classes, zero, one, or two, these will be the values present in the dataset feature columns). Also, all features are used in Level 00 to generate this dataset.
- *Classes and Feature Selection:* It is similar to the ''Classes and All features'' above, but with the difference that the models in Level 00 use the feature selection process.

Having explained the datasets used for both levels, we next explain the feature selection process performed at each level.

3) FEATURE SELECTION - LEVEL 00

The best eight features used in Level 00 originate from our previous work [36] that analyzes feature selection in different scenarios. The best result from such work is using the Feature Selection based on a SHAP XAI method. In that work, we extracted feature importance lists using SHAP for each model analyzed (e.g., ADA, LGBM, RF, KNN, MLP, DNN, and SVM), and we combined the lists using frequency analysis to generate a common feature importance list across all AI models. The results showed that this approach yielded better results than classic feature selection methods (e.g., information gain, chi-square, and others) for the three datasets (RoEduNet-SIMARGL2021, CICIDS-2017, and NSL-KDD) considered in this current work.

4) FEATURE SELECTION - LEVEL 01

For Level 01, which uses the newly generated datasets, we opted for using Information Gain as our method to obtain the top five most influential features when applying feature selection for two reasons. First, information gain is a well-established, easy-to-implement, quick, and renowned method that measures how much of gained information after splitting the dataset for each feature, diminishing the entropy by each iteration. Second, we do not have XAI-based feature selection for Level 01 since our prior work [36] considered feature importance for base models and did not consider the two-level learning process in this current work.

Having explained the feature selection process for both levels, we next detail our model space.

5) BASE LEARNERS AI MODELS - LEVEL 00

Once the dataset preprocessing is complete, we split it (i.e., 70% training and 30% testing) before training the AI models. For this component, we have built six popular AI classification models: Decision Tree (DT), Logistic Regression (LR), deep neural network (DNN), multi-layer perceptron (MLP), k-nearest neighbors (KNN), and support vector machine (SVM). Each model requires specific parameters to ensure the best possible performance (see Section V).

6) ENSEMBLE LEARNERS AI MODELS - LEVEL 00

We also use Boosting and Bagging techniques to check the performance of ensemble learning in Level 00, following the same pipeline of performing splitting of the data with a split of 70% for the training while leaving the unseen 30% for testing purposes. For this component, we have built four popular Boosting techniques which are:

- LightGBM (LGBM)
- adaptive boosting (AdaBoost)
- eXtreme Gradient Boosting (XGB)
- CatBoosting (CAT)

Bagging Variants: Furthermore, we used 10 Bagging variations which are:

- Random Forest (RF)
- Bagging with RF
- Combination of different learners (DT, KNN, MLP, LR, SVM, ADA, DNN, MLP, XGB) in a single bagging algorithm
- Bagging of boosting algorithms (ADA, CAT, LGBM)
- Bagging of base learners (DT, KNN, MLP, LR, SVM)

Each model requires specific parameters to ensure the best possible performance (see Section V for details).

7) ENSEMBLE LEARNERS AI MODELS - LEVEL 01

We also use Boosting and Bagging techniques to check the performance of ensemble learning in Level 01, following the same pipeline of performing splitting of the data with a split of 70% for the training while leaving the unseen 30% for testing purposes. For this component, we have built the same four popular Boosting techniques used in Level 00, which are LightGBM (LGBM), adaptive boosting (AdaBoost), eXtreme Gradient Boosting (XGB), and CatBoosting (CAT).

Bagging Variants of Level 01: We used 10 Bagging variations:

- Random Forest (RF)
- Bagging with RF
- Combination of different learners (DT, KNN, LR, SVM, ADA, DNN, MLP, XGB) in a single bagging algorithm
- Bagging of boosting algorithms (ADA, CAT, LGBM)
- Bagging of base learners (DT, KNN, MLP, LR, SVM)

Stacking on Level 01: Arguably, every model in Level 01 is a stacking method variant for Level 00. Since stacking uses the predictions of base learners as input for a meta-model, it implies that every model of Level 01 is a meta-model. For this work, we considered the following nine methods as stacking:

- Simple Stacking (Voting, Averaging, and Weighted Average).

TABLE 2. Description of main features for RoEduNet-SIMARGL2021 dataset [53].

RoEduNet-SIMARGL2021 Features	Explanation
FLOW_DURATION_MILLISECONDS	Flow duration in milliseconds
PROTOCOL_MAP	IP protocol name (tcp, ipv6, udp, icmp)
TCP_FLAGS	Cumulation of all flow TCP flags
TCP_WIN_MAX_IN	Max TCP Window (src->dst)
TCP_WIN_MAX_OUT	Max TCP Window (dst->src)
TCP_WIN_MIN_IN	Min TCP Window (src->dst)
TCP_WIN_MIN_OUT	Min TCP Window (dst->src)
TCP_WIN_SCALE_IN	TCP Window Scale (src->dst)
TCP_WIN_MSS_IN	TCP Max Segment Size (src->dst)
TCP_WIN_SCALE_OUT	TCP Window Scale (dst->src)
SRC_TOS	TOS/DSCP (src->dst)
DST_TOS	TOS/DSCP (dst->src)
FIRST_SWITCHED	SysUptime of First Flow Packet
LAST_SWITCHED	SysUptime of Last Flow Packet
TOTAL_FLOWS_EXP	Total number of exported flows

TABLE 3. Description of the main features for the CICIDS-2017 dataset [54].

CICIDS-2017 Features	Explanation
Packet Length Std	Standard deviation length of a packet
Total Length of Bwd Packets	Total size of packet in backward direction
Subflow Bwd Bytes	Average number of bytes in backward sub-flow
Destination Port	Destination Port Address
Packet Length Variance	Variance length of a packet
Bwd Packet Length Mean	Mean size of packet in backward direction
Avg Bwd Segment Size	Average size observed in the backward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Init_Win_Bytes_Backward	Total number of bytes in initial backward window
Total Length of Fwd Packets	Total packets in the forward direction
Subflow Fwd Bytes	Average number of bytes in a forward sub-flow
Init_Win_Bytes_Forward	Total number of bytes in initial forward window
Average Packet Size	Average size of packet
Packet Length Mean	Mean length of a packet
Max Packet Length	Maximum length of a packet

- Stacking with all base models (DT, KNN, LR, SVM, DNN, MLP).
- Combination of different learners (DT, KNN, LR, SVM, ADA, DNN, MLP, XGB) in a single bagging algorithm.
- Bagging of boosting algorithms (ADA, CAT, LGBM).
- Bagging of base learners (DT, KNN, MLP, LR, SVM).

Each model requires specific parameters to ensure the best possible performance (see Section V).

8) RESULT METRICS - LEVEL 00|01

As the result metrics for both levels, we use the five performance indicators: Accuracy, Precision, Recall, F1, FPR, and runtime. We then group all results for analysis and comparison.

9) AI MODELS SELECTION CRITERIA

The AI models considered in this work were chosen for the following reasons. The first reason is that they are ubiquitous in many similar works that focus on IDS (e.g., [37], [38], [40]). The second reason is to study the effect of XAI-based feature selection on the performances of AI models in IDS, which enables us to compare our methods with prior works on the three considered datasets [37], [38], [40]. In this manner, we can help keep consistency with the literature in IDS when comparing different models paired with the two-level ensemble learning techniques considered in our current work.

TABLE 4. Description of main features for the NSL-KDD dataset [40].

NSL-KDD Features	Explanation
duration	Length of the connection
protocol_type	Type of the protocol (e.g., TCP, UDP)
service	Network service on the destination (e.g., HTTP, FTP)
flag	Normal or error status of the connection
src_bytes	Number of data bytes from source to destination
dst_bytes	Number of data bytes from destination to source
logged_in	1 if successfully logged in; 0 otherwise
count	Number of connections to the same host as the current connection in the past two seconds
srv_count	Number of connections to the same service as the current connection in the past two seconds
dst_host_same_srv_rate	Rate of connection to the same service on destination host
dst_host_srv_count	Count of connections to the same service on destination host
dst_host_same_src_port_rate	Rate of connections to same source port on destination
dst_host_serror_rate	Rate of connections that have activation flags indicating various types of errors
dst_host_rerror_rate	Rate of connections that have rejected flags

C. TOP INTRUSION FEATURES LIST AND USAGE IN ENSEMBLE LEARNING

We now show the full list of the top network intrusion features along with their explanations for the three studied datasets since we use those frequently throughout the rest of the paper. In particular, Tables 2-4 show the description for each feature in RoEduNet-SIMARGL2021, CICIDS-2017, and NSL-KDD network intrusion datasets, respectively. We emphasize that we will utilize such features in evaluating the ensemble learning methods on the three datasets in our evaluation (Section V).

Remark: In our article, Tables 2-4 provide a description of the key features for the RoEduNet-SIMARGL2021, CICIDS-2017, and NSL-KDD datasets, respectively. These tables are intended to highlight some of the key features from each dataset for clarity and context. However, we did indeed utilize all the features listed in Table 5b in our initial experiments. This comprehensive approach allowed us to fully leverage the datasets for our analysis of network intrusion detection. It is important to note that Table 5b summarizes the overall scope of each dataset, including the number of features. The feature selection used in evaluation experiments of our two-level ensemble learning framework was tailored to the objectives of those specific analyses. This selective usage of features was based on their importance and relevance to the specific experiment being conducted.

V. FOUNDATIONS OF EVALUATION

We next show our detailed evaluation. Our evaluation aims to answer the following research questions:

- What are the best base models for a given dataset?
- Which ensemble method has the best performance for a given dataset?

TABLE 5. Summary and statistics of the three network intrusion datasets used in this work, including the size of the dataset, number of attack types (labels), number of intrusion features, and distribution of samples among attack types.

(a) Basic statistics of datasets

Dataset	No. of Labels	No. of Features	No. of Samples
CICIDS-2017	7	78	2,775,364
RoEduNet-SIMARGL2021	3	29	31,433,875
NSL-KDD	5	41	148,517

(b) Distribution of samples among different attack types

Dataset	Normal	DoS	PortScan	Brute Force	Web	Bot	Infilt.	Probe	U2R	R2L
CICIDS-2017	84.442%	9.104%	5.726%	0.498%	0.157%	0.071%	0.001%	-	-	-
RoEduNet2021	62.20%	24.53%	13.27%	-	-	-	-	-	-	-
NSL-KDD	48%	35%	-	-	-	-	-	10.05%	0.47%	6.48%

- What are the performances of the evaluated methods in our two-level framework in terms of Accuracy, Precision, Recall, FPR, F1, and runtime?
- Which ensemble methods and AI models are statistically significant?
- What are the limitations and strengths of ensemble learning methods when applied to network intrusion detection?
- What are the main insights from the evaluation experiments on each dataset?

Before showing our detailed evaluation results, we first detail the experimental setup in this section.

A. DATASET DESCRIPTION

RoEduNet-SIMARGL2021 Dataset [38]: This dataset originates from the SIMARGL project, a collaborative effort supported by the European Union through the Horizon program, in partnership with the Romanian Education Network (RoEduNet). It comprises authentic network traffic data, including features derived from real-time traffic analysis. The dataset is structured following a data schema resembling Netflow [55], a network protocol developed by CISCO for capturing and monitoring network flows.

CICIDS-2017 Dataset [39]: This dataset serves as a benchmark for intrusion detection and was developed by the Canadian Institute for Cybersecurity at the University of Brunswick in 2017. It encompasses six distinct attack profiles, encompassing activities such as brute force, heartbleed, botnet, DoS, portscan, web attack, and infiltration attack. To create a realistic context, the dataset incorporates background traffic generated through a B-Profile system [56], which extracts various user behaviors based on network protocols.

NSL-KDD Dataset [40]: This dataset represents an improved version designed to address inherent issues found in the older KDD dataset [57]. It was jointly developed through collaboration between the University of New Brunswick and the National Research Council of Canada. This dataset incorporates a blend of network traffic data, featuring a mixture of various attack types and normal instances. It adheres to a structured data schema, with features extracted from raw traffic data, rendering it a valuable resource for assessing intrusion detection techniques. For our research purposes, we employ the KDDTrain+ subset for model training and the KDDTest+ subset for testing and evaluation [58].

Summary and Statistics of the Datasets: Having explained the three datasets, Table 5a shows the size of each dataset, the number of attack types (labels), and the number of intrusion features.

B. EXPERIMENTAL SETUP

Computing Resources: For the executed experiments, we used a high-performance computer (HPC) that has four NVIDIA A100 GPUs, 64 GPU-accelerated nodes (each with 256 GB of memory), and a single 64-core AMD EPYC 7713 processor (2.0 GHz and 225-watt) reaching a maximum performance of approximately 7 petaFLOPs. This supercomputer is designed to aid researchers in advanced AI and machine learning tasks [59].

Coding Tools: To make use of several open-source tools and to have our implementation as open-source, we chose the Python programming language along with different AI toolboxes (including Keras and ScikitLearn). We also used other toolboxes (including Pandas and Matplotlib).

AI Models: We detail now the used AI models in our work.

(i) Base Learners: By pairing six popular AI classification algorithms as base learners (which are deep neural network (DNN) [7], k-nearest neighbor (KNN) [60], support vector machine (SVM) [8], multi-layer perceptron (MLP) [61], Decision Tree (DT) [62], Logistic Regression (LR) [63]), we evaluate AI methods and different components of our proposed two-level framework for IDS.

(i) Ensemble Methods: Regarding the ensemble techniques used in our framework, the Boosting techniques are Cat Boosting (CAT) [64], light gradient-boosting machine (LGBM) [65], AdaBoost (ADA) [66], and Extreme-Gradient-Boosting (XGBoost) [67]. Also, we apply Bagging techniques (i.e., using different instances of the same model), which includes Random Forest (RF) [15] and the models mentioned before (i.e., DT, KNN, MLP, LGBM, CAT, LR, SVM, ADA, and RF). Besides, we created a Bagging method [68] training different models (i.e., LR, CAT, DNN, SVM, ADA, LGBM, RF, and XGB) in parallel instead of different instances of the same model. Moreover, in stacking, we use all the previous models with the addition of Voting [69], Averaging [70], and Weighted Averaging.

Hyperparameters: We provide our main hyperparameter choices for each AI model and each ensemble method used in our work in Appendix A.

Having provided the main experimental setup, we next detail our evaluation results and findings.

VI. EVALUATION RESULTS

A. NSL-KDD ANALYSIS

1) MAIN RESULTS OF LEVEL 00

Table 6 shows similar accuracy, precision, recall, and F1 scores for Level 00 using the NSL-KDD dataset. The left-hand side of the table (Table 6) presents the results

TABLE 6. NSL-KDD Level 00: organized by F1 (highest to lowest) when all features are used (i.e., left-hand side). We observe that Bag_DT achieved the best scores overall, followed by KNN. However, when feature selection is applied (i.e., right-hand side). We observe that Bag_DT achieved the best scores overall, followed by KNN.

All Features					Feature Selection				
Models	ACC-00	PRE-00	REC-00	F1-00	Models	ACC-00	PRE-00	REC-00	F1-00
Bag_DT	**0.996**	**0.952**	**0.924**	**0.937**	**Bag_DT**	**0.996**	**0.912**	**0.903**	**0.907**
KNN	0.991	0.957	0.907	0.929	KNN	0.990	0.931	0.884	0.904
Bag_knn	0.991	0.951	0.908	0.927	Bag_knn	0.990	0.929	0.885	0.904
DT	0.995	0.901	0.923	0.923	MLP	0.993	0.897	0.907	0.902
Bag_mlp	0.994	0.947	0.900	0.919	XGB	0.992	0.912	0.891	0.900
XGB	0.993	0.928	0.891	0.906	Bag_lgbm	0.995	0.933	0.879	0.900
MLP	0.992	0.951	0.870	0.904	Bag_comb	0.988	0.939	0.867	0.897
Bag_lgbm	0.996	0.932	0.868	0.893	DT	0.995	0.879	0.896	0.896
CAT	0.992	0.923	0.857	0.882	CAT	0.992	0.927	0.867	0.891
Bag_cat	0.992	0.912	0.848	0.872	Bag_LR	0.978	0.894	0.874	0.883
Bag_LR	0.978	0.906	0.845	0.870	Bag_cat	0.992	0.919	0.859	0.881
Bag_comb	0.987	0.944	0.827	0.869	Bag_mlp	0.993	0.888	0.869	0.877
LR	0.977	0.913	0.836	0.864	LR	0.977	0.883	0.867	0.875
LGBM	0.983	0.794	0.882	0.813	LGBM	0.981	0.806	0.897	0.834
SVM	0.970	0.868	0.781	0.813	SVM	0.969	0.903	0.780	0.820
Bag_svm	0.968	0.875	0.754	0.798	Bag_svm	0.966	0.895	0.769	0.808
Bag_ada	0.786	0.719	0.617	0.646	Bag_ada	0.849	0.740	0.680	0.702
RF	0.942	0.771	0.587	0.622	ADA	0.840	0.684	0.644	0.658
Bag_rf	0.948	0.578	0.575	0.576	RF	0.945	0.771	0.602	0.640
ADA	0.597	0.654	0.584	0.519	Bag_rf	0.948	0.579	0.574	0.576
DNN	0.859	0.341	0.390	0.364	DNN	0.859	0.341	0.390	0.364

when all features of the dataset are utilized, and the right-hand side details the outcomes using the top eight features (shown in Table 22). The use of all features yields slightly higher values for precision, recall, and F1 scores, indicating better overall performance. Notably, the top three performing models, which include Bagging with Decision Trees, KNN, and Bagging with KNN (one base learner and two ensemble methods: Bag_DT and Bag_knn), demonstrate this advantage. Conversely, the models showing the weakest performance are consistent across both setups, namely DNN, Bagging with Random Forest, and ADA. Therefore, using all features of the NSL-KDD dataset proves beneficial, though there is still potential for further improvements by enhancing the performance of ensemble models in Level 01.

2) MAIN RESULTS OF LEVEL 01

Recall that Level 01 refers to the application of base learners and ensemble models to four generated datasets from Level 00. These datasets are based on the models' probability predictions and predicted classes for each sample in Level 00 as follows: The first dataset includes the predicted probabilities for each model in Level 00 using all features. The second dataset contains the predicted probabilities using the top eight features (as per Table 22). Similarly, the third and fourth datasets include the predicted classes using all features and the top eight features, respectively.

Tables 7 and 8 display the results of Level 01 for the four configurations previously mentioned. There are two primary comparisons of interest: one among all Level 01 tables, and the other between Level 00 and Level 01 results. Comparing all Level 01 tables by the F1 score, the highest performance is observed in Table 8, which considers probabilities using all features. This is followed by the same table considering probabilities with feature selection, and then Table 7 with classes using all features, and finally, the same table with feature selection. The results suggest that for the NSL-KDD dataset, models perform better when utilizing prediction probabilities rather than predicted classes, with a further preference for using all features to achieve optimal results. The most effective models across these configurations include Bagging_LGBM, LGBM, and Bagging_DT, with an exception noted for all features in Table 7.

3) COMPARISON OF LEVEL 01 AND LEVEL 00

The comparison between Level 00 and Level 01 provides a clear advantage for Level 01, indicating that stacking significantly enhances model performance for the NSL-KDD dataset. Specifically, when evaluating performance using all features and prediction probabilities from Level 01 (Table 7), eight models–including Bagging-LGBM, LGBM, Bagging-DT, XGB, CAT, and MLP–outperform the best model from Level 00 (Table 6, which features Bagging-DT). The highest F1 scores from Table 7 reach 0.995, compared to 0.937 from

TABLE 7. NSL-KDD Level 01: organized by F1 (highest to lowest). The input dataset from Level 00 is predicted classes and all features on the left-hand side (DT achieved the best scores overall) and predicted classes and feature selection on the right hand side (Bag_lgbm achieved the best scores overall).

All Features					Feature Selection				
Models	ACC-01	PRE-01	REC-01	F1-01	Models	ACC-01	PRE-01	REC-01	F1-01
DT	**0.995**	**0.893**	**0.926**	**0.906**	**DT**	**0.994**	**0.956**	**0.886**	**0.912**
Bag_cat	0.982	0.760	0.763	0.762	Bag_lgbm	0.996	0.890	0.920	0.903
RF	0.982	0.759	0.763	0.761	XGB	0.996	0.890	0.920	0.903
Bag_mlp	0.981	0.758	0.763	0.760	Bag_DT	0.996	0.891	0.918	0.902
XGB	0.981	0.759	0.762	0.760	MLP	0.996	0.889	0.887	0.888
Bag_DT	0.981	0.761	0.759	0.760	LR	0.991	0.987	0.842	0.881
Bag_comb	0.981	0.758	0.761	0.760	Bag_LR	0.991	0.921	0.841	0.869
CAT	0.981	0.758	0.761	0.759	LGBM	0.996	0.871	0.853	0.862
Bag_lgbm	0.981	0.761	0.758	0.759	Bag_comb	0.996	0.871	0.852	0.861
Bag_rf	0.981	0.757	0.761	0.759	RF	0.996	0.854	0.823	0.833
KNN	0.980	0.759	0.758	0.759	Bag_knn	0.996	0.855	0.819	0.831
LGBM	0.981	0.758	0.759	0.758	KNN	0.996	0.855	0.819	0.831
MLP	0.980	0.754	0.762	0.758	Bag_rf	0.996	0.840	0.822	0.829
Bag_knn	0.980	0.757	0.755	0.756	CAT	0.996	0.839	0.821	0.828
Bag_ada	0.978	0.759	0.749	0.754	Bag_cat	0.996	0.838	0.821	0.828
ADA	0.969	0.755	0.729	0.742	weighed_avg	0.990	0.884	0.804	0.827
LR	0.962	0.728	0.720	0.724	VOTING	0.995	0.838	0.819	0.827
Bag_LR	0.961	0.730	0.718	0.723	avg	0.989	0.885	0.801	0.825
VOTING	0.964	0.750	0.687	0.711	Bag_mlp	0.996	0.789	0.788	0.788
Bag_svm	0.933	0.699	0.708	0.701	DNN	0.967	0.551	0.596	0.571
SVM	0.909	0.695	0.647	0.664	Bag_svm	0.899	0.668	0.580	0.568
DNN	0.950	0.543	0.576	0.559	SVM	0.899	0.644	0.580	0.568
avg	0.917	0.697	0.552	0.556	Bag_ada	0.616	0.557	0.467	0.331
weighed_avg	0.914	0.673	0.549	0.556	ADA	0.616	0.600	0.434	0.305

TABLE 8. NSL-KDD Level 01: organized by F1 (highest to lowest). The input dataset from Level 00 is prediction probabilities and all features on the left-hand side, and prediction probabilities and feature selection on the right-hand side. In both cases, DT achieved the best scores overall.

All Features					Feature Selection				
Models	ACC-01	PRE-01	REC-01	F1-01	Models	ACC-01	PRE-01	REC-01	F1-01
Bag_lgbm	**0.996**	**0.994**	**0.995**	**0.995**	**DT**	**0.988**	**0.966**	**0.966**	**0.966**
LGBM	0.996	0.993	0.994	0.994	XGB	0.985	0.953	0.963	0.958
Bag_DT	0.992	0.991	0.990	0.991	Bag_DT	0.991	0.945	0.944	0.944
XGB	0.984	0.982	0.990	0.986	Bag_lgbm	0.990	0.923	0.928	0.925
CAT	0.986	0.980	0.963	0.970	CAT	0.986	0.928	0.924	0.925
MLP	0.965	0.933	0.944	0.938	KNN	0.984	0.946	0.899	0.918
Bag_cat	0.985	0.978	0.905	0.930	Bag_knn	0.985	0.925	0.904	0.913
DT	0.987	0.948	0.906	0.924	Bag_cat	0.985	0.923	0.903	0.911
Bag_mlp	0.964	0.926	0.915	0.919	Bag_mlp	0.931	0.840	0.846	0.840
KNN	0.975	0.918	0.883	0.896	LGBM	0.972	0.817	0.899	0.836
Bag_comb	0.969	0.964	0.839	0.861	MLP	0.939	0.856	0.823	0.832
Bag_knn	0.975	0.858	0.857	0.856	Bag_comb	0.947	0.937	0.775	0.781
Bag_rf	0.957	0.737	0.778	0.755	Bag_rf	0.967	0.727	0.770	0.746
RF	0.943	0.737	0.751	0.742	RF	0.959	0.725	0.763	0.742
Bag_LR	0.832	0.652	0.669	0.658	LR	0.878	0.644	0.671	0.654
LR	0.830	0.651	0.667	0.657	Bag_LR	0.877	0.641	0.667	0.651
Bag_svm	0.807	0.632	0.660	0.636	Bag_svm	0.842	0.614	0.636	0.618
SVM	0.791	0.630	0.651	0.626	SVM	0.839	0.618	0.633	0.618
ADA	0.744	0.800	0.610	0.618	ADA	0.612	0.756	0.623	0.594
Bag_ada	0.560	0.800	0.612	0.557	DNN	0.599	0.248	0.393	0.303
DNN	0.608	0.369	0.391	0.319	Bag_ada	0.028	0.455	0.320	0.207
weighed_avg	0.366	0.136	0.201	0.115	avg	0.358	0.095	0.197	0.110
avg	0.365	0.128	0.200	0.114	weighed_avg	0.363	0.131	0.200	0.108

TABLE 9. NSL-KDD timetable in seconds showing how long each model takes in training and testing combined. Overall, the best results are from RF, Average, and Weighted Average. On the other hand, the worst models in terms of runtime are Bag_KNN, Bag_LGBM, and Bag_MLP.

Models	All_feat_00	feat_Sel_00	Class_All_feat_01	Class_feat_Sel_01	Prob_All_feat_01	Prob_feat_Sel_01
RF	**0.493**	**0.493**	0.313	0.318	0.652	0.321
DT	1.485	1.693	0.078	0.076	0.256	0.137
LGBM	2.237	2.765	195.688	451.964	0.789	462.938
CAT	2.947	3.036	0.493	0.547	1.215	0.796
LR	4.940	9.039	1.489	1.229	1.109	1.225
Bag_rf	5.748	8.134	3.456	2.977	3.904	3.186
ADA	8.409	8.385	0.580	0.594	1.320	0.917
XGB	9.370	73.908	47.697	60.510	1.875	62.612
Bag_DT	10.892	19.873	0.188	0.117	1.001	0.556
Bag_lgbm	21.101	368.351	4.331	3.588	2173.16	741.341
Bag_cat	28.360	28.730	4.456	5.013	11.611	6.456
SVM	34.594	35.065	0.258	0.292	0.305	0.211
Bag_LR	51.671	831.574	10.579	10.956	10.654	10.326
Bag_ada	64.872	80.312	4.442	4.423	9.338	6.376
DNN	67.233	50.928	20.874	9.006	61.612	9.054
Bag_svm	79.710	95.955	2.323	2.127	2.331	2.014
MLP	119.158	107.966	12.863	9.024	28.477	33.225
Bag_comb	563.579	748.475	82.313	115.755	317.177	529.088
KNN	758.682	919.402	3.165	2.587	1.547	0.429
Bag_mlp	921.355	1214.870	121.176	94.681	299.985	416.981
Bag_knn	4046.57	5142.25	28.654	22.914	13.626	1.614
average	NA	NA	**0.063**	0.089	**0.063**	**0.095**
weighed_avg	NA	NA	0.065	**0.074**	0.068	0.105
voting	NA	NA	0.272	0.329	NA	NA

the best result in Table 6. Additionally, in the scenario with feature selection and prediction probabilities (Table 8), models such as XGB and Bagging_DT also exhibit F1 scores surpassing 0.937. These findings confirm that, for the NSL-KDD dataset, utilizing prediction probabilities rather than predicted classes in Level 01 leads to superior outcomes.

4) RUNTIME PERFORMANCE

Another consideration is the time added for getting better predictions. Table 9 displays the time in seconds for each model in each level variant in our work. This table is important because one can check how long it takes to obtain a better result. Considering the NSL-KDD dataset, recall that the best result came from the Bagging_LGBM ensemble method, and the second best came from LGBM. Both models are from the case with all features and prediction probabilities (Table 8) for Level 01. However, when considering the time it takes to run both models, Table 9 shows that LGBM (0.789 seconds) is quicker than Bagging_LGBM (2173.16 seconds) by roughly 2172 seconds. Overall, the best runtime results are from RF, Average, and Weighted Average. On the other hand, the worst models in terms of runtime are Bag_KNN, Bag_LGBM, and Bag_MLP. From this analysis, the choice between the best models (by the security analyst) is problem-dependent which depends on time, accuracy, precision, recall, and F1.

We next show the main evaluation results for the RoEduNet-SIMARGL2021 dataset.

B. ROEDUNET-SIMARGL2021 ANALYSIS

1) MAIN RESULTS OF LEVEL 00

Table 10 display similar accuracy, precision, recall, and F1 scores for Level 00 using the RoEduNet-SIMARGL2021 dataset. The table elucidates the results from experiments using all features, as well as those using the top eight features (according to Table 22). The complete feature set yields slightly higher values for precision, recall, and F1 scores, leading to better overall performance. Notably, the top three models (e.g., Bagging_DT, DT, and LGBM), comprising two base learners and one ensemble method, perform best. Conversely, the weakest performances are consistent across both feature sets, including models such as Bagging_ADA, Bagging_LR, Bagging_SVM, and ADA. Despite achieving near-perfect metrics of 0.999 in many models, we proceeded to analyze the impact of using Level 01 variants with the RoEduNet-SIMARGL2021 dataset to determine if performance could be enhanced or compromised.

2) MAIN RESULTS OF LEVEL 01

Tables 11 and 12 showcase the main results of Level 01 for the RoEduNet-SIMARGL2021 dataset, reflecting the four configurations previously mentioned. The analysis involves two primary comparisons: one among all Level 01 tables, and another between the results of Level 00 and Level 01. Initially, when assessing all Level 01 tables by the F1 score, the highest results are observed in Table 12, which accounts for prediction probabilities using all features. This is followed by prediction probabilities with feature selection,

TABLE 10. RoEduNet-SIMARGL2021 Level 00: organized by F1 (highest to lowest) with all features used (left-hand side). Note that DT, Bag_DT, LGBM, CAT, Bag_CAT, RF, Bag_Comb, Bag_RF, and Bag_lgbm) yield the best performance metrics. In the case of Feature Selection(right-hand side) Bag_DT, DT, LGBM, CAT, Bag_CAT, Bag_LGBM, KNN, and Bag_KNN have the best performance metrics.

All Features					Feature Selection				
Models	ACC-00	PRE-00	REC-00	F1-00	Models	ACC-00	PRE-00	REC-00	F1-00
DT	**0.999**	**0.999**	**0.999**	**0.999**	**Bag_DT**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_DT	**0.999**	**0.999**	**0.999**	**0.999**	**DT**	**0.996**	**0.996**	**0.996**	**0.996**
LGBM	**0.999**	**0.999**	**0.999**	**0.999**	**LGBM**	**0.996**	**0.996**	**0.996**	**0.996**
CAT	**0.999**	**0.999**	**0.999**	**0.999**	**CAT**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_cat	**0.999**	**0.999**	**0.999**	**0.999**	**Bag_cat**	**0.996**	**0.996**	**0.996**	**0.996**
RF	**0.999**	**0.999**	**0.999**	**0.999**	**Bag_lgbm**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_comb	**0.999**	**0.999**	**0.999**	**0.999**	**KNN**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_rf	**0.999**	**0.999**	**0.999**	**0.999**	**Bag_KNN**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_lgbm	**0.999**	**0.999**	**0.999**	**0.999**	Bag_comb	0.995	0.995	0.995	0.995
KNN	0.998	0.998	0.998	0.998	XGB	0.995	0.995	0.995	0.995
XGB	0.998	0.998	0.998	0.998	Bag_rf	0.995	0.995	0.995	0.995
Bag_knn	0.998	0.998	0.998	0.998	RF	0.993	0.993	0.993	0.993
DNN	0.943	0.949	0.943	0.943	MLP	0.991	0.991	0.991	0.991
ADA	0.913	0.923	0.913	0.909	bag_MLP	0.991	0.991	0.991	0.991
SVM	0.862	0.878	0.862	0.862	DNN	0.985	0.985	0.985	0.985
Bag_svm	0.859	0.877	0.859	0.859	LR	0.984	0.984	0.984	0.984
LR	0.854	0.868	0.854	0.854	SVM	0.983	0.984	0.983	0.983
Bag_LR	0.854	0.868	0.854	0.854	Bag_LR	0.983	0.983	0.983	0.983
Bag_ada	0.584	0.461	0.584	0.501	Bag_svm	0.975	0.976	0.975	0.975
mlp	0.333	0.111	0.333	0.167	ADA	0.625	0.501	0.625	0.537

TABLE 11. RoEduNet-SIMARGL2021 Level 01: organized by F1 (highest to lowest). The input dataset from Level 00 is predicted classes and all features on the left-hand side. We note that the majority of models yield identical performances for this setup. On the right-hand side, the input dataset from Level 00 is predicted classes and feature selection. We observe that all models perform very well except ADA, and Bag_ADA.

All Features					Feature Selection				
Models	ACC-01	PRE-01	REC-01	F1-01	Models	ACC-01	PRE-01	REC-01	F1-01
Bag_DT	**0.996**	**0.996**	**0.996**	**0.996**	**KNN**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_lgbm	**0.996**	**0.996**	**0.996**	**0.996**	**LGBM**	**0.996**	**0.996**	**0.996**	**0.996**
ADA	**0.996**	**0.996**	**0.996**	**0.996**	**RF**	**0.996**	**0.996**	**0.996**	**0.996**
LGBM	**0.996**	**0.996**	**0.996**	**0.996**	**Bag_knn**	**0.996**	**0.996**	**0.996**	**0.996**
RF	**0.996**	**0.996**	**0.996**	**0.996**	**Bag_DT**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_ada	**0.996**	**0.996**	**0.996**	**0.996**	**Bag_rf**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_comb	**0.996**	**0.996**	**0.996**	**0.996**	**Bag_lgbm**	**0.996**	**0.996**	**0.996**	**0.996**
MLP	**0.996**	**0.996**	**0.996**	**0.996**	**Bag_comb**	**0.996**	**0.996**	**0.996**	**0.996**
CAT	**0.996**	**0.996**	**0.996**	**0.996**	**MLP**	**0.996**	**0.996**	**0.996**	**0.996**
XGB	**0.996**	**0.996**	**0.996**	**0.996**	**CAT**	**0.996**	**0.996**	**0.996**	**0.996**
LR	**0.996**	**0.996**	**0.996**	**0.996**	**XGB**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_LR	**0.996**	**0.996**	**0.996**	**0.996**	**LR**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_mlp	**0.996**	**0.996**	**0.996**	**0.996**	**Bag_LR**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_cat	**0.996**	**0.996**	**0.996**	**0.996**	**Bag_mlp**	**0.996**	**0.996**	**0.996**	**0.996**
KNN	**0.996**	**0.996**	**0.996**	**0.996**	**Bag_cat**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_knn	**0.996**	**0.996**	**0.996**	**0.996**	**DNN**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_rf	**0.996**	**0.996**	**0.996**	**0.996**	**VOTING**	**0.996**	**0.996**	**0.996**	**0.996**
DNN	**0.996**	**0.996**	**0.996**	**0.996**	**avg**	**0.996**	**0.996**	**0.996**	**0.996**
weighed_avg	**0.996**	**0.996**	**0.996**	**0.996**	**SVM**	**0.996**	**0.996**	**0.996**	**0.996**
SVM	**0.996**	**0.996**	**0.996**	**0.996**	**Bag_svm**	**0.996**	**0.996**	**0.996**	**0.996**
Bag_svm	**0.996**	**0.996**	**0.996**	**0.996**	**DT**	**0.996**	**0.996**	**0.996**	**0.996**
DT	**0.996**	**0.996**	**0.996**	**0.996**	weighed_avg	0.995	0.995	0.995	0.995
avg	0.995	0.996	0.995	0.995	ADA	0.335	0.745	0.336	0.172
VOTING	0.995	0.995	0.995	0.995	Bag_ada	0.333	0.739	0.334	0.168

predicted classes with all features, and classes with feature selection (all in Table 11). Hence, models demonstrate better performance when utilizing probabilities rather than class predictions, with minimal difference between using all features versus selected features. The top performers across these configurations include Bagging_LGBM and

TABLE 12. RoEduNet-SIMARGL2021 Level 01: organized by F1 (highest to lowest). The input dataset from Level 00 is prediction probabilities and all features on the left-hand side. Here, all models performed very well except the Avg and Weigted_Avg methods. On the right-hand side, the input dataset from Level 00 is predicted probabilities and feature selection. We observe that all models perform very well except ADA, and Bag_ADA.

All Features					Feature Selection				
Models	ACC-01	PRE-01	REC-01	F1-01	Models	ACC-01	PRE-01	REC-01	F1-01
CAT	**1.000**	**1.000**	**1.000**	**1.000**	**LGBM**	**1.000**	**1.000**	**1.000**	**1.000**
XGB	**1.000**	**1.000**	**1.000**	**1.000**	**Bag_lgbm**	**1.000**	**1.000**	**1.000**	**1.000**
LGBM	**1.000**	**1.000**	**1.000**	**1.000**	**Bag_comb**	**1.000**	**1.000**	**1.000**	**1.000**
Bag_DT	**1.000**	**1.000**	**1.000**	**1.000**	KNN	0.999	0.999	0.999	0.999
Bag_lgbm	**1.000**	**1.000**	**1.000**	**1.000**	Bag_knn	0.999	0.999	0.999	0.999
Bag_cat	**1.000**	**1.000**	**1.000**	**1.000**	Bag_DT	0.999	0.999	0.999	0.999
KNN	0.999	0.999	0.999	0.999	Bag_cat	0.999	0.999	0.999	0.999
Bag_knn	0.999	0.999	0.999	0.999	XGB	0.999	0.999	0.999	0.999
RF	0.999	0.999	0.999	0.999	MLP	0.999	0.999	0.999	0.999
Bag_comb	0.999	0.999	0.999	0.999	CAT	0.999	0.999	0.999	0.999
Bag_rf	0.999	0.999	0.999	0.999	Bag_mlp	0.999	0.999	0.999	0.999
ADA	0.999	0.999	0.999	0.999	Bag_rf	0.999	0.999	0.999	0.999
Bag_ada	0.999	0.999	0.999	0.999	Bag_ada	0.999	0.999	0.999	0.999
Bag_mlp	0.999	0.999	0.999	0.999	ADA	0.999	0.999	0.999	0.999
MLP	0.999	0.999	0.999	0.999	RF	0.999	0.999	0.999	0.999
DT	0.999	0.999	0.999	0.999	DT	0.999	0.999	0.999	0.999
Bag_LR	0.961	0.965	0.961	0.961	SVM	0.692	0.774	0.691	0.606
LR	0.961	0.965	0.961	0.961	LR	0.684	0.772	0.684	0.587
DNN	0.947	0.954	0.947	0.947	Bag_LR	0.684	0.772	0.684	0.587
SVM	0.927	0.938	0.926	0.926	Bag_svm	0.526	0.473	0.526	0.463
Bag_svm	0.926	0.938	0.926	0.925	DNN	0.673	0.603	0.504	0.420
avg	0.329	0.112	0.330	0.165	weighed_avg	0.329	0.112	0.330	0.165
weighed_avg	0.329	0.111	0.330	0.165	avg	0.329	0.111	0.330	0.165

LGBM, although many models exhibit nearly perfect performance. Notably, ensemble learning methods generally outperform base learners, as evident from the data in Tables 11 and 12.

3) COMPARISON OF LEVEL 01 AND LEVEL 00

Regarding the comparison between Level 00 and Level 01, the results indicate that stacking may be unnecessary, as the outcomes from Level 01 mirror those from the best configurations of Level 00 for the RoEduNet-SIMARGL2021 dataset. To better assess the impact of stacking, we analyzed improvements across all models: seven out of sixteen models showed better performance in the best configuration of Level 01 (Table 12) compared to Level 00 (Table 10). However, only three out of sixteen models improved in the second-best Level 01 configuration compared to the best Level 00 setup. Although the tables containing predicted classes from Level 01 (Table 11) indicate enhanced performance for some models, they also show diminished results for others. Consequently, the majority of base learners do not substantially benefit from the transition to Level 01. In summary, it is advisable to maintain the analysis at Level 00, as there is no significant advantage to progressing to Level 01 with this dataset.

4) RUNTIME PERFORMANCE

We next show the runtime amount for each model. Table 13 shows the time in seconds for each model in each level variant. Considering the RoEduNet-SIMARGL2021 dataset, it is best to consider the result from Level 00 since it achieved the best results. Therefore, among the models with a 0.999 score in all metrics (i.e., DT, Bagging_DT, LGBM, CAT, Bagging_CAT, RF, Bagging_Comb, and Bagging_RF), it is wise to choose the quickest, which is RF with roughly 9 seconds followed by LGBM with 27 seconds. The slowest model is the Bagging_comb with a duration of 40 minutes. Also, note that this dataset is large (e.g., approximately 30 million samples), and we limited the number of samples to 6 million for our experiments using random sampling.

C. CICIDS-2017 ANALYSIS

1) MAIN RESULTS OF LEVEL 00

Tables 14 present the accuracy, precision, recall, and F1 scores for Level 00 using the CICIDS-2017 dataset, detailing experiments with all features and the top eight features (according to Table 22). The complete feature set delivers slightly higher precision, recall, and F1 scores, indicating better overall performance. The top performing models include Bagging_DT, DT, and Bagging_LGBM, with a mix of base and ensemble methods showing strong results. Conversely, models such as Bagging_ADA, DNN, SVM, and ADA underperform in both setups. Given that nine models have an F1 score below 0.9, there is clearly potential for improvement. Thus, we proceeded to analyze the performance of ensemble models at Level 01 to determine their effectiveness with this dataset.

TABLE 13. RoEduNet-SIMARGL2021 timetable in seconds, how long each model takes in training and testing combined.

Models	All_feat_00	feat_Sel_00	Class_All_feat_01	Class_feat_Sel_01	Prob_All_feat_01	Prob_feat_Sel_01
RF	**9.314**	**4.520**	4.845	2.358	3.492	3.064
DT	46.169	13.456	2.568	1.799	2.547	2.491
LGBM	26.822	19.467	642.364	140.609	287.318	342.337
CAT	52.058	57.649	36.011	35.476	23.090	11.218
LR	114.472	120.463	47.447	47.311	34.858	29.068
Bag_rf	80.395	58.802	14.298	13.831	24.183	17.936
ADA	232.106	67.153	43.283	19.574	68.522	36.514
XGB	233.771	149.593	124.672	133.681	109.726	65.475
Bag_DT	300.164	74.900	8.185	4.476	10.547	16.942
Bag_lgbm	283.994	221.295	47.334	39.470	2062.180	45.363
Bag_cat	417.715	373.420	102.928	106.750	215.549	149.089
SVM	450.796	275.876	9.923	7.089	8.324	10.196
Bag_LR	1029.020	1365.330	269.193	253.114	366.542	254.403
Bag_ada	2659.840	993.793	191.792	165.788	343.843	262.717
DNN	1986.220	916.598	456.373	189.507	319.007	158.293
Bag_svm	315.692	321.503	57.499	53.517	58.906	99.091
MLP	1122.573	661.901	155.997	81.413	217.626	300.410
Bag_comb	2398.550	1776.020	15515.900	6415.180	3644.920	3402.890
KNN	1026.3657	53881.918	15406.000	10641.100	2917.630	2095.380
Bag_mlp	9358.280	5876.632	1163.390	808.948	1839.620	3324.450
Bag_knn	372504.328	9939.241	106268.000	64846.300	29783.200	27281.600
average	NA	NA	**1.512**	1.386	1.317	1.432
weighed_avg	NA	NA	1.529	**1.377**	**1.305**	**1.407**
voting	NA	NA	9.718	8.852	NA	NA

TABLE 14. CICIDS-2017 Level 00: organized by F1 (highest to lowest) and all features are used (left hand side). We note that 13 models have an F-1 score higher than 0.9 while 8 models have an F-1 score lower than 0.9 for this setup. In the case of feature selection (right hand side), we note that 12 models have an F-1 score higher than 0.9 while 9 models have an F-1 score lower than 0.9 for this setup.

All Features					Feature Selection				
Models	ACC-00	PRE-00	REC-00	F1-00	Models	ACC-00	PRE-00	REC-00	F1-00
Bag_lgbm	**0.999**	**0.995**	**0.999**	**0.997**	**Bag_lgbm**	**0.999**	**0.995**	**0.999**	**0.997**
DT	0.999	0.995	0.997	0.997	Bag_DT	0.999	0.995	0.998	0.997
Bag_DT	0.999	0.995	0.998	0.996	DT	0.999	0.995	0.997	0.997
Bag_cat	0.997	0.992	0.987	0.990	XGB	0.997	0.992	0.988	0.990
CAT	0.997	0.992	0.986	0.989	Bag_cat	0.997	0.993	0.987	0.990
XGB	0.997	0.991	0.986	0.988	CAT	0.997	0.993	0.987	0.990
KNN	0.992	0.979	0.994	0.986	Bag_knn	0.992	0.980	0.994	0.987
Bag_knn	0.992	0.979	0.994	0.986	KNN	0.992	0.980	0.994	0.987
Bag_mlp	0.982	0.945	0.946	0.943	Bag_comb	0.994	0.993	0.963	0.977
MLP	0.982	0.950	0.933	0.938	Bag_mlp	0.983	0.947	0.948	0.945
LGBM	0.987	0.940	0.926	0.933	MLP	0.982	0.949	0.941	0.942
Bag_comb	0.989	0.989	0.876	0.923	LGBM	0.965	0.853	0.938	0.864
RF	0.963	0.562	0.521	0.537	RF	0.959	0.703	0.518	0.536
Bag_LR	0.931	0.519	0.525	0.520	Bag_LR	0.931	0.518	0.526	0.520
LR	0.929	0.519	0.522	0.518	LR	0.928	0.517	0.525	0.519
Bag_rf	0.957	0.560	0.483	0.506	Bag_rf	0.956	0.560	0.483	0.506
Bag_svm	0.909	0.374	0.410	0.390	Bag_svm	0.910	0.376	0.410	0.391
SVM	0.909	0.374	0.410	0.390	SVM	0.910	0.376	0.410	0.391
DNN	0.873	0.377	0.375	0.370	DNN	0.856	0.354	0.288	0.288
Bag_ada	0.797	0.250	0.235	0.236	ADA	0.794	0.274	0.235	0.239
ADA	0.796	0.252	0.234	0.235	Bag_ada	0.794	0.274	0.235	0.239

2) MAIN RESULTS OF LEVEL 01

Tables 16 and 15 showcase the main results of Level 01 for the CICIDS-2017 dataset, comparing two configurations: one using all features and another using the top eight features. The analysis involves two primary comparisons: one among all Level 01 tables, and another between the results of Level 00 and Level 01. The highest F1 scores are observed in Table 15, which considers prediction probabilities with all

TABLE 15. CICIDS-2017 Level 01: organized by F1, highest to lowest. On the left-hand side, the input dataset from Level 00 is predicted probabilities, and all features are considered. This setup gives the best results for the CICIDS-2017 dataset. On the right-hand side, the input dataset from Level 00 is predicted probabilities, and feature selection is considered. This setup gives the second-best results for the CICIDS-2017 dataset.

All Features					Feature Selection				
Models	ACC-01	PRE-01	REC-01	F1-01	Models	ACC-01	PRE-01	REC-01	F1-01
XGB	**1.000**	**1.000**	**1.000**	**1.000**	**Bag_lgbm**	**1.000**	**1.000**	**1.000**	**1.000**
LGBM	**1.000**	**1.000**	**1.000**	**1.000**	Bag_DT	0.999	0.999	0.999	0.999
Bag_DT	**1.000**	**1.000**	**1.000**	**1.000**	LGBM	0.999	0.999	0.999	0.999
Bag_lgbm	**1.000**	**1.000**	**1.000**	**1.000**	XGB	0.999	0.999	0.999	0.999
DT	0.999	0.999	0.998	0.998	Bag_cat	0.999	0.999	0.999	0.999
CAT	0.999	0.999	0.998	0.998	CAT	0.999	0.999	0.998	0.999
Bag_cat	0.999	0.998	0.996	0.997	DT	0.999	0.999	0.998	0.999
Bag_comb	0.999	0.993	0.992	0.992	Bag_knn	0.999	0.995	0.995	0.995
Bag_knn	0.996	0.978	0.981	0.979	KNN	0.999	0.995	0.995	0.995
MLP	0.996	0.979	0.980	0.979	Bag_comb	0.999	0.995	0.994	0.994
KNN	0.996	0.977	0.980	0.978	Bag_mlp	0.999	0.989	0.989	0.989
Bag_mlp	0.996	0.978	0.978	0.978	MLP	0.999	0.988	0.987	0.987
Bag_rf	0.998	0.978	0.978	0.977	RF	0.998	0.976	0.980	0.978
RF	0.998	0.975	0.970	0.971	Bag_rf	0.998	0.974	0.977	0.975
Bag_LR	0.873	0.869	0.849	0.855	Bag_LR	0.978	0.853	0.828	0.826
LR	0.869	0.857	0.790	0.813	LR	0.978	0.851	0.826	0.824
Bag_ada	0.666	0.826	0.786	0.793	Bag_ada	0.779	0.747	0.847	0.770
Bag_svm	0.814	0.768	0.657	0.660	Bag_svm	0.920	0.561	0.531	0.539
SVM	0.794	0.677	0.639	0.639	SVM	0.923	0.560	0.529	0.537
ADA	0.597	0.469	0.537	0.386	ADA	0.764	0.569	0.577	0.466
DNN	0.747	0.211	0.283	0.242	DNN	0.942	0.367	0.388	0.377
avg	0.220	0.040	0.143	0.053	weighed_avg	0.217	0.037	0.142	0.055
weighed_avg	0.220	0.045	0.143	0.052	avg	0.220	0.036	0.143	0.054

features, followed by the same table using feature selection. The results from Table 16 with feature selection and all features show that models perform better when utilizing probabilities rather than class predictions, with minimal difference between using all features versus selected features for achieving the best results. The top performers across these configurations include Bagging_LGBM, LGBM, XGB, and Bagging_DT, with several other models like DT, CAT, and Bagging_CAT also displaying near-perfect performance. Notably, ensemble learning methods generally outperform base learners, as evidenced by the data in Tables 16 and 15.

3) COMPARISON OF LEVEL 01 AND LEVEL 00

Regarding the comparison between Level 00 and Level 01 for the CICIDS-2017 dataset, the results indicate that stacking at Level 01 is beneficial, as it generally outperforms Level 00. Specifically, 17 out of 21 models show improvement in the best configuration of Level 01 (Table 15) compared to the best of Level 00 (Table 14), and all models show improvement in the second-best configuration of Level 01. Additionally, the results from Table 16, which focus on predicted classes, demonstrate better performance for some models compared to their counterparts at Level 00, although a few models do not perform as well. The analysis supports the recommendation to employ Level 01 ensemble learning strategies for this dataset, as most models exhibit a notable performance gain.

4) RUNTIME PERFORMANCE

We next show the runtime amount for each model for the CICIDS-2017 dataset. Table 17 shows the time in seconds for each model in each level variant. Considering the CICIDS-2017 dataset, it is best to consider the result from Level 00 and Level 01 (i.e., probabilities and all features) since it achieved the best results. Therefore, among the Level 00 models with the 0.997 score in F1 metrics (i.e., DT, Bagging_DT, and Bagging_LGBM), it is wise to choose the quickest, which is DT with roughly one minute followed by Bagging_DT with around seven minutes. The slowest model is the Bagging_KNN, with a duration of nine hours and 20 minutes approximately. Regarding Level 01 Probabilities and all features among the Level 01 models with the 1.000 score in all metrics (i.e., XGB, Bagging_DT, LGBM, and Bagging_LGBM), it is wise to choose the quickest, which is Bagging_DT with roughly one and a half minute. The slowest model is the LGBM, with a duration of 15 minutes. As a side note, this dataset is large (e.g., approximately 3 million samples with 78 features). Thus, the time analysis gives insight into the runtime complexity of each model considered here.

D. FALSE POSITIVE RATE (FPR)

We next display the results for False Positive Rates for our two-level framework for the three considered datasets.

TABLE 16. CICIDS-2017 Level 01: organized by F1 (highest to lowest). On the left-hand side, the input dataset from Level 00 is predicted classes, and all features are considered. On the right-hand side, the input dataset from Level 00 is predicted classes, and feature selection is applied.

All Features					Feature Selection				
Models	ACC-01	PRE-01	REC-01	F1-01	Models	ACC-01	PRE-01	REC-01	F1-01
DT	**0.998**	**0.990**	**0.994**	**0.992**	**XGB**	**0.999**	**0.996**	**0.998**	**0.997**
Bag_DT	0.999	0.990	0.993	0.991	**LGBM**	**0.999**	**0.996**	**0.998**	**0.997**
XGB	0.999	0.992	0.990	0.991	**Bag_lgbm**	**0.999**	**0.996**	**0.998**	**0.997**
LGBM	0.999	0.991	0.991	0.991	**CAT**	**0.999**	**0.996**	**0.998**	**0.997**
Bag_comb	0.998	0.991	0.990	0.991	Bag_cat	0.999	0.996	0.997	0.997
MLP	0.998	0.988	0.993	0.991	Bag_DT	0.999	0.996	0.998	0.997
Bag_lgbm	0.999	0.989	0.992	0.990	Bag_comb	0.999	0.996	0.997	0.996
Bag_rf	0.998	0.990	0.991	0.990	Bag_rf	0.999	0.995	0.998	0.996
CAT	0.998	0.990	0.991	0.990	Bag_mlp	0.999	0.996	0.997	0.996
Bag_cat	0.998	0.990	0.991	0.990	MLP	0.999	0.995	0.997	0.996
RF	0.998	0.989	0.991	0.990	RF	0.999	0.995	0.997	0.996
Bag_mlp	0.998	0.988	0.991	0.989	KNN	0.999	0.995	0.996	0.995
Bag_knn	0.998	0.986	0.990	0.988	Bag_knn	0.999	0.994	0.997	0.995
KNN	0.998	0.984	0.989	0.986	VOTING	0.999	0.995	0.993	0.994
Bag_LR	0.994	0.963	0.931	0.945	DT	0.998	0.986	0.996	0.991
LR	0.994	0.961	0.928	0.943	weighed_avg	0.988	0.878	0.980	0.917
VOTING	0.986	0.986	0.810	0.838	Bag_LR	0.990	0.873	0.843	0.842
Bag_svm	0.979	0.858	0.808	0.819	avg	0.979	0.798	0.936	0.842
ADA	0.910	0.784	0.801	0.776	LR	0.990	0.834	0.840	0.837
Bag_ada	0.902	0.813	0.716	0.698	Bag_ada	0.686	0.574	0.703	0.571
SVM	0.963	0.574	0.510	0.521	ADA	0.679	0.433	0.570	0.434
weighed_avg	0.917	0.493	0.577	0.505	SVM	0.879	0.376	0.289	0.264
DNN	0.967	0.421	0.428	0.424	Bag_svm	0.879	0.368	0.289	0.263
avg	0.771	0.261	0.370	0.277	DNN	0.877	0.236	0.286	0.257

TABLE 17. CICIDS-2017 timetable in seconds, how long each model takes in training and testing combined.

Models	All_feat_00	feat_Sel_00	Class_All_feat_01	Class_feat_Sel_01	Prob_All_feat_01	Prob_feat_Sel_01
RF	**3.293**	**4.655**	0.356	0.787	0.347	0.880
DT	58.719	50.045	**0.053**	**0.031**	0.189	0.115
LGBM	22.930	46.739	614.118	126.836	914.519	444.575
CAT	35.048	34.993	1.018	5.727	8.039	4.924
LR	196.571	124.619	2.266	13.469	11.264	12.701
Bag_rf	31.400	35.561	4.046	5.963	4.201	6.619
ADA	243.058	452.856	1.137	6.186	5.068	9.791
XGB	182.262	236.288	77.373	8.973	326.647	34.766
Bag_DT	429.327	399.648	0.286	1.234	1.483	2.106
Bag_lgbm	2747.17	15531.100	8.059	29.973	8.817	20.878
Bag_cat	326.218	557.323	9.271	55.924	12.062	46.992
SVM	148.869	213.330	0.648	3.592	0.521	1.881
Bag_LR	1087.750	1345.780	21.997	133.123	24.752	118.946
Bag_ada	1567.900	1878.410	9.104	56.788	17.451	78.002
DNN	318.268	169.189	44.338	72.035	17.050	129.836
Bag_svm	485.063	621.683	8.811	37.487	4.092	15.903
MLP	862.157	2025.750	15.083	33.392	144.713	242.389
Bag_comb	5899.980	6874.490	187.794	340.418	197.654	447.121
KNN	4223.340	5307.190	12.452	645.616	20.3749	20.531
Bag_mlp	13872.800	15789.100	111.08	342.938	783.752	2386.950
Bag_knn	28514.900	33048.500	117.288	3792.240	83.353	181.463
average	NA	NA	0.098	0.417	**0.084**	**0.381**
weighed_avg	NA	NA	0.114	0.418	0.096	0.383
voting	NA	NA	0.454	2.62764	NA	NA

(i) FPR Comparison among Level 00: Feature Selection vs. All Features: We begin by analyzing the False Positive Rate (FPR) for Level 00, comparing both configurations: using all features and feature selection. The FPR results,

TABLE 18. False Positive Rates (%) - Level 00 all features. We notice that the effectiveness of feature selection may vary depending on the specific network intrusion dataset and model configuration.

	All Features			Feature Selection		
Models	CICIDS-2017	NSL-KDD	RoEduNet-SIMARGL2021	CICIDS-2017	NSL-KDD	RoEduNet-SIMARGL2021
DT	**0.492**	**6.443**	**0.001**	1.055	12.118	0.377
LR	60.269	**10.293**	13.177	**47.632**	11.684	**1.608**
SVM	47.319	14.737	12.237	**46.213**	**11.574**	**1.635**
DNN	31.359	**5.829**	5.053	**19.157**	54.768	**1.467**
MLP	6.989	**8.076**	22.228	**5.403**	27.904	**1.161**
KNN	**1.875**	**8.007**	**0.295**	3.383	15.854	0.391
ADA	**42.717**	**36.839**	**7.703**	42.724	47.972	49.861
LGBM	**23.432**	**19.525**	**0.001**	41.768	23.553	0.379
CAT	**0.683**	8.394	**0.020**	1.069	**6.467**	0.389
XGB	**0.978**	**9.509**	0.024	1.280	28.649	**0.492**
RF	43.327	**3.003**	**0.052**	**28.727**	7.715	0.669
Bag_RF	**28.701**	2.224	**0.071**	28.731	**2.142**	0.508
Bag_DT	0.527	**5.451**	**0.001**	**0.501**	8.852	0.376
Bag_LR	**48.105**	**10.572**	**13.227**	48.193	10.644	**1.688**
Bag_SVM	47.278	13.181	12.336	**47.089**	**10.509**	**2.410**
Bag_MLP	5.654	**5.026**	55.561	**5.287**	11.203	**0.848**
Bag_KNN	**1.880**	8.180	**0.293**	2.013	**7.096**	0.390
Bag_ADA	**12.970**	35.459	**53.880**	26.384	**25.949**	49.848
Bag_LGBM	0.684	8.015	**0.004**	**0.460**	**6.730**	0.383
Bag_CAT	**0.676**	**7.581**	**0.021**	0.714	8.080	0.392
Bag_Comb	1.234	**5.115**	**0.060**	**0.721**	6.069	0.480

TABLE 19. False Positive Rates (%) - Level 01 Prediction Probabilities. Level 01 yields lower FPR compared to Level 00 for most cases for all of the three datasets considered in our work.

	All Features			Feature Selection		
Models	CICIDS-2017	NSL-KDD	RoEduNet-SIMARGL2021	CICIDS-2017	NSL-KDD	RoEduNet-SIMARGL2021
DT	0.083	5.120	0.001	**0.028**	**3.315**	0.001
LR	**14.278**	34.666	**5.928**	14.898	**16.201**	23.208
SVM	51.901	38.206	**6.127**	**43.695**	**37.458**	23.513
DNN	**14.590**	**9.866**	**3.868**	20.486	11.862	19.099
MLP	2.118	11.193	**0.029**	**1.207**	**9.008**	0.037
KNN	2.272	4.790	**0.018**	**0.496**	**2.474**	0.037
ADA	**54.185**	49.626	4.066	77.653	**36.646**	**0.003**
LGBM	**0.000**	20.823	**0.000**	0.012	**1.021**	0.003
CAT	**0.087**	**1.780**	**0.000**	0.103	2.277	0.003
XGB	**0.000**	**1.695**	**0.000**	0.014	2.652	0.007
RF	2.509	7.581	**0.000**	**2.366**	**7.008**	0.007
Bag_RF	**2.245**	6.420	**0.003**	2.595	**6.024**	0.007
Bag_DT	**0.000**	4.066	0.001	0.003	**2.891**	0.001
Bag_LR	**13.103**	34.624	**5.933**	14.668	**16.188**	23.208
Bag_SVM	**23.160**	**18.305**	**6.220**	43.516	38.181	25.510
Bag_MLP	2.207	10.574	0.014	**1.123**	**10.438**	**0.001**
Bag_KNN	2.230	25.021	**0.018**	**0.523**	**2.658**	0.037
Bag_ADA	**17.361**	50.702	4.060	81.014	**32.683**	**3.514**
Bag_LGBM	0.000	**1.002**	0.003	0.000	1.803	0.003
Bag_CAT	0.235	**1.940**	0.003	**0.064**	2.276	0.003
Bag_Comb	0.750	**2.884**	**0.003**	**0.538**	5.239	0.022

presented in Tables 18 for all features and feature selection, vary across datasets. For the NSL-KDD dataset, feature selection resulted in a decreased FPR for 7 out of 21 models, indicating that it is beneficial for a minority of models. In contrast, for the CICIDS-2017 dataset, 11 out of 21 models showed a decrease in FPR with feature selection, suggesting a more substantial benefit. Similarly, for the RoEduNet-SIMARGL2021 dataset, 8 out of 21 models experienced a reduction in FPR due to feature selection. These results imply that the effectiveness of feature selection may vary depending on the specific network intrusion dataset and model configuration.

(ii) FPR Comparison among Level 01: Feature Selection vs. All Features: The FPR tables for Level 01, both with all features and feature selection, exhibit variable impacts across datasets, as shown in Table 19. For the

NSL-KDD dataset, feature selection reduced the FPR for 14 out of 21 models, underscoring its utility. In the CICIDS-2017 dataset, feature selection was beneficial for 10 out of 21 models due to a decreased FPR. However, for the RoEduNet-SIMARGL2021 dataset, only 2 out of 21 models saw a reduction in FPR with feature selection, suggesting limited effectiveness. Given these results, the general recommendation for Level 01 is not to apply feature selection, aligning with earlier recommendations based on F1 scores. This consistency confirms the strategy's relevance across different evaluation metrics considered in our work.

(iii) FPR Comparison between Level 00 and Level 01:

a. All Features: For CICIDS-2017, 16 out of 21 models benefit from going to Level 01 (i.e., they have lower FPR). For NSL-KDD, 8 out of 21 models benefit from going to level 01. For RoEduNet-SIMARGL2021, 19 out of 21 models benefit from going to Level 01.

b. Feature Selection: For CICIDS-2017, 18 out of 21 models benefit from going to Level 01. For NSL-KDD, 15 out of 21 models benefit from going to level 01 For RoEduNet-SIMARGL2021, 16 out of 21 models benefit from going to Level 01.

FPR Summary: The best setup based on FPR results is to stay at Level 00 - All features for the NSL-KDD. For CICIDS-2017, it is Level 01 - All probabilities and all features, followed closely by Level 01 - Feature Selection. For RoEduNet-SIMARGL2021, the best setup is Level 01 - All probabilities and all features. Thus, overall, it is clear that our two-level framework provides a gain in false positive rates compared to just applying Level 00.

E. STATISTICAL ANALYSIS

1) STATISTICAL ANALYSIS

We next provide statistical analysis to validate our findings. Hence, we chose to apply the non-parametric Wilcoxon signed-rank statistical test to the three datasets used in this work. The rationale behind our choice is that our datasets do not have a normal distribution. For our statistical analysis, we pick 12 models (i.e., SVM, RF, DNN, LGBM, MLP, KNN, LR, DT, CAT, XGB, bag_DT, and ADA) that include the models with the best results overall and other relevant models. Considering 12 models, it will generate 66 p-values for each pair combination of models for each dataset (i.e., 66 rows in Table 20). Then, we apply the significance test using k-fold cross-validation and the accuracy scores for each model. This gives the full process to generate model pairs that will be used for statistical analysis.

The result of the above process generates two samples, one for each model (for the two models being compared against each other). Later, we applied the Wilcoxon test to obtain the difference between both samples. If they differ from each other in a significant way, one can imply that there is significant evidence that one model is more accurate than the other. The algorithm ranks which differences have the highest absolute difference value. Finally, we check whether the calculated p-values are lower than 0.05 (significance level) to reject the null hypothesis (i.e., there is no difference in the performance of the two models being compared). Our statistical analysis results are shown in Table 20.

2) MAIN INSIGHTS

Considering the three datasets, Table 20 shows that most pair models reject the null hypothesis ($p < 0.05$). Therefore, there is evidence that one model performs better than the other in most cases. Thus, we can separate the tests into two groups: The group with statistical significance (i.e., $p < 0.05$), and the group that does not have statistical significance ($p > 0.05$). We can further divide the first group into two subgroups by analyzing the medians of their array of accuracy scores of the estimator for each run of the cross-validation method. The first subgroup, indeed, has a model that is performing better (i.e., one model has a higher median than the other one). For this case, we highlighted the best model in bold in Table 20. However, for the second subgroup (i.e., both models display the same median), we cannot know which is better. In this case, we highlighted both models in bold in Table 20. Although the majority of methods are significant (i.e., SVM, RF, DNN, LGBM, MLP, KNN, LR, DT, CAT, XGB, bag_DT, and ADA), the following pair methods for the CICIDS-2017 are not: SVM-LGBM, RF-LGBM, LGBM-MLP, LGBM-KNN, LGBM-LR, LR-ADA (i.e., 6 out of 66). In contrast, DNN, KNN, DT, bag_DT, CAT, and XGB are significant for all comparisons. When analyzing the RoEduNet-SIMARGL2021 dataset, the following pair methods are non-significant: SVM-LR, RF-KNN, RF-DT, LGBM-KNN, LGBM-DT, LGBM-Bag_DT, MLP-LR, KNN-DT, KNN-Bag_DT, DT-CAT, DT-Bag_DT, bag_DT-CAT (i.e., 12 out of 66). In contrast, DNN, ADA, and XGB are significant for all comparisons. Lastly, for the NSL-KDD, the only non-significant pair method is SVM-LGBM (i.e., one out of 66). Therefore, RF, DNN, MLP, KNN, LR, DT, CAT, XGB, bag_DT, and ADA are statistically significant for all comparisons. Across all datasets, XGB and DNN are always significant.

F. GAINS OF OUR FRAMEWORK - DATASET-WISE

Next, we break the quantitative gains down by each dataset.

1) NSL-KDD

By implementing the Level 01 Probabilities - All Features, and comparing the best model in Level 00 to the best model in Level 01, we have an enhancement in its Precision from 0.952 to 0.996, its Recall from 0.924 to 0.995, and its F1 from 0.937 to 0.995. Also, it decreased its FPR from 2.142 % (best in level 00) to 1.002% in Level 01. On the other hand, considering the best base learner model in Level 00, it enhanced its Accuracy from 0.991 to 0.996, its Precision from 0.957 to 0.996, its Recall from 0.907 to 0.995, and its F1 from 0.929 to 0.995. Also, it decreased its FPR from 5.829 % to 1.002%.

TABLE 20. The pairwise statistical test results between every pair of AI models by Wilcoxon signed rank test. Statistically better method ($p < 0.05$) shown in bold (both marked bold if there is significance and the median of the accuracies are the same while only one is marked bold if there is significance and one model has a higher median). On the left, the CICIDS-2017 dataset is shown. In the middle, the RoEduNet-SIMARGL2021 dataset is shown. On the right, the NSL-KDD dataset is shown. For all datasets, our method is predominantly statistically significant with a few exceptions where $p > 0.05$. Across all datasets, XGB and DNN are always statistically significant.

CICIDS-2017		
Model 1	Model 2	p-value
SVM	**RF**	0.00013
SVM	**DNN**	1.9073×10^{-6}
SVM	LGBM	0.43043
SVM	**MLP**	1.9073×10^{-6}
SVM	**KNN**	1.9073×10^{-6}
SVM	LR	0.04566
SVM	**DT**	1.9073×10^{-6}
SVM	**CAT**	1.9073×10^{-6}
SVM	**XGB**	0.001953125
SVM	**bag_DT**	1.9073×10^{-6}
SVM	ADA	0.02395
RF	**DNN**	1.9073×10^{-6}
RF	LGBM	0.694047
RF	MLP	3.81469×10^{-6}
RF	**KNN**	3.81469×10^{-6}
RF	LR	0.021484
RF	**DT**	3.81469×10^{-6}
RF	**CAT**	1.9073×10^{-6}
RF	**XGB**	0.001953125
RF	**bag_DT**	1.9073×10^{-6}
RF	ADA	1.9073×10^{-6}
DNN	**LGBM**	0.001953125
DNN	**MLP**	1.9073×10^{-6}
DNN	**KNN**	1.9073×10^{-6}
DNN	**LR**	1.9073×10^{-6}
DNN	**DT**	1.9073×10^{-6}
DNN	**CAT**	1.9073×10^{-6}
DNN	**XGB**	0.001953125
DNN	**bag_DT**	1.9073×10^{-6}
DNN	**ADA**	1.9073×10^{-6}
LGBM	MLP	0.28786
LGBM	KNN	0.09225
LGBM	LR	0.42038
LGBM	**DT**	0.0019531
LGBM	**CAT**	0.0019531
LGBM	**XGB**	0.001953125
LGBM	**bag_DT**	0.0019531
LGBM	**ADA**	0.0019531
MLP	**KNN**	1.9073×10^{-6}
MLP	LR	0.023950
MLP	**DT**	1.9073×10^{-6}
MLP	**CAT**	1.9073×10^{-6}
MLP	**XGB**	0.001953125
MLP	**bag_DT**	1.9073×10^{-6}
MLP	ADA	3.8146×10^{-6}
KNN	LR	0.0239505
KNN	**DT**	1.9073×10^{-6}
KNN	**CAT**	2.6702×10^{-5}
KNN	**XGB**	0.001953125
KNN	**bag_DT**	1.9073×10^{-6}
KNN	ADA	1.9073×10^{-6}
LR	**DT**	0.008308
LR	**CAT**	0.0094356
LR	**XGB**	0.001953125
LR	**bag_DT**	0.0083084
LR	ADA	0.89831
DT	CAT	4.7375×10^{-4}
DT	**XGB**	0.001953125
DT	bag_DT	1.9073×10^{-6}
DT	ADA	1.9073×10^{-6}
CAT	**XGB**	0.001953125
CAT	**bag_DT**	7.2387×10^{-4}
CAT	ADA	1.9073×10^{-6}
XGB	**bag_DT**	0.001953125
XGB	**ADA**	0.001953125
bag_DT	ADA	1.9073×10^{-6}

RoEduNet-SIMARGL2021		
Model 1	Model 2	p-value
SVM	**RF**	1.9073×10^{-6}
SVM	**DNN**	1.9073×10^{-6}
SVM	**LGBM**	1.9073×10^{-6}
SVM	**MLP**	1.9073×10^{-6}
SVM	**KNN**	1.9073×10^{-6}
SVM	LR	0.206021
SVM	**DT**	1.9073×10^{-6}
SVM	**CAT**	1.9073×10^{-6}
SVM	**XGB**	0.001953125
SVM	**bag_DT**	1.9073×10^{-6}
SVM	ADA	1.9073×10^{-6}
RF	**DNN**	1.9073×10^{-6}
RF	**LGBM**	0.00036373
RF	MLP	0.0001834
RF	KNN	0.182600
RF	LR	1.9073×10^{-6}
RF	DT	0.050403
RF	**CAT**	0.01147
RF	**XGB**	0.001953125
RF	**bag_DT**	0.01147
RF	ADA	1.9073×10^{-6}
DNN	**LGBM**	0.001953125
DNN	**MLP**	1.9073×10^{-6}
DNN	**KNN**	1.9073×10^{-6}
DNN	**LR**	1.9073×10^{-6}
DNN	**DT**	1.9073×10^{-6}
DNN	**CAT**	1.9073×10^{-6}
DNN	**XGB**	0.001953125
DNN	**bag_DT**	1.9073×10^{-6}
DNN	**ADA**	1.9073×10^{-6}
LGBM	MLP	0.000104083
LGBM	KNN	0.29681813
LGBM	LR	0.00195312
LGBM	DT	0.560444
LGBM	CAT	0.00588927
LGBM	**XGB**	0.001953125
LGBM	bag_DT	0.153006
LGBM	ADA	0.001953125
MLP	**KNN**	0.001953125
MLP	LR	0.0768
MLP	**DT**	0.00046781
MLP	**CAT**	0.000182742
MLP	**XGB**	0.001953125
MLP	**bag_DT**	0.00032646
MLP	ADA	0.0010074
KNN	LR	0.0001727
KNN	DT	0.05548
KNN	**CAT**	0.0061161
KNN	**XGB**	0.001953125
KNN	bag_DT	0.093775
KNN	ADA	5.7220×10^{-6}
LR	**DT**	0.000183696
LR	**CAT**	0.0001349
LR	**XGB**	0.001953125
LR	**bag_DT**	0.00018345
LR	ADA	0.001821
DT	CAT	0.26261
DT	**XGB**	0.001953125
DT	bag_DT	0.47412
DT	ADA	0.0001112
CAT	**XGB**	0.001953125
CAT	bag_DT	0.629505
CAT	ADA	0.0001391
XGB	**bag_DT**	0.001953125
XGB	**ADA**	0.001953125
bag_DT	ADA	0.000111

NSL-KDD		
Method 1	Method 2	p-value
SVM	RF	1.9073×10^{-6}
SVM	**DNN**	1.9073×10^{-6}
SVM	LGBM	0.20244
SVM	**MLP**	1.9073×10^{-6}
SVM	**KNN**	0.001953125
SVM	**LR**	1.9073×10^{-6}
SVM	**DT**	1.9073×10^{-6}
SVM	**CAT**	1.9073×10^{-6}
SVM	**XGB**	0.001953125
SVM	**bag_DT**	1.9073×10^{-6}
SVM	ADA	1.9073×10^{-6}
RF	**DNN**	1.9073×10^{-6}
RF	**LGBM**	0.001953125
RF	**MLP**	1.9073×10^{-6}
RF	**KNN**	0.001953125
RF	**LR**	1.9073×10^{-6}
RF	**DT**	1.9073×10^{-6}
RF	**CAT**	1.9073×10^{-6}
RF	**XGB**	0.001953125
RF	**bag_DT**	1.9073×10^{-6}
RF	ADA	1.9073×10^{-6}
DNN	**LGBM**	0.001953125
DNN	**MLP**	1.9073×10^{-6}
DNN	**KNN**	0.001953125
DNN	**LR**	1.9073×10^{-6}
DNN	**DT**	1.9073×10^{-6}
DNN	**CAT**	1.9073×10^{-6}
DNN	**XGB**	0.001953125
DNN	**bag_DT**	1.9073×10^{-6}
DNN	**ADA**	1.9073×10^{-6}
LGBM	**MLP**	0.001953125
LGBM	**KNN**	0.001953125
LGBM	**LR**	0.001953125
LGBM	**DT**	0.001953125
LGBM	**CAT**	0.001953125
LGBM	**XGB**	0.001953125
LGBM	**bag_DT**	0.001953125
LGBM	**ADA**	0.001953125
MLP	KNN	0.001953125
MLP	**LR**	1.9073×10^{-6}
MLP	**DT**	0.001953125
MLP	**CAT**	0.001953125
MLP	**XGB**	0.001953125
MLP	**bag_DT**	0.001953125
MLP	ADA	0.001953125
KNN	LR	0.00588927
KNN	**DT**	0.001953125
KNN	**CAT**	0.001953125
KNN	**XGB**	0.001953125
KNN	**bag_DT**	0.001953125
KNN	ADA	0.001953125
LR	**DT**	1.9073×10^{-6}
LR	**CAT**	1.9073×10^{-6}
LR	**XGB**	0.001953125
LR	**bag_DT**	1.9073×10^{-6}
LR	ADA	1.9073×10^{-6}
DT	CAT	1.9073×10^{-6}
DT	**XGB**	0.001953125
DT	**bag_DT**	0.01231
DT	ADA	1.9073×10^{-6}
CAT	**XGB**	0.001953125
CAT	**bag_DT**	1.9073×10^{-6}
CAT	ADA	1.9073×10^{-6}
XGB	**bag_DT**	0.001953125
XGB	**ADA**	0.001953125
bag_DT	ADA	1.9073×10^{-6}

2) CICIDS-2017

By implementing the Level 01 Probabilities - All Features, comparing the best model in Level 00 to the best model in Level 01, enhanced its Accuracy from 0.999 to 1.000, its Precision from 0.995 to 1.000, its Recall from 0.999 to 1.000, and its F1 from 0.997 to 1.000. Also, it decreased its FPR from 0.460 % (best in Level 00) to 0.002% in Level 01. On the other hand, considering the best model in Level 00 is also a base learner to the best ensemble learning method, it enhanced its Accuracy from 0.999 to 1.000, enhanced its Precision from 0.995 to 1.000, its Recall from 0.999 to 1.000, and its F1 from 0.999 to 1.000. Also, it decreased its FPR from 0.460 % to 0.002%. The overlap in results is due to DT being among the best models for Level 00.

3) ROEDUNET-SIMARGL2021

By implementing the Level 01 Probabilities - All Features, comparing the best model in Level 00 to the best model in Level 01, enhanced its Accuracy from 0.999 to 1.000, its Precision from 0.999 to 1.000, its Recall from 0.999 to 1.000,

TABLE 21. The summary of all results in this work. Our framework provides higher performance metrics and lower FPR.

Higher Metrics (ACC, PRE, REC, F1)	CICIDS-2017	NSL-KDD	RoEduNet-SIMARGL2021	Overall
Best Setup	LV01 (ALL Features & Probabilities) LV01 (Feature Selection & Probabilities)	LV01 (ALL Features & Probabilities)	LV01 (ALL Features & Probabilities) LV01 (Feature Selection & Probabilities) LV00 (ALL Features & Probabilities)	LV01 (ALL Features & Probabilities)
Best models	Bag_LGBM, LGBM, Bag_DT, XGB	Bag_LGBM, LGBM, Bag_DT, XGB	CAT, LGBM, Bag_DT, XGB Bag_LGBM, Bag_CAT	Bag_LGBM, LGBM, Bag_DT, XGB
Lower FPR (%)	**CICIDS-2017**	**NSL-KDD**	**RoEduNet-SIMARGL2021**	**Overall**
Best Setup	LV01 (ALL Features & Probabilities) LV01 (Feature Selection & Probabilities)	LV01 (ALL Features & Probabilities)	LV01 (ALL Features & Probabilities) LV01 (Feature Selection & Probabilities) LV00 (ALL Features & Probabilities)	LV01 (ALL Features & Probabilities)
Best models	Bag_LGBM, LGBM, Bag_DT, XGB	Bag_LGBM, Bag_CAT, CAT, XGB	CAT, LGBM, RF, XGB	CAT, LGBM, Bag_LGBM, XGB
Runtime	**CICIDS-2017**	**NSL-KDD**	**RoEduNet-SIMARGL2021**	**Overall**
Fastest models (Less than 10 min in all level variants)	RF, DT, CAT,LR Bag_RF, ADA, XGB Bag_DT, Bag_Cat SVM,DNN Avg, Weighed Avg,Voting	RF,DT,LGBM,CAT LR, Bag_RF, ADA,XGB, Bag_DT,Bag_CAT,SVM Bag_ADA, DNN,Bag_SVM, MLP Avg, Weighed Avg Voting	RF, DT, CAT, LR Bag_rf, ADA, XGB, Bag_DT,Bag_LGBM, Bag_CAT SVM, Avg, Weighed Avg Voting	RF, DT, CAT, LR Bag_rf, ADA, XGB, Bag_DT, Bag_CAT SVM, Avg, Weighed Avg Voting
Average Models (between 10 and 100 min in all level variants)	KNN, MLP,LGBM Bag_SVM,Bag_ADA, Bag_LR	Bag_KNN, Bag_MLP, Bag_Comb,Bag_LGBM, KNN, Bag_LR	KNN, DNN, MLP, Bag_SVM,Bag_ADA, Bag_LR	KNN, MLP, Bag_LR, LGBM
Slowest models (More than 100 min in all level variants)	Bag_KNN, Bag_MLP, Bag_Comb,Bag_LGBM	-	Bag_KNN, Bag_MLP, Bag_Comb,Bag_LGBM	Bag_KNN, Bag_MLP, Bag_Comb,Bag_LGBM

and its F1 from 0.999 to 1.000. Also, it decreased its FPR from 0.001 % (best in level 00) to 0.000% in Level 01. On the other hand, considering the best base learner model in Level 00 to the best ensemble learning method, it enhanced its Accuracy from 0.999 to 1.000, enhanced its Precision from 0.995 to 1.000, its Recall from 0.997 to 1.000, and its F1 from 0.997 to 1.000. Also, it decreased its FPR from 0.001 % to 0.000 %. Again, we emphasize that the overlap in results is due to DT being among the best models for Level 00.

Overall, our extensive evaluation has shown the prospective benefits of our proposed two-level ensemble learning framework, detailing each metric for each dataset considered in this work.

VII. LIMITATIONS, DISCUSSION, AND FUTURE DIRECTIONS

A. DISCUSSION

1) IMPORTANCE OF OUR TWO-LEVEL FRAMEWORK

In today's world of exponential information growth, it is natural that network attacks will become more frequent (e.g., see the recent study by the Center for Strategic & International Studies (CSIS) [71]). Although IDS has evolved throughout the years, security analysts are responsible for double-checking possible attack occurrences in this fast-paced environment. Therefore, having an accurate framework for intrusion detection systems can help solve such a problem by having fewer FPR instances to analyze and focus on the crucial traffic data. Our framework presented in this work helps to reduce the FPR and to increase performance (in terms of accuracy, recall, precision, and F1) in intrusion detection systems thoroughly, which is a key point to deploy them for network security.

2) SUMMARY OF FINDINGS

All the results in this paper are summarized in Table 21. It answers the questions the reader might have. Which AI model is the best, or is it context-dependent?; Is the two-level framework beneficial after all?; Is it viable? Table 21 shows three crucial points (i.e., Metrics used, FPR, and Runtime) in analyzing the framework.

a: PERFORMANCE METRICS

The first takes into consideration the metrics (i.e., Accuracy, Precision, Recall, and F1) used to analyze all possible setups (i.e., Level 00 All features, Level 00 Feature selection, Level 01 Classes - All features, Level 01 Classes - Feature Selection, Level 01 Probabilities - All features, and Level 01 Probabilities - Feature Selection). Overall, the best results (i.e., higher metrics) are achieved when using Level 01 Probabilities - All features and the best models are XGB, LGBM, Bag_DT, and Bag_LGBM.

b: FALSE POSITIVE RATE

The second part is analyzing which setup provided the lower FPR. The results show that, once more Level 01 Probabilities - All features provided the lowest FPR overall. The

models with the lowest FPR are CAT, LGBM, XGB, and Bag_LGBM.

c: RUNTIME

The third part consists of analyzing the runtime of the 21 models (plus voting, averaging, and weighted averaging) used. Table 21 divides them by the fastest models (runtime < 10 min), slowest models (runtime > 100 min), and average models (10 min < runtime < 100 min). To fit models in each one of the three categories, the model runtime analysis has to be valid for all Level variants (i.e., Level 00 All features, Level 00 Feature selection, Level 01 Classes - All features, Level 01 Classes - Feature Selection, Level 01 Probabilities - All features, and Level 01 Probabilities - Feature Selection). In other words, if at least one of the variants has a runtime higher than the requisite, the model will fall in the next slower category. From this analysis, the following models are the overall fastest ones: RF, DT, CAT, LR, Bag_rf, ADA, XGB, Bag_DT, Bag_CAT, SVM, Avg, Weighed Avg, and Voting. The following have an average runtime: KNN, MLP, Bag_LR, and LGBM. The Slowest models are Bag_KNN, Bag_MLP, Bag_Comb, and Bag_LGBM.

d: BEST MODELS

When overlapping these results (performance metrics, FPR, and Runtime) in Table 21, we can extract the best models overall, which are XGB and Bag_DT, followed by LGBM due to a slower runtime. These models were the best in achieving the metrics proposed in this work, and they achieved their best performance using the Level 01 Probabilities - All features setup. These suggestions are viable since the time added to its head time is kept under a few minutes when using the fastest models.

3) INSIGHTS ABOUT BOOSTING TECHNIQUES

It is interesting that the best techniques in Table 21 are all boosting techniques (except ADA). This shows its importance in enhancing AI-based intrusion detection models, particularly using two-level ensemble learning. This might be an indication that Boosting techniques are a good fit for the ensemble in Level 01 of our framework (or other stacking methods). This might be due to the correlation between the way that boosting works (learning from past mistakes) and the relationship between the probabilities and classes. This shows the suitability of using boosting in Level 01 in our two-level framework for IDS.

4) DECISION TREE CONSIDERATION

The results show that Decision Tree is one of the best models for all three datasets, and the only base model to rival ensemble methods in Level 00. However, it is most likely to lack generality when applied in different scenarios due to its tendency to overfit and bias [72]. Therefore, checking the next best base learner performances reassures the relevance of using ensemble learning for improving the metrics. As an example, for CICIDS-2017, the best base learner excluding DT is KNN which achieved an Accuracy of 0.992, Precision of 0.979, Recall of 0.994, and F1 of 0.986. For RoEduNet-SIMARGL2021, the next best base learner is KNN which achieved an Accuracy of 0.998, Precision of 0.998, Recall of 0.998, and F1 of 0.998. For NSL-KDD, KNN performs better than DT, however, this dataset has characteristics that make it less prone to over-fitting.

5) BENEFITS OF OUR TWO-LEVEL FRAMEWORK

Our two-level framework offers flexibility in the way it is built. Differently from previous works, it goes beyond applying ensemble learning techniques directly to the datasets, it uses the predictions obtained to generate four new meta-datasets based on classes and prediction probabilities which is applied to all the models used in Level 00 and ensemble learning techniques. This setup provides an extensive evaluation of the three datasets used, among them the RoEduNet-SIMARGL2021 which is considered real-world and not yet evaluated in this context from the literature gathered to the best of our knowledge. Moreover, it permits the use of feature selection techniques at both levels, enriching the pool of results and insights gathered. Moreover, this work presents an extensive evaluation that is not present in other works, we analyze 21 models/ensemble methods in 6 different setups (two Level 00 setups and four Level 01 setups) in three different datasets, generating results for Accuracy, Precision, Recall, F1, Runtime, False Positive Rates, and a Statistical Significance test. Plus, we achieve perfect and near-perfect results for a few models considering the FPR and the F1 score, both metrics are particularly crucial for IDS. This is because security analysts, stakeholders, and users need to do everything in their power to identify a possible threat accurately and as fastest as possible as undetected attacks can cause significant damage.

We also want to stress that we took the extra step and made the codes open source. The way they are built is to be easily expandable to use with other datasets and further analysis. As a side note, we want to express that when we say Framework, we are specifically meaning that our program is an end-to-end solution delineated by our low-level framework (Figure 2) that starts at the datasets, then processing all the inner work and extracting the results. However, it is not a deployable solution for production since it is not extensively tested or validated by an auditory company, but instead, a proof of concept of the benefits of using our proposed framework and a crucial stepping-stone in enhancing the field of AI-based network IDS.

B. LIMITATIONS

1) DATASET ANALYSIS AND BIAS

The experiments allow us to extract a few interesting conclusions about the datasets used. First, we conclude that the models in our framework achieved better overall gains in Level 01 for NSL-KDD when compared to

CICIDS-2017, and RoEduNet-SIMARGL2021. Each one of these datasets has unique characteristics. For instance, RoEduNet-SIMARGL2021 has almost 30 Million data points with roughly 20 feature columns, and CICIDS-2017 has almost 2 Million data points but roughly 70 feature columns 5. This explains the perspective of why it can become heavy on the models increasing their runtime for these datasets. Another implication is the amount of data and the number of classes present, three prediction classes for RoEduNet-SIMARGL2021 and seven for CICIDS-2017. In the case of RoEduNet-SIMARGL2021, the AI models seem to learn its patterns easily with due to its huge size and only three prediction classes. However, the CICIDS-2017 has seven prediction classes, and the models seem to adapt well and achieve high scores without much issue. The reason behind it might be its high unbalanced number of data points between classes 5 (four classes combined represent less than 1% of the whole dataset). Differently, NSL-KDD represents a more heterogeneous dataset with overall fewer samples than the other two and with better balance among classes. It could be the reason behind the more plausible metrics for NSL-KDD in contrast to the other two datasets. This limitation indicates the need for new testing in future work with different datasets or uncalibrated models to expand benchmarking and testing for our two-level framework.

2) BAGGING METHOD EFFICACY FOR SOME VARIANTS

Another insight is that bagging efficacy and viability are dependent. For instance, most experiments in this paper show that bag_KNN and bag_MLP showed almost identical results compared to their single counterparts, which indicates that is not worth performing bagging with the following models (KNN, MLP, SVM, LR) due to the added time and little improvement offered. Also, the same happened when bagging other ensemble methods (e.g., RF, ADA, CAT, LGBM) except decision tree (Bagging_DT). Also, RF itself is a Bagging method that is also fast. The reason behind it might be related to the quality of the datasets used, which did not offer much room for improving the metrics, and the already good performance of base models and ensemble learning methods.

3) TIME PERFORMANCE VARIATIONS AMONG ENSEMBLE METHODS AND AI MODELS

Table 21 shows the summary of Tables 17, 13, and 9. In other words, a simplified version divides the models into three categories (i.e., fast, average, and slow). As expected, some ensemble techniques (such as Bag_knn and Bag_mlp) require great amounts of time, while tree-based and regression models tend to be faster, including RF, DT, and LR. By delving further into the model's behavior, there are a few interesting behaviors, such as the huge runtime drop (e.g., some drops are 20X-30X) for KNN, MLP, SVM, DNN, and its variants when applied to Level 01 in contrast to Level 00. This behavior can be due to the new dataset size (30% its original size) when transitioning from Level 00 to Level 01, its reduced number of column features, and learning from the features (e.g., the information might be laid out in a simpler way to learn, given the model's inner works).

Another interesting relationship in the runtime tables (Tables 17, 13, and 9) is regarding the Boosting methods. All boosting methods, but LGBM, have a faster runtime in Level 01 when compared to Level 00. The reason is probably similar to KNN, MLP, and SVM. However, the LGBM case might be related to the model parameters.

Overall, the models vary from a few milliseconds to many hours. Among the 21 models, 13 models are in the fastest category (under 10 minutes), four models are in the average category (10 - 100 minutes), and four models are in the slowest category (over 100 minutes). Since 17 models are not in the slowest category, the scenario is beneficial for ensemble model combinations without adding excessive headtime.

C. FUTURE DIRECTIONS

We believe this work is a stepping-stone towards an enhanced AI-based IDS. However, there are future experiments and implementations to further enhance our framework. One is to expand our framework to other datasets, other AI models, and more ensemble methods to create an even more comprehensive framework and extract more insights. Another improvement is to find other problems to further validate the benefits of the framework stacking Levels. One other fruitful direction of future research is considering multi-level ensemble learning (i.e., using Level 02 or even Level 03). Moreover, another research direction is to do more experimentation with more feature selection methods. Furthermore, exploring the application of XAI frameworks to generate explanations for the ensemble methods is one feasible future direction for related research. Finally, one ultimate goal of improvement includes implementation in real-time and validations with security experts and analysts to gather valuable insights and improvements.

VIII. CONCLUSION

The objective of security intrusion detection tools is to prevent intrusions, with AI offering automation potential in these tools. The increasing occurrence of intrusions in networked systems has prompted research into AI techniques for intrusion detection systems (IDS). Different AI models have been utilized for automating network intrusion detection tasks, each with its own merits and drawbacks. However, choosing the most appropriate model for a specific dataset can be difficult. To overcome this challenge, combining multiple AI models can improve their overall effectiveness and suitability for network intrusion detection.

This paper aims to bridge this gap by evaluating diverse ensemble methods for IDS. Specifically, we introduced a two-level ensemble learning framework for assessing these methods in network intrusion detection tasks. In the first level, we trained base learners and ensemble methods, generating evaluation metrics and new datasets for the

second level. The second level involved training ensemble methods on the datasets generated in the first level and producing evaluation metrics. Additionally, feature selection has been considered for both levels, utilizing explainable AI (XAI) and Information Gain-based techniques. Results have been presented for 24 ensemble model combinations in our framework, employing various bagging, stacking, and boosting methods on different base learners. Evaluation has been conducted on three network intrusion datasets with varying characteristics. Our analysis has categorized AI models based on their performance metrics (accuracy, precision, recall, and F1), statistical analysis, runtime, and false positive rates, demonstrating the benefits of two-level learning across most setups.

Furthermore, we provide our source codes to the community, offering a baseline two-level ensemble learning framework for network intrusion detection and building on it with new models and datasets. Moreover, we provided insights related to the best models for each dataset, and common and different behaviors among them that are related to their performance and results. We then provided an in-depth discussion about our main findings and main benefits of our framework. We believe that this study marks progress in bridging the divide in utilizing ensemble learning methods for network intrusion detection systems. It achieves this by conducting comprehensive evaluations and comparisons of various metrics to assess the effectiveness of these ensemble methods. Future research directions include testing our framework with other datasets and more ensemble methods, considering multi-level ensemble learning, and exploring the application of XAI frameworks to generate explanations for the ensemble methods.

APPENDIX A
AI MODELS AND HYPER-PARAMETERS

We now provide the hyperparameters of our different AI models considered in this work.

A. AI MODELS AND HYPERPARAMETERS DETAILS

1) BASE MODELS

First, we lay the main details of base models, which are given as follows.

a: DEEP NEURAL NETWORK (DNN)

The first classifier is a deep neural network (DNN). The architecture of this classifier consists of an input layer with the count of neurons corresponding to the used number of features, and the rectified linear unit (ReLU) activation function. This is followed by a dropout layer with the dropout set at 0.01, a hidden layer with a size of 16 neurons, and the ReLU activation function. The setup closes with a ''softmax''. The loss function was set to the ''categorical_crossentropy'' method, while the chosen optimization algorithm was adaptive momentum (ADAM). Eleven epochs were needed to train the model with a batch size of 1024.

b: RANDOM FOREST (RF)

The next classifier used to detect malicious samples in the network traffic was the RandomForest (RF). The hyperparameters we used for this classifier are as follows: n_estimators (a parameter that signifies number of trees used) was set to the value of 100, the maximum tree depth was set to the value of 10, the minimum number of samples required to separate internal node was set to the value of 2, and the rest were used as provided by default.

c: ADABOOST (ADA)

The next classifier used in this study was AdaBoost. The parameter configuration for this classifier is as follows: the maximum number of estimators at which boosting will be completed was set to a value of 50, the weight applied to each classifier in each boosting iteration was set to a value of 1, and the base estimator from which the boosted ensemble is built was set to Decision_Tree_Classifier.

d: LOGISTIC REGRESSION (LR)

The next classifier is Logistic Regression. The parameter configuration for this classifier is as follows: default.

e: DECISION TREE (DT)

The next classifier is Decision Tree. The parameter configuration for this classifier is as follows: default.

f: K-NEAREST NEIGHBOUR (KNN)

We also used the KNN classifier. For this study, we used KNN in its default hyperparameters as follows: the 'n' neighbors value was set to five, all weights were uniform, and the search algorithm was set to 'auto'.

g: SUPPORT VECTOR MACHINE (SVM)

We also used the SVM classifier. The parameter configuration for this classifier is as follows: Kernel is set to 'linear', gamma to 0.5, the probability is set to 'True', and regularization is set to 0.5.

h: MULTI-LAYER PERCEPTRON (MLP)

The next used classifier was MLP. We used the same setup as DNN for this classifier.

2) ENSEMBLE METHODS

Second, we lay out the main details of ensemble methods.

a: LIGHTGBM (LGBM)

The next used classifier was LightGBM. We used it in its default hyper-parameters as follows: the n_splits was set to 10, n_repeats was set to 3, error_score was set to 'raise', n_jobs set to 1, and scoring was set to 'accuracy'.

TABLE 22. List of top significant features by baselines. The works [37] and [38] are used for RoEduNet-SIMARGL2021, CICIDS-2017, respectively. [38] is also used for the NSL-KDD dataset.

Importance	CICIDS-2017	RoEduNet-SIMARGL2021	NSL-KDD
1	Packet Length Std	FLOW_DURATION_MS	same_srv_rate
2	Total length of Bwd Packets	FIRST_SWITCHED	flag_SF
3	Subflow Backward Bytes	TOTAL_FLOWS_EXP	flag_S0
4	Destination Port	TCP_WIN_MSS_IN	dst_host_srv_serror_rate
5	Packet Length Variance	LAST_SWITCHED	dst_host_serror_rate
6	Bwd Packet Length Mean	TCP_WIN_MAX_IN	serror_rate
7	Avg Bwd Segment Size	TCP_WIN_MIN_IN	srv_serror_rate
8	Init_Win_Bytes_Backward	TCP_WIN_MIN_OUT	dst_host_same_srv_rate

b: EXTREME GRADIENT BOOST (XGB)

The next classifier is XGB. The parameter configuration for this classifier is as follows: The number of iterations is set to 100, weight is set to 1, depth is set to 3, learning rate to 0.1, loss function to multi: softmax, and custom metric to mlogloss.

c: CATBOOST (CAT)

The next classifier used in this study was Catboost. The parameter configuration for this classifier is as follows: The number of iterations is set to 100, the depth is set to 6, the learning rate is set to 0.1, and the loss function is set to MultiClass with the custom metric set to Accuracy.

d: VOTING

The next classifier is Voting. This method is a simple stacking method that checks each model decision in Level 01 and reaches its decision by frequency analysis.

e: AVERAGE

The next classifier is Average. This method is a simple stacking method that checks each model decision in Level 01 and averages its decision to decide its outcome.

f: WEIGHED AVERAGE

The next classifier is Weighed Average. This method is a simple stacking method that checks each model decision in Level 01 and averages its decision to decide its outcome, but each model has a different weight based on its importance.

g: BAGGING

The next classifier set is a class of ensemble methods called Bagging. This method divides the dataset into subsets with replacements, and these different subsets are the input data to the models used. Then the outcomes are combined to reach a combined decision. We applied Bagging for level 00 and level 01.

h: STACKING

The next classifier set is a class of ensemble methods called Stacking. This method stacks the models' decisions from Level 00 and uses their outcomes to create a new dataset. Then, models and ensemble methods are applied again in Level 01 to a decision.

B. HYPER-PARAMETER RATIONALE

For this work, we performed hyper-parameter tuning to maximize the performance of AI-based (or ML-based) classifiers. The rationale behind the hyper-parameter tuning in Appendix A is trial and error. Before applying the post-model XAI techniques in this work, we extensively trained and tested the same model with different parameters, checking its performance on a standard set of well-known metrics to evaluate each AI model for intrusion detection classification problems. These are accuracy (Acc), precision (Prec), recall (Rec), and F1-score (F1).

APPENDIX B
LIST OF TOP INTRUSION FEATURES

We show the list of top intrusion features for the three datasets in Table 22.

REFERENCES

[1] S. Northcutt and J. Novak, *Network Intrusion Detection*. Carmel, IN, USA: Sams, 2002.

[2] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Netw.*, vol. 8, no. 3, pp. 26–41, May 1994.

[3] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, "Modeling realistic adversarial attacks against network intrusion detection systems," *Digital Threats, Res. Practice (DTRAP)*, vol. 3, no. 3, pp. 1–19, 2022.

[4] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[5] A. S. Dina and D. Manivannan, "Intrusion detection based on machine learning techniques in computer networks," *Internet Things*, vol. 16, Dec. 2021, Art. no. 100462.

[6] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim, "Method of intrusion detection using deep neural network," in *Proc. IEEE Int. Conf. big data smart Comput. (BigComp)*, 2017, pp. 313–316.

[7] C. Tang, N. Luktarhan, and Y. Zhao, "SAAE-DNN: Deep learning method on intrusion detection," *Symmetry*, vol. 12, no. 10, p. 1695, 2020.

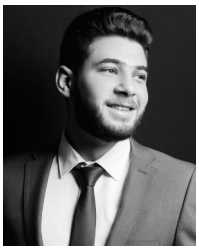
[8] P. Tao, Z. Sun, and Z. Sun, "An improved intrusion detection algorithm based on GA and SVM," *IEEE Access*, vol. 6, pp. 13624–13631, 2018.

[9] H. Deng, Q.-A. Zeng, and D. P. Agrawal, "SVM-based intrusion detection system for wireless ad hoc networks," in *Proc. IEEE 58th Veh. Technol. Conference. VTC Fall*, vol. 3, 2003, pp. 2147–2151.

[10] M. A. Ferrag, L. Maglaras, A. Ahmim, M. Derdour, and H. Janicke, "RDTIDS: Rules and decision tree-based intrusion detection system for Internet-of-Things networks," *Future Internet*, vol. 12, no. 3, p. 44, 2020.
[11] M. Al-Omari, M. Rawashdeh, F. Qutaishat, M. Alshira'H, and N. Ababneh, "An intelligent tree-based intrusion detection model for cyber security," *J. Netw. Syst. Manage.*, vol. 29, no. 2, pp. 1–18, 2021.
[12] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive Bayes vs decision trees in intrusion detection systems," in *Proc. ACM Symp. Appl. Comput.*, 2004, pp. 420–424.
[13] R. Panigrahi, S. Borah, M. Pramanik, A. K. Bhoi, P. Barsocchi, S. R. Nayak, and W. Alnumay, "Intrusion detection in cyber–physical environment using hybrid Naïve Bayes—Decision table and multi-objective evolutionary feature selection," *Comput. Commun.*, vol. 188, pp. 133–144, Jun. 2022.
[14] A. K. Balyan, S. Ahuja, U. K. Lilhore, S. K. Sharma, P. Manoharan, A. D. Algarni, H. Elmannai, and K. Raahemifar, "A hybrid intrusion detection model using EGA-PSO and improved random forest method," *Sensors*, vol. 22, no. 16, p. 5986, 2022.
[15] S. Waskle, L. Parashar, and U. Singh, "Intrusion detection system using PCA with random forest approach," in *Proc. Int. Conf. Electron. Sustain. Commun. Syst. (ICESC)*, Jul. 2020, pp. 803–808.
[16] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and M. Guizani, "A survey on IoT intrusion detection: Federated learning, game theory, social psychology, and explainable AI as future directions," *IEEE Internet Things J.*, vol. 10, no. 5, pp. 4059–4092, Mar. 2023.
[17] S. I. Sabev, "Integrated approach to cyber defence: Human in the loop. Technical evaluation report," *Inf. Security, An Int. J.*, vol. 44, pp. 76–92, Mar. 2020.
[18] S. D. DCunha. (2017). *Is AI Shifting The Human-In-The-Loop Model In Cybersecurity?*. Accessed: Oct. 21, 2021. [Online]. Available: https://datatechvibe.com/ai/is-ai-shifting-the-human-in-the-loop-model-in-cybersecurity/
[19] J. Mijalkovic and A. Spognardi, "Reducing the false negative rate in deep learning based network intrusion detection systems," *Algorithms*, vol. 15, no. 8, p. 258, 2022.
[20] N. H. Al-A'araji, S. O. Al-Mamory, and A. H. Al-Shakarchi, "Classification and clustering based ensemble techniques for intrusion detection systems: A survey," *J. Phys., Conf. Ser.*, vol. 1818, no. 1, 2021, Art. no. 012106.
[21] A. A. Aburomman and M. B. I. Reaz, "A survey of intrusion detection systems based on ensemble and hybrid classifiers," *Comput. Secur.*, vol. 65, pp. 135–152, Mar. 2017.
[22] B. A. Tama and S. Lim, "Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation," *Comput. Sci. Rev.*, vol. 39, Feb. 2021, Art. no. 100357.
[23] R. Lazzarini, H. Tianfield, and V. Charissis, "A stacking ensemble of deep learning models for IoT intrusion detection," *Knowl.-Based Syst.*, vol. 279, Nov. 2023, Art. no. 110941. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705123006913
[24] A. Mahfouz, A. Abuhussein, D. Venugopal, and S. Shiva, "Ensemble classifiers for network intrusion detection using a novel network attack dataset," *Future Internet*, vol. 12, no. 11, p. 180, 2020. [Online]. Available: https://www.mdpi.com/1999-5903/12/11/180
[25] N. Thockchom, M. Singh, and U. Nandi, "A novel ensemble learning-based model for network intrusion detection," *Complex Intell. Syst.*, vol. 9, pp. 1–21, Apr. 2023.
[26] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Netw. Distrib. Syst. Secur. (NDSS) Symp.*, 2018, pp. 1–15.
[27] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proc. ICML*, 2004, pp. 1–9.
[28] A. Zainal, M. Maarof, and S. M. Shamsuddin, "Ensemble classifiers for network intrusion detection system," *J. Inf. Assurance Secur.*, vol. 4, pp. 217–225, Jul. 2009.
[29] A. Z. Kiflay, A. Tsokanos, and R. Kirner, "A network intrusion detection system using ensemble machine learning," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2021, pp. 1–6.
[30] S. Das, S. Saha, A. T. Priyoti, E. K. Roy, F. T. Sheldon, A. Haque, and S. Shiva, "Network intrusion detection and comparative analysis using ensemble machine learning and feature selection," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 4821–4833, Dec. 2022.
[31] H. Zhang, J.-L. Li, X.-M. Liu, and C. Dong, "Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection," *Future Gener. Comput. Syst.*, vol. 122, pp. 130–143, Sep. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X2100114X
[32] Y.-F. Hsu, Z. He, Y. Tarutani, and M. Matsuoka, "Toward an online network intrusion detection system based on ensemble learning," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2019, pp. 174–178.
[33] Y. Alotaibi and M. Ilyas, "Ensemble-learning framework for intrusion detection to enhance Internet of Things' devices security," *Sensors*, vol. 23, no. 12, p. 5568, Jun. 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/12/5568
[34] R. Kumar Singh Gautam and Er. A. Doegar, "An ensemble approach for intrusion detection system using machine learning algorithms," in *Proc. 8th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence)*, Jan. 2018, pp. 14–15.
[35] T. Divyasree and K. Sherly, "A network intrusion detection system based on ensemble CVM using efficient feature selection approach," *Proc. Comput. Sci.*, vol. 143, pp. 442–449, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918321136
[36] O. Arreche, T. R. Guntur, J. W. Roberts, and M. Abdallah, "E-XAI: Evaluating black-box explainable AI frameworks for network intrusion detection," *IEEE Access*, vol. 12, pp. 23954–23988, 2024.
[37] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020.
[38] M.-E. Mihailescu, D. Mihai, M. Carabas, M. Komisarek, M. Pawlicki, W. Hołubowicz, and R. Kozik, "The proposition and evaluation of the RoEduNet-SIMARGL2021 network intrusion detection dataset," *Sensors*, vol. 21, no. 13, p. 4319, 2021.
[39] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 3, pp. 479–482, 2018.
[40] L. Dhanabal and S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446–452, 2015.
[41] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK: Design and philosophy," The MITRE Corp., McLean, VA, USA, Tech. Rep. 10AOH08A-JC, 2018.
[42] C. B. Lee, C. Roedel, and E. Silenok, "Detection and characterization of port scan attacks," Dept. Comput. Sci. Eng., Univeristy California, Los Angeles, CA, USA, Tech. Rep., 2003.
[43] *Drive-by Compromise*. Accessed: Oct. 21, 2023. [Online]. Available: https://attack.mitre.org/techniques/T1189/
[44] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: Analysis of a botnet takeover," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 635–647.
[45] Y. Chen, Q. Lin, W. Wei, J. Ji, K.-C. Wong, and C. A. Coello Coello, "Intrusion detection using multi-objective evolutionary convolutional neural network for Internet of Things in fog computing," *Knowl.-Based Syst.*, vol. 244, May 2022, Art. no. 108505. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705122002179
[46] V. Gorodetski and I. Kotenko, "Attacks against computer network: Formal grammar-based framework and simulation tool," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses (RAID)*. Heraklion, Greece: Springer, 2002, pp. 219–238.
[47] M. Skwarek, M. Korczynski, W. Mazurczyk, and A. Duda, "Characterizing vulnerability of DNS AXFR transfers with global-scale scanning," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2019, pp. 193–198.
[48] A. Khan, H. Kim, and B. Lee, "M2MON: Building an MMIO-based security reference monitor for unmanned vehicles," Usenix, Adv. Comput. Syst. Assoc., Vancouver, BC, Canada, Tech. Rep. 978-1-939133-24-3, 2021.
[49] S. R. Hussain, I. Karim, A. A. Ishtiaq, O. Chowdhury, and E. Bertino, "Noncompliance as deviant behavior: An automated black-box noncompliance checker for 4G LTE cellular devices," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 1082–1099.
[50] O. Mirzaei, R. Vasilenko, E. Kirda, L. Lu, and A. Kharraz, "Scrutinizer: Detecting code reuse in malware via decompilation and machine learning," in *Proc. 18th Int. Conf. (DIMVA)*. Cham, Switzerland: Springer, Jul. 2021, pp. 130–150.
[51] S. Lukacs, D. H. Lutas, A. V. COLESA, "Strongly isolated malware scanning using secure virtual containers," U.S. Patent 9 117 081, Aug. 25, 2015.

[52] A. Kim, M. Park, and D. H. Lee, "AI-IDS: Application of deep learning to real-time Web intrusion detection," *IEEE Access*, vol. 8, pp. 70245–70261, 2020.
[53] *Flow Information Elements—Nprobe 10.1 Documentation*. Accessed: Apr. 9, 2024. [Online]. Available: https://www.ntop.org/guides/nprobe/flow_information_elements.html
[54] Ahlashkari. (Jun. 2021). *Cicflowmeter/Readme. Txt At Master · Ahlashkari/Cicflowmeter*. [Online]. Available: https://github.com/ahlashkari/cicflowmeter/blob/master/readme.txt
[55] B. Claise, "Cisco systems Netflow services export version 9," Cisco Syst., San Jose, CA, USA, Tech. Rep. RFC 3954 , 2004.
[56] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Softw. Netw.*, vol. 2018, no. 1, pp. 177–200, 2018.
[57] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.
[58] R. Zhao. (2022). *NSL-KDD*. [Online]. Available: https://dx.doi.org/10.21227/8rpg-qt98
[59] C. A. Stewart, V. Welch, B. Plale, G. C. Fox, M. Pierce, and T. Sterling, "Indiana university pervasive technology institute," Indiana Univ., Bloomington, IN, USA, Tech. Rep. K8G44NGB, 2017.
[60] W. Li, P. Yi, Y. Wu, L. Pan, and J. Li, "A new intrusion detection system based on KNN classification algorithm in wireless sensor network," *J. Electr. Comput. Eng.*, vol. 2014, Jun. 2014, Art. no. 240217.
[61] J. O. Mebawondu, O. D. Alowolodu, J. O. Mebawondu, and A. O. Adetunmbi, "Network intrusion detection system using supervised learning paradigm," *Sci. African*, vol. 9, Sep. 2020, Art. no. e00497.
[62] Y.-Y. Song and L. Ying, "Decision tree methods: Applications for classification and prediction," *Shanghai Arch. Psychiatry*, vol. 27, no. 2, p. 130, 2015.
[63] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: A methodology review," *J. Biomed. Informat.*, vol. 35, nos. 5–6, pp. 352–359, 2002.
[64] A. Veronika Dorogush, V. Ershov, and A. Gulin, "CatBoost: Gradient boosting with categorical features support," 2018, *arXiv:1810.11363*.
[65] D. Jin, Y. Lu, J. Qin, Z. Cheng, and Z. Mao, "SwiftIDS: Real-time intrusion detection system based on LightGBM and parallel intrusion detection mechanism," *Comput. Secur.*, vol. 97, Oct. 2020, Art. no. 101984.
[66] A. Yulianto, P. Sukarno, and N. A. Suwastika, "Improving adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset," *J. Phys., Conf. Ser.*, vol. 1192, no. 1, 2017, Art. no. 012018.
[67] S. S. Dhaliwal, A.-A. Nahid, and R. Abbas, "Effective intrusion detection system using XGBoost," *Information*, vol. 9, no. 7, p. 149, 2018.
[68] P. Bühlmann, "Bagging, boosting and ensemble methods," in *Handbook of Computational Statistics: Concepts and Methods*, 2012, pp. 985–1022.
[69] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. Int. Workshop Multiple Classifier Syst.* Günzburg, Germany: Springer, 2000, pp. 1–15.
[70] M. Zounemat-Kermani, O. Batelaan, M. Fadaee, and R. Hinkelmann, "Ensemble machine learning paradigms in hydrology: A review," *J. Hydrol.*, vol. 598, Jun. 2021, Art. no. 126266.
[71] I Insights. (Feb. 2020). *42 Cyber Attack Statistics by Year: A Look At the Last Decade*. Accessed: Mar. 10, 2023. [Online]. Available: https://sectigostore.com/blog/42-cyber-attack-statistics-by-year-a-look-at-the-last-decade/
[72] V. G. Costa and C. E. Pedreira, "Recent advances in decision trees: An updated survey," *Artif. Intell. Rev.*, vol. 56, no. 5, pp. 4765–4800, 2023.

OSVALDO ARRECHE received the bachelor's degree in instrumentation, automation, and robotics from the Federal University of ABC (UFABC), Santo André, Brazil. He is currently pursuing the master's degree in electrical and computer engineering with the Purdue School of Engineering and Technology, Indiana University–Purdue University Indianapolis (IUPUI). He is a Research Assistant focusing on cybersecurity and explainable AI. He has several industry and manufacturing experience, having spent three years as an Automation Engineer in food and pharma manufacturing, he has hands-on experience with technology's practical applications. He aspires to leverage his expertise in creating transparent and ethically-driven technological advancements.



ISMAIL BIBERS received the bachelor's degree in computer science. He is currently pursuing the master's degree in cybersecurity and trusted systems with the Purdue School of Engineering and Technology, Indiana University–Purdue University Indianapolis (IUPUI). He has worked as a Research Support, concentrating on cybersecurity and machine learning projects. Additionally, he has a background in education, having served as a coding teacher for a year. Furthermore, he has spent time as a Backend Developer, contributing to various industry projects. His academic research interests include cybersecurity, network security, and machine learning. During his academic journey, he has gained practical experience in both academia and industry.

MUSTAFA ABDALLAH (Member, IEEE) received the M.Sc. degree in engineering mathematics from the Faculty of Engineering, Cairo University, and the Ph.D. degree from the Elmore Family School of Electrical and Computer Engineering, Purdue University. He is currently an Assistant Professor with the Purdue School of Engineering and Technology, Indiana University–Purdue University Indianapolis (IUPUI). He has also several industrial research experiences, including internships with the Adobe Research, and a Principal, and a five-year machine learning research experience with RDI (a leading machine learning company in Egypt). His research interests include game-theory, human decision-making, explainable AI, and machine-learning with applications, including network security and autonomous driving systems. His research contribution is recognized by receiving the Purdue Bilsland Dissertation Fellowship and having many publications in top IEEE/ACM journals and conferences. He was a recipient of the M.Sc. Fellowship from the Faculty of Engineering, Cairo University, in 2013.