scientific reports

OPEN

Leveraging stacking machine learning models and optimization for improved cyberattack detection

Neha Pramanick[1], Jimson Mathew[1], Shitharth Selvarajan[2,3,4] & Mayank Agarwal[1]

The ever-growing number of complex cyber attacks requires the need for high-level intrusion detection systems (IDS). While the available research deals with traditional, hybrid, and ensemble methods for network data analysis, serious challenges are still being met in terms of producing robust and highly accurate detection systems. There are high hurdles in managing high-dimensional network traffic since current methodologies are limited in dealing with imbalanced data issues of minority classes versus the majority and high false positive rate in classification accuracy. This study introduces an innovative framework that directly addresses these persistent challenges through a novel approach to intrusion detection. The proposed method integrates two ML models: J48 and ExtraTreeClassifier for classification. Besides, we propose an improved equilibrium optimizer (EO) approach whereby the previous EO is modified. In this enhanced equilibrium optimizer (EEO), the Fisher score and accuracy score of the K-Nearest Neighbors (KNN) algorithm select the attributes optimally, whereas the synthetic minority oversampling technique combined with iterative partitioning filters (SMOTE-IPF) used to provide class balancing. The KNN technique is also used for data imputation to improve the overall system accuracy. The superior performance of the framework has been validated experimentally on several benchmark datasets, i.e., NSL-KDD, and UNSW-NB15, achieving 99.7% and 98.1% accuracy and F1 score 99.6 and 98.0 respectively. By subjecting the system to a comparative analysis with recent state-of-the-art works, this paper has shown that the proposed methodology yields better improvement in feature selection precision classification accuracy, handling of minority class instance, less demanding storage and computational efficiency.

Keywords IDS, Stacking ML, UNSW-NB15, NSL-KDD, Optimization, SMOTE

Today, cyberattacks target individuals, businesses, and governments. These attacks include data breaches, ransomware, malware, phishing, and DDoS attacks that exploit system, network, or human behavior vulnerabilities. Cybercriminals use artificial intelligence and machine learning to automate attacks and avoid detection as technology advances. Deloitte's Cybersecurity Threat Trends Report indicates that ransomware affected 66% of firms in 2023, and IoT malware attacks, particularly in manufacturing, surged 400%. The abuse of authorized credentials caused nearly half of data breaches, making it a major concern for cybersecurity experts in the coming year[1]. Intrusion Detection Systems (IDS) play a pivotal role in enhancing cybersecurity by identifying and mitigating unauthorized or malicious activities within networks and systems[2]. These systems monitor network traffic and system behavior to detect potential threats through indicators such as unusual patterns, known attack signatures, or unauthorized access attempts. By providing real-time alerts and detailed logs, IDS enable security teams to respond promptly, thus minimizing the impact of cyberattacks. As cyber threats continue to evolve, modern IDS leverage advanced methods, including machine learning (ML) and artificial intelligence (AI), to enhance detection capabilities and reduce false positives[3].

IDS can be classified based on the detection methods and the scope of monitoring. Signature-based IDS detect threats by comparing network traffic against known attack patterns, identifying threats through predefined signatures. In contrast, anomaly-based IDS flag deviations from normal behavior, offering a more dynamic approach to uncover unknown threats[4]. In terms of scope, network-based IDS monitor and analyze traffic across an entire network, providing broad coverage against external threats, while host-based IDS focus on local system

[1]Computer Science and Engineering, IIT Patna, Patna, Bihar 801103, India. [2]Department of Computer Science, Kebri Dehar University, 250 Kebri Dehar, Ethiopia. [3]Department of Computer Science and Engineering, Chennai Institute of Technology, Chennai, India. [4]Centre for Research Impact & Outcome, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab 140401, India. email: shitharthS@kdu.edu.et

activities and logs on individual hosts to detect suspicious behavior within a specific system. Together, these types of IDS provide comprehensive security solutions essential for modern networks and systems.

The integration of ML into IDS is increasingly popular among researchers aiming to boost detection accuracy and adaptability against emerging cyber threats[5–8]. ML models excel at identifying complex patterns within data, enabling IDS to effectively detect both known and unknown attacks[9]. However, the incorporation of ML highlights the significance of feature selection. Feature selection plays a critical role by pinpointing the most relevant attributes from large datasets, minimizing noise and irrelevant data that could negatively impact model performance[10]. By focusing on the key features, researchers can significantly enhance the efficiency, accuracy, and responsiveness of ML models.

IDS face various key challenges, including big data and high-dimensionality of features in network traffic data, thereby hampering successful identification of the key features. Imbalanced target classes often make it difficult for the models to accurately eliminate attacks from normal traffic. One of the negative consequences of the redundant or irrelevant variables is the decrease in the effectiveness of the model to separate normal and attack traffic and take more processing time. Furthermore, attacks are ever-changing in nature, thus requiring dynamic and resilient IDS models to work effectively in the long term. To mitigate these challenges, this research incorporates the enhanced equilibrium optimizer (EEO), modified algorithm of traditional equilibrium optimizer (EO) algorithm for feature selection to determine the most optimal features to retain, based on a combination of fisher scores and KNN-based accuracy assessment. For the class imbalanced, we have introduced synthetic minority over-sampling technique- iterative partitioning filter (SMOTE-IPF) to refine the representation of the minority attack classes and several noisy instances are thereby removed. Subsequently, yet another ML model of type stacking is included, using a base classifier model, J48, and a meta classifier, ExtraTreeClassifier. This stacking method significantly consolidated the predictive powers of models improving the classification accuracy and also broadening the range of attacks identified by the system. Taken together, the several aspects work to build the IDS with better detection performance and resilience against the newer information security threats in them.

Research gap: The research presented in this paper highlights a significant gap in the domain of IDS, emphasizing the need for a more functional approach to address persistent challenges such as imbalanced datasets, high false positive rates, and feature selection. Existing IDS methodologies have consistently struggled to adapt to the evolving nature of cyber threats, limiting their effectiveness in maximizing system performance. Given the rapid advancements in technology, there is a pressing need for a robust and well-designed IDS framework that leverages modern ML and optimization techniques to enhance security and threat detection.

Novelty: This study introduces a unique IDS framework that integrates feature selection, data balancing, and advanced classification techniques to improve detection accuracy. A key innovation is the development of the EEO, designed for faster convergence and efficient feature selection. Additionally, this research employs SMOTE-IPF to address class imbalance and mitigate noisy data, which is a critical challenge in cybersecurity datasets. To further enhance classification robustness, a stacking model combining J48 and ExtraTreeClassifier is implemented, ensuring improved detection accuracy and system reliability.

Research questions: This study aims to answer several key research questions. It investigates the pre-processing techniques used to eliminate irrelevant parameters and handle missing values. Furthermore, it explores the meta-heuristic approach employed for selecting optimal features from the extracted dataset. The study also examines how the class imbalance issue is effectively addressed in the pre-processed data and evaluates the role of a stacking model in enhancing classification accuracy for intrusion detection.

Hypothesis: The central hypothesis of this research is that integrating EEO for feature selection, SMOTE-IPF for class balancing, and a stacking-based ML approach will significantly enhance the accuracy and efficiency of IDS. This combination is expected to result in lower false positive rates and improved detection of both known and unknown cyber threats, thereby strengthening the overall cybersecurity framework.

Contribution: This paper makes several significant contributions to the field of IDS. It proposes a scalable and computationally efficient IDS framework that is adaptable to modern cyber threats. The study enhances detection precision through optimized feature selection using EEO, ensuring that only the most relevant features contribute to classification. The application of SMOTE-IPF improves detection rates for rare attacks by effectively balancing class distributions. Additionally, the hybrid stacking model employed in this research leads to higher classification accuracy and reduced false positive rates. Ultimately, this study presents a comprehensive solution for robust defense mechanisms against both known and emerging cyber threats, making a substantial impact on the field of intrusion detection with less processing time.

In the rest of the paper, the abbreviations and terms presented in Table 1 will be used to ensure consistency and readability.

Related work

Turk et al.[11], evaluated some ML algorithms over the UNSW-NB15 and NSL-KDD datasets, which included DT, SVM, and KNN. In this work, both challenges to these algorithms were noticed in relation to feature selection and class imbalance. Even though the algorithms showed reasonably good performance, the necessity of a better feature extraction method and balancing technique has been concluded in the said study, to enhance their effectiveness in any IDS. The comparative analysis showed the difference in the accuracy with false positive rates, thereby proving the need for more robust solutions within the framework of IDS. Wang et al.[12] developed a hybrid IDS by integrating RF with Autoencoder, aiming to leverage the strengths of both supervised and unsupervised learning approaches. The hybrid model achieved a higher detection rate than using either algorithm alone, particularly excelling in identifying known attack patterns. However, due to the imbalanced nature of the datasets, the model experienced a significant drop in precision and recall, particularly in detecting minority class attacks. The authors concluded that the implementation of robust data balancing techniques could

Term	Meaning
AI	Artificial intelligence
CNN	Convolutional neural network
DL	Deep learning
DT	Decision tree
EEO	Enhanced equilibrium optimizer
EO	Equilibrium optimizer
IDS	Intrusion detection system
KNN	k-Nearest neighbors
ML	Machine learning
SMOTE-IPF	Synthetic minority over-sampling technique with iterative-partitioning filter
SVM	Support vector machine

Table 1. A list of abbreviations and terms used in the paper.

substantially improve the model's performance, particularly in accurately identifying less frequent attack types. Zhour et al.[13], proposed a hybrid IDS combining RF, DT and Multilayer Perceptron algorithms (MLP). The idea behind the approach was that the strengths of each algorithm would complement each other in improving overall detection rates. However, the authors of the study pointed out that the process of managing feature selection and class imbalance across different algorithms was complex. The hybrid system still performed better compared to individual algorithms.

Kasong et al.[14] employed the UNSW-NB15 dataset within an IDS framework utilizing XGBoost. The study selected 17 features with XGBoost and achieved 87.07% accuracy for binary classification using RNN. For multi-class classification, the highest accuracy obtained was 78.4% using GRU. However, the study noted limitations in its approach, including challenges with feature selection and the generalizability of the models to other datasets. Khan et al.[15] built an IDS with RF combined with multiple ML models. In the RF algorithm, there are scores given to features to help in determining feature importance; the greater the score, the greater its significance. They selected 11 features out of 49 in the UNSW-NB15 dataset by running several iterations. This reduced feature set made the classification process more efficient because there existed a strong relation between the feature scores and classificatory outputs. Individual classification models used in this study included DT, KNN, XGBoost, and bagging meta estimators, such as RF. Almomani et al.[16] utilized particle swarm optimization (PSO) for feature reduction in their IDS implementation on the UNSW-NB15 dataset. They also applied genetic algorithms, firefly optimization, and grey wolf optimization for feature selection. To achieve optimal attack detection accuracy, they ultimately chose 30 attributes. For the classification process, they employed SVM and the J48 tree-based method.

Yin et al.[17] proposed a hybrid feature selection method with a combination of information gain and RF, while using MLP for classification, in order to better the result. Generalizing from this approach, IGRF-RFE was a combination of the merits of information gain and RF with the accuracy of recursive feature elimination. It hence greatly enhances feature selection, which provides a major accuracy boost through feature-space dimensionality reduction from 42 to 23 for the UNSW-NB15 dataset. In a similar context, researchers achieved notable results with advanced techniques on the UNSW-NB15 dataset. Kasongo et al.[18] utilized the XGBoost algorithm for feature selection and applied ML models such as SVM and ANN, attaining an accuracy of 77%. Ethala et al.[19] proposed a hybrid optimization strategy that combines Spider Monkey Optimization (SMO) with Hierarchical Particle Swarm Optimization (HPSO) to handle the large volumes of IDS data.

Abdulganiyu et al.[20] introduced the XIDINTFL-VAE method, comparing its performance against various oversampling techniques such as SMOTE, Borderline-SMOTE, and Adaptive Synthetic Sampling (ADASYN), as well as traditional classifiers like Logistic Regression, KNN, SVM, and DT. While conventional methods primarily focus on dataset balancing or generating realistic data, the CWFL-VAE approach takes a more strategic direction by synthesizing data specifically to enhance the classifier's ability to handle challenging minority class instances. Their experiments, conducted on the NSL-KDD and CSE-CIC-IDS2018 datasets, showed that the XIDINTFL-VAE model outperformed traditional techniques, achieving a precision of 99.67% and an F1 score of 94.74%, with a minor trade-off in recall at 89.41%. Saheed et al.[21] proposed a hybrid feature selection method that integrates the Bat metaheuristic algorithm with the Residue Number System (RNS). The Bat algorithm is first employed to partition the training data and remove irrelevant features. However, due to the Bat algorithm's slower training and testing times, RNS is introduced to improve computational efficiency.

Saheed et al.[22] introduced a feature selection approach that combines the Information Gain and Ranker (IG+R) method with Naïve Bayes (NB), SVM, and KNN classifiers. They evaluated the performance of IG+R-NB, IG+R-SVM, and IG+R-KNN on the NSL-KDD dataset. Experimental results demonstrated that their proposed method achieved high accuracy while maintaining a low false alarm rate. Saheed et al.[23] implemented the Modified Genetic Algorithm (MGA) for feature selection and Genetic Algorithm (GA) for optimizing the Long Short-Term Memory (LSTM) model parameters within an Evolutionary Computing (EC) framework. The GA fitness function was used to fine-tune the LSTM parameters by adjusting the number of hidden layers. The MGA was specifically customized to improve feature selection efficiency, optimizing resource utilization on IoT devices and edge nodes. YK Saheed et al.[24] integrated deep Shapley Additive Explanations (SHAP) to enhance model transparency, making its decisions more interpretable for cybersecurity professionals. Furthermore, they employed a hybrid bidirectional long short-term memory with autoencoders (BiLAE) to reduce the dimensionality of Internet of Vehicles (IoV) network traffic, thereby improving computational efficiency.

Saheed et al.[25] implemented an IDS utilizing an autoencoder for feature dimensionality reduction, trained on network flow data using a Deep Convolutional Neural Network (DCNN) and Long Short-Term Memory (LSTM). Their approach did not require prior knowledge of the network's architecture or topology. The experimental analysis was conducted on the ICS dataset and gas pipeline data provided by Mississippi State University (MSU). Recent advancements in adjacent technologies have significantly influenced modern intrusion detection approaches. In the IoT service composition domain, Vakili et al.[26] proposed a novel method combining Grey Wolf Optimization with MapReduce framework that offers optimization techniques applicable to distributed IDS architectures. Similarly, deepfake detection research by Heidari et al.[27] comprehensively reviewed DL that share fundamental feature extraction approaches with modern network traffic analysis. Heidari et al.[28] extended this work by introducing blockchain-based deepfake detection using federated learning, demonstrating security mechanisms transferable to distributed IDS environments. DL applications in IoT-based medical informatics, analyzed by Amiri et al.[29], revealed pattern recognition techniques that can be adapted for identifying anomalous network behavior in healthcare systems. For industrial environments, Heidari et al.[30] developed a reliable data aggregation method using hybrid optimization algorithms that can enhance IDS accuracy while minimizing false positives in OT networks. Directly addressing IDS applications, Heidari et al.[31] introduced a secure intrusion detection platform for IoT drone networks using blockchain and neural networks, achieving superior detection rates compared to traditional approaches. The reliability assessment methodology for wireless sensor networks in industrial settings by A Heidari et al.[32] provided fault-tolerance mechanisms relevant to maintaining IDS availability in distributed environments. Finally, the systematic review by Amiri et al.[33] on nature-inspired algorithms for healthcare IoT services offers optimization approaches that could enhance the computational efficiency of our proposed IDS framework. In the blockchain-based industrial IoT domain, Zanbouri et al.[34] proposed a GSO-based multi-objective technique for performance optimization that offers security enhancement methods relevant to distributed IDS frameworks. The comprehensive analysis of ChatGPT capabilities and applications by Heidari et al.[35] revealed potential AI based approaches for automating threat intelligence and attack pattern recognition in modern IDS systems.

Amiri et al.[36] provided a comprehensive survey of AI techniques for climate change mitigation, showcasing adaptive learning methods transferable to evolving threat detection in network environments. Directly addressing cybersecurity concerns, Asadi et al.[37] unveiled botnets through a comprehensive survey of evolving threats and defense strategies, establishing detection patterns applicable to our IDS implementation. For vehicular networks, Heidari et al.[38] developed fuzzy logic multicriteria decision-making approaches for broadcast storm resolution that can be adapted to prioritize alerts in high-traffic IDS deployments. The computational optimization techniques by A Heidari et al.[39] for enhancing solar convection analysis with multi-core processors and GPUs demonstrated parallel processing methods that could significantly improve IDS performance for real-time threat detection. The systematic review of DL applications in Alzheimer's disease by Toumaj et al.[40] revealed anomaly detection methodologies adaptable to identifying subtle deviations in network behavior patterns. While seemingly distant from cybersecurity, the historical computing research in Iran by Heidari et al.[41] provides valuable context for understanding regional cybersecurity approaches and threat landscapes. Finally, Jabraeil Jamali et al.[42] outlined the IoT landscape, establishing fundamental architectural considerations that directly impact security monitoring and intrusion detection in connected device environments.

IDS have greatly developed through the intervention of various ML techniques, including classical, hybrid, and ensemble techniques. Classical ML, like DT, SVM, and KNN, are most used in IDS because of their interpretability and efficiency. However, due to high dimensional data, class imbalance, and overfitting, these models often become less useful when attempting to detect complex attack patterns. Hybrid methods mix the learning practices in the form of the integration of deep DL models with conventional ML models to enhance performance. For instance, various studies have tried CNNs together with feature-based ML classifiers. Although hybrid methodologies improve detection accuracy, they usually carry a high computational load. In the ensemble learning methods, the classification accuracy tends to be increased because of the combination of different models, leading to generalizations. These methods would indeed prove as more reliable than separate classifiers for guiding toward over-learning, and the way they employ only homogeneous learners may lead to redundancy and excessive computational burden. Table 2 is presented to give an overview of existing models in IDS.

Large research works reveal that considerable work has been done by the researchers themselves in IDS, but they have found out that modern approaches are constrained by some challenges. Most of these problems are faced by existing solutions in that the feature selection and class imbalance go hand in hand while proving class discrimination. Our design attempts to incorporate the best of both hybrid and ensemble approaches and introduces a stacking-based IDS model that integrates a feature selection, data balancing, and a diversified ensemble approach to gain an improvement in efficiency. Feature selection, which was done with EEO, resulted in a reduced multi-dimensionality without losing the necessary attack features. By reducing the number of irrelevant and redundant features, it helps to extend beyond classic ML models due to different accurate selections. For the problem of data imbalance in cybersecurity datasets we use SMOTE-IPF, which balances the classes better and tends to eliminate the noise that is usually produced due to synthetic samples. They design a method that is essentially most cost-efficient than other hybrid methodologies where DL is called to rescue and yet having the same or improved effects on balancing. This study also upgrades the traditional ensemble methods with an abstract form by focusing on installing a stacking model of J48, a classical ML model as base classifier and ExtraTreeClassifier, an ensemble ML method as meta classifier. A stacking approach allows us to model relatively non-linear decision patterns and increased generalization from the data through randomness, thereby producing no smaller classification accuracy performance.

Author	Technique	Limitation
Turk et al.[11]	Decision trees, SVM, KNN	The study faced challenges in feature selection, leading to redundant or irrelevant features affecting classification performance. Additionally, class imbalance in the dataset resulted in biased predictions favoring majority classes
Wang et al.[12]	Random forest and autoencoder	Despite feature extraction using autoencoders, the model exhibited reduced precision and recall due to the severe class imbalance, making detection of minority-class attacks less reliable
Zhour et al.[13]	RF, decision tree, MLP	The study implemented multiple models for classification, but the feature selection process was complex and computationally expensive. Moreover, handling class imbalance required additional techniques, increasing system overhead
Kasong et al.[14]	XGBoost, RNN, GRU	The model suffered from feature selection issues, leading to suboptimal performance. Furthermore, DL-based models like RNN and GRU posed challenges in generalizability across different datasets due to overfitting risks
Khan et al.[15]	Random forest with various ML models	The approach focused primarily on feature importance rankings without effectively addressing feature redundancy. This could lead to misinterpretation of feature contributions and potential overfitting
Almomani et al.[16]	PSO, Genetic algorithms, etc	The use of multiple evolutionary algorithms for feature selection introduced complexity in optimization. Balancing multiple objectives, such as feature reduction and accuracy improvement, proved to be computationally expensive and sensitive to parameter tuning
Yin et al.[17]	IG, RF with MLP	The hybrid approach showed strong performance, but the integration of Information Gain (IG) and Random Forest (RF) with MLP led to a risk of overfitting. Additionally, generalization across unseen datasets remained a challenge
Kasongo et al.[18]	XGBoost, SVM, ANN	The proposed model achieved only 77% accuracy, indicating potential issues with feature selection, hyperparameter tuning, or class imbalance handling, which limited its effectiveness
Ethala et al.[19]	SMO and HPSO	The hybrid optimization approach aimed at improving feature selection, but it did not fully resolve data imbalance issues, potentially leading to misclassification of minority-class attacks
Yao et al.[43]	Ensemble on NSL-KDD	The study employed an ensemble approach; however, it did not provide details on how class imbalance was handled, raising concerns about the effectiveness of attack detection in underrepresented categories
Das et al.[44]	Decision tree on NSL-KDD	The use of a single decision tree model made the system prone to overfitting, reducing its adaptability to new attack patterns and limiting its generalizability across datasets

Table 2. Comparison of IDS techniques with limitations.

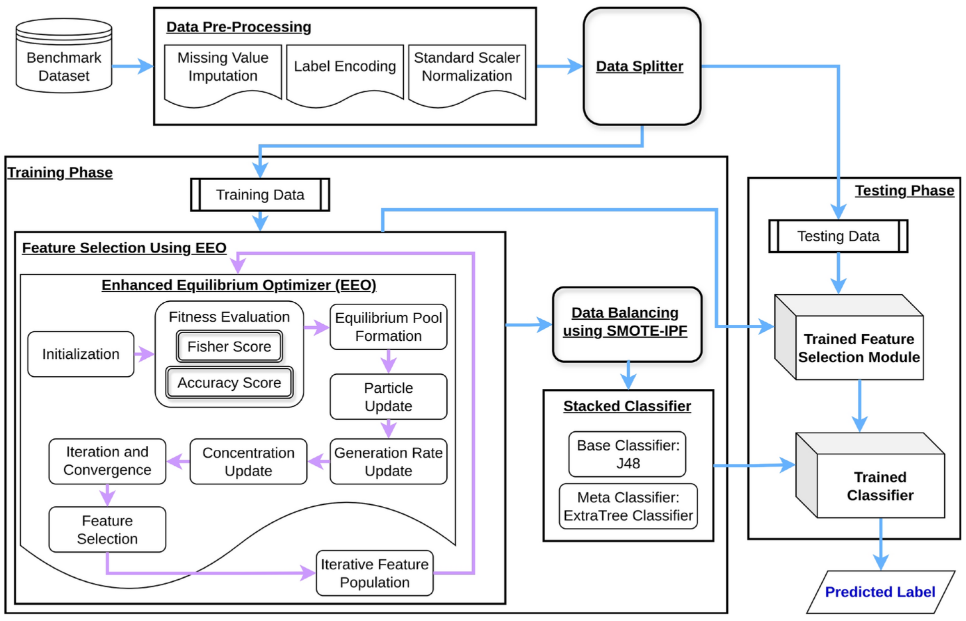


Fig. 1. Proposed model: the model uses EEO for feature selection, SMOTE-IPF for data balancing and stacking model (base classifier:J48, meta classifier: ExtraTreeClassifier) for classification of attack classes.

Proposed method

The main challenges for IDS are high dimensionality and class imbalance, which lower its detection performance. Most of the feature selection methods and data balancing techniques applied separately to combat this suffer from deficiencies in providing better accuracy and generalization. To this end, more effective approaches should integrate robust feature selection, class balancing, and advanced ML techniques for boosting IDS efficiency. In this paper, we present a complete IDS framework using cybersecurity datasets that could handle such challenges in a multi-step manner. First, after preprocessing the dataset, we selected features using the metaheuristic technique EEO, which reduces the dataset's dimensionality by focusing on only relevant features. To address class imbalance, we apply SMOTE-IPF to ensure balanced training data. Finally, proposed stacking ML classifier uses the strengths of several models to improve detection accuracy. The proposed method is designed to improve accuracy, precision, recall, and F1-score while also reducing training and prediction times. This method will contribute to a new hybrid approach in IDS. The architecture of proposed method is shown in Fig. 1.

Data pre-processing

Processing raw data for ML models begins with data pre-processing, which includes handling missing values, scaling features, encoding categorical variables, reducing outliers, and feature engineering. This step is crucial as it guarantees the completeness, cleanliness, and suitability of the dataset for model training. Managing missing values and feature scales during pre-processing enhances model correctness, speed, and generalizability. The model requires the numerical expression of categorical variables. Data pre-processing is critical for machine learning model performance and accurate, meaningful results.

Missing value imputation

KNN imputes missing values by finding the ‘k’ nearest neighbors for each instance with missing data using a distance metric, typically Euclidean distance[45]. For the ‘Service’ feature in the UNSW-NB15 dataset, KNN calculates the distance between the incomplete instance and all other complete instances. Once we identify the ‘k’ nearest neighbors, we impute the missing value by either averaging the neighbor values for numerical features or using a majority vote for categorical features. We chose KNN because it effectively captures local patterns in the data and does not require assumptions about the underlying data distribution, making it more flexible and accurate than many other imputation methods. Table 3 shows the results of imputation in ‘Service’ feature in UNSW-NB15 dataset.

Label encoding

Three categorical features—“service”, “proto”, and “state”—with nominal values are included in the UNSW-NB15 dataset. NSL-KDD has “Protocol_Type”, “Service”, and “Flag” in a similar manner. Label encoding was used to process these features, turning each nominal value into a binary feature.

Value standardization using standard scaler normalization

Normalization using a standard scaler transforms the feature values so that they have a mean of zero and a standard deviation of one. Given a feature X with n observations, the standard scaler normalization process involves two steps:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{1}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2} \tag{2}$$

Normalize the Values: For each observation X_i, normalize its value X_i' using the formula:

$$X_i' = \frac{X_i - \mu}{\sigma} \tag{3}$$

This formula subtracts the mean (μ) from each observation and divides by the standard deviation (σ). As a result, the normalized values will have a mean of zero and a standard deviation of one. We used Standard Scaler normalization in our research to make sure that all features had the same mean and variance. This helped J48,

	Prior to the utilization of KNN		**After implementing KNN**	
Sl No.	**Service**	**Count**	**Service**	**Count**
1	Others	141321	dns	104559
2	dns	68661	http	56229
3	http	27011	smtp	13408
4	smtp	6909	ftp-data	13306
5	ftp-data	5391	ftp	9315
6	ftp	4980	pop3	2812
7	pop3	1528	ssh	5473
8	ssh	1506	dhcp	3761
9	dhcp	120	snmp	4550
10	snmp	109	ssl	168
11	ssl	86	irc	50
12	irc	30	Radius	6944
13	Radius	21		

Table 3. Comparison of counts of each feature elements of “service” of UNSW-NB15 dataset before and after applying KNN, where ‘other’ represents the missing values of feature “service”.

ExtraTreeClassifier, and metaheuristic methods like EEO while keeping the integrity of the feature distribution. Outliers can cause other methods to distort data.

Enhanced equilibrium optimizer for feature selection

The EO algorithm[46] drawn from control volume mass balance models that are used to estimate dynamic and equilibrium states. It can be effectively used for feature selection. Algorithm 1 explains about EO algorithm while the Algorithm 2 explains the steps of feature selection using EEO. In the proposed methodology we enhanced the performance of EO by changing the fitness function of the optimization technique. We incorporated the fitness function by combining Fisher score and the accuracy score generated by KNN in EEO.

```
1: Input: Dataset $D$ with features $X = \{x_1, x_2, \ldots, x_m\}$, population size $n$, max iterations $Max\_Iter$, constants $a_1$, $a_2$
2: Output: Selected subset of features $X_{\text{best}}$
3: Initialize the population of particles (solutions), each representing a subset of features
4: for each particle $i$ in the population do
5:     Initialize concentration $C_i^{\text{initial}} = C_{\min} + \text{rand}_i \times (C_{\max} - C_{\min})$
6: end for
7: Evaluate the fitness of each particle based on a selected fitness function (e.g., classification accuracy)
8: Sort particles based on their fitness and identify the best-so-far particles
9: for iteration Iter = 1 to Max_Iter do
10:     Form the equilibrium pool $\vec{C}_{\text{eq,pool}} = \{\vec{C}_{\text{eq}(1)}, \vec{C}_{\text{eq}(2)}, \vec{C}_{\text{eq}(3)}, \vec{C}_{\text{eq}(4)}, \vec{C}_{\text{eq,ave}}\}$
11:     Update time $t = \left(1 - \frac{\text{Iter}}{\text{Max\_Iter}}\right) \times \left(\frac{a_2 \times \text{Iter}}{\text{Max\_Iter}}\right)$
12:     Calculate the exponential term $\vec{F} = a_1 \times \text{sign}(\vec{r} - 0.5) \times \left[e^{-\lambda t} - 1\right]$
13:     Calculate the generation rate $\vec{G} = \vec{G}_0 \times e^{-\lambda(t - t_0)}$
14:     for each particle $i$ in the population do
15:         Update concentration $\vec{C}_i = \vec{C}_{\text{eq}} + \left(\vec{C}_i - \vec{C}_{\text{eq}}\right) \times \vec{F} + \frac{\vec{G}}{\lambda V} \times (1 - \vec{F})$
16:         Evaluate the updated particle's fitness
17:     end for
18:     Sort particles based on their updated fitness and update the equilibrium pool
19: end for
20: return the feature subset corresponding to the best-performing particle $X_{\text{best}}$
```

Algorithm 1. Feature selection using EO

```
1: Input: Dataset $D$ with features $X = \{x_1, x_2, \ldots, x_m\}$, population size $n$, max iterations Max_Iter, constants $a_1, a_2, \lambda$, and hyperparameters: Adaptive_Factor, Progress_Factor, Update_Strategy
2: Output: Selected subset of features $X_{\text{best}}$
3: Step 1: Initialization
4: Initialize a population of particles with randomly assigned concentration vectors
5: Ensure a diverse starting population
6: Step 2: Fitness Evaluation
7: Evaluate each particle's fitness using a combined measure that includes feature importance scores (e.g., Fisher scores) and KNN accuracy:
       $f(\mathbf{S}) = w_1 \cdot \mathcal{A}(\mathbf{S}) + w_2 \cdot \mathcal{F}(\mathbf{S})$
8: Step 3: Equilibrium Pool Formation
9: Form the equilibrium pool with top-performing solutions:
       $\vec{C}_{\text{eq,pool}} = \{\vec{C}_{\text{eq}(1)}, \vec{C}_{\text{eq}(2)}, \vec{C}_{\text{eq}(3)}, \vec{C}_{\text{eq}(4)}, \vec{C}_{\text{eq,ave}}\}$
10: Step 4: Particle Update with Enhanced Dynamics
11: Enhance particle update mechanism:
       $\vec{F} = a_1 \cdot \text{sign}(\vec{r} - 0.5) \cdot \left(e^{-\lambda t} - 1\right) \cdot \text{Adaptive\_Factor}$
12: Step 5: Generation Rate Update
13: Refine generation rate:
       $\vec{G} = \vec{G}_0 \cdot e^{-\lambda(t - t_0)} \cdot \text{Progress\_Factor}$
14: Step 6: Concentration Update with Enhanced Strategies
15: Update each particle's concentration:
       $\vec{C} = \vec{C}_{\text{eq}} + (\vec{C} - \vec{C}_{\text{eq}}) \cdot \vec{F} + \frac{\vec{G}}{\lambda V} \cdot (1 - \vec{F}) \cdot \text{Update\_Strategy}$
16: Step 7: Iteration and Convergence
17: Iterate with enhanced stopping criteria considering convergence in fitness and solution stability
18: Step 8: Feature Selection
19: return The optimal feature subset based on the best-performing particle: $X_{\text{best}}$
```

Algorithm 2. Modified enhanced equilibrium optimizer (EEO) for feature selection

In Algorithm 2, a population of particles is initialized with randomly assigned concentration vector (Line 4). Then the fitness function of each paricle are calculated (Line 7), which includes additional criteria such as feature importance scores (e.g., Fisher scores) and the classification accuracy score from KNN. This combined measure helps to enhance the relevance of the selected features. Where, $f(\mathrm{S})$ represents the fitness function for the selected feature subset S. The term $\mathcal{A}(\mathrm{S}) = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$ denotes the classification accuracy obtained using KNN, with TP, TN, FP, and FN representing True Positives, True Negatives, False Positives, and False Negatives, respectively. The feature importance score $\mathcal{F}(\mathrm{S}) = \sum_{i \in \mathrm{S}} \mathrm{FisherScore}(x_i)$ is the sum of Fisher scores for the selected features. Finally, w_1 and w_2 are the weights assigned to the classification accuracy and feature importance, respectively, where $w_1 + w_2 = 1$ to ensure a balanced influence from both components in the fitness function. After that the Equilibrium Pool is updated (Line 9), where $\vec{C}_{\text{eq,ave}}$ is the arithmetic mean of the four best particles. The particle update mechanism is enhanced with adaptive parameters (Line 11). The exponential term $\vec{F}$, balances exploration and exploitation is defined in Step 4 (Line 11), where a_1 controls exploration ability, $\bar{r}$ is a random vector, t is the iteration index, and $\mathrm{Adaptive\_Factor}$ adjusts dynamically based on the current state of the optimization. $\vec{G}_0$ is the initial generation rate vector, t_0 is the initial time. The generation rate $\vec{G}$ is updated using $\vec{G}_0$ and $\mathrm{Progress\_Factor}$, reflects the algorithm's advancement (Line 13). After updating generation rate, Concentration is updated using the turnover rate λ, the control volume V, and $\mathrm{Update\_Strategy}$ (Line 15). It introduces additional improvements to the update process. After that iterations with enhanced stopping criteria are performed (Line 17) and finally the optimal feature subset with best performance of particles are selected (Line 19).

In the context of enhancing the EO for the purpose of feature selection, we came up with a refined version called EEO. The most relevant advancement encompasses configuring the fitness function to see Fisher Score and the classification accuracy simultaneously achieved by KNN. The main signature of change occurred with the fitness function, EEO is designed to assign high fitness not merely to correct classification but also for high selection frequency. In addition, the Adaptive_Factor in Line No. 11 of Algorithm 2 and Progress_Factor in Line No. 13 of Algorithm 2, being different than the adaptation of EEO, led to the contrasting state of balance between exploration and exploitation levels throughout the optimization process, producing adaptation among its parameters. These amendments tilted the algorithm in favor of efficiently completing optimization toward a more optimal feature subset. By reconstructing concentration updating with an updated method, a refined pairwise equilibrium status of chosen subset features is now realizable. Generation rate updates now follow exponential decay in conjunction with general Update_Strategy in Line No. 15 of Algorithm 2, which shifts dynamically with search advancement. The method of choosing the sixth self-explains its ideal stability, averting premature convergence. These enhancements enable EEO to maintain a robust trade-off between feature reduction and classification accuracy, making it more effective for high-dimensional cybersecurity datasets. We applied the EEO algorithm to both datasets and identified an optimal feature set consisting of 18 features for the UNSW-NB15 dataset and 19 features for the NSL-KDD cybersecurity dataset. The Table 4 displays the chosen attributes of both datasets.

Imbalance issue and SMOTE-IPF

Imbalanced data makes IDS development challenging. Traditional resampling methods often damage the minority class, resulting in poor classification performance. SMOTE and IPF balance better than other approaches[47]. The Algorithm 3 describes about the data balancing using SMOTE-IPF.

	UNSW-NB15		NSL-KDD	
Sl No.	**Index**	**Feature name**	**Index**	**Feature name**
1	1	dur	1	Protocol type
2	2	proto	2	Flag
3	3	dttl	3	Hot
4	4	Service	4	src_bytes
5	5	sttl	5	wrong_fragment
6	6	sinpkt	6	Service
7	7	dinpkt	7	num_failed_logins
8	8	smean	8	num_file_creations
9	9	dmean	9	is_host_login
10	10	response_body_len	10	srv_serror_rate
11	11	ct_dst_ltm	11	rerror_rate
12	12	ct_src_dport_ltm	12	diff_srv_rate
13	13	ct_dst_sport_ltm	13	num_shells
14	14	ct_dst_src_ltm	14	su_attempted
15	15	ct_ftp_cmd	15	dst_host_same_src_port_rate
16	16	ct_flw_http_mthd	16	root_shell
17	17	ct_src_ltm	17	srv_count
18	18	is_sm_ips_ports	18	srv_rerror_rate
19			19	serror_rate

Table 4. Selected features in datasets using the proposed EEO algorithm.

```
Require: Imbalanced dataset $D$, number of nearest neighbors $k$, oversampling rate $r$
Ensure: Balanced dataset $D'$
 1: Initialize minority class instances $M$
 2: Initialize majority class instances $N$
                                                        ▷ SMOTE Phase
 3: for each instance $x \in M$ do
 4:     Find $k$ nearest neighbors of $x$ within $M$
 5:     for each neighbor $x_{\text{nn}}$ do
 6:         Generate synthetic sample using:
                    $x_{\text{syn}} = x + \delta \cdot (x_{\text{nn}} - x)$
 7:         where $\delta \sim \text{Uniform}(0, 1)$
 8:         Add $x_{\text{syn}}$ to $D$
 9:     end for
10: end for
                                                        ▷ IPF Phase
11: Partition dataset $D$ into training and testing sets
12: Train k-NN classifier on the training set
13: repeat
14:     Classify instances in the testing set using the k-NN classifier
15:     Identify and remove misclassified instances from $D$
16:     Re-train k-NN classifier on the updated training set
17: until convergence criteria met
18: return Balanced dataset $D'$
```

Algorithm 3. SMOTE-IPF for balancing imbalanced datasets

The SMOTE-IPF process involves two parts. SMOTE interpolates minority instances to create synthetic minority samples in the first phase. Simple oversampling algorithms can overfit, but our strategy reduces the danger. SMOTE relies on careful adjustment of factors like nearest neighbors and oversampling. For good IDS performance, synthetic samples must accurately represent attack data's complicated patterns. The second phase uses IPF to split and filter the dataset. IPF iteratively filters out instances that have been misclassified by a classifier, usually *K*-NN. Filtering is necessary to avoid overfitting to irrelevant patterns and noise. Iterative refining improves dataset purity and creates more robust and generalizable classification models. This dual-phase technique balances the dataset and prevents synthetic samples from creating noise that degrades the model. According to empirical research, SMOTE-IPF outperforms both SMOTE-ENN and regular SMOTE. Our studies with the UNSW-NB15 and NSL-KDD datasets showed that precision, recall, and F1-Score all got a lot better. This demonstrates the ability of SMOTE-IPF to handle unbalanced cybersecurity data.

Stacking model

Stacking, also known as stacked generalization, is an ensemble machine learning technique designed to enhance predictive performance by combining multiple models. Unlike other ensemble methods like bagging and boosting, this technique involves training a meta-learner to integrate the predictions of base learners, thereby leveraging their individual strengths and compensating for their weaknesses. In the proposed method, we employ a stacking ensemble approach that leverages the strengths of the J48 decision tree algorithm and the ExtraTreeClassifier. The stacking technique involves two layers of models: the base layer and the meta layer.

Base layer: J48 classifiers

The base layer consists of multiple instances of the J48 algorithm[48], a widely-used implementation of the C4.5 decision tree. Given a dataset *D* with *n* samples, where *X* represents the feature matrix and *y* represents the target vector, the process is as follows:

1. **Training base models**: For each base model J_i in the set of J48 classifiers $\{J_1, J_2, \ldots, J_k\}$:

$$\hat{y}^{(i)} = J_i(X)$$

Here, $\hat{y}^{(i)}$ denotes the predicted output of the *i*-th J48 classifier.

2. **Generating meta-features**: The predictions from these J48 classifiers form a new dataset *Z*, which will be used as input for the meta-classifier:

$$Z = \left[\hat{y}^{(1)}, \hat{y}^{(2)}, \ldots, \hat{y}^{(k)}\right]$$

Z is an $n \times k$ matrix containing the predictions of the *k* base models.

Meta layer: ExtraTreeClassifier
The meta layer uses the ExtraTreeClassifier[49], an ensemble method that creates multiple randomized decision trees to improve accuracy and control overfitting.

1. **Training meta-classifier**: The meta-classifier *E* is trained on the new feature matrix *Z* and the original target vector *y*:

$$E(Z) \rightarrow \hat{y}_{\text{meta}}$$

Here, $\hat{y}_{\text{meta}}$ represents the final predictions from the ExtraTreeClassifier.

2. **Final prediction**: During the prediction phase, the base models (J48 classifiers) first generate their predictions, which are then input into the trained ExtraTreeClassifier to produce the final output.

Let $J_i(x)$ denote the *i*-th J48 base model, and *E*(*z*) denote the ExtraTreeClassifier, where *z* is the vector of base model predictions. The final prediction $\hat{y}$ for a new instance *x* is calculated as follows:

1. **Base model predictions**:

$$z = [J_1(x), J_2(x), \ldots, J_k(x)]$$

2. **Meta model prediction**:

$$\hat{y} = E(z)$$

By stacking the predictions from the J48 classifiers and using the ExtraTreeClassifier as the meta-learner, our method aims to combine the interpretability and simplicity of decision trees with the enhanced predictive power of an ensemble method. The J48 classifiers, as base learners, capture diverse patterns in the data, while the ExtraTreeClassifier integrates these patterns, leading to a model that is both accurate and robust against overfitting. This combination ensures improved generalization and overall performance on various types of datasets.

Performance evaluation
This section performs an overview of system configuration for performing our methodology along with the description of the experimental datasets. Next, we discuss the results obtained for the proposed model and provide an indepth analysis. We have compare and contrast our approach with some state of the art models in IDS.

Parameter tuning process using GridSearchCV
The hyperparameter grid used to tune various ML models via GridSearchCV is presented in Table 5. Finding the best combination of hyperparameters is essential to enhance your model's performance, integrate generalization, and avert overfitting. For the J48 and ExtraTreeClassifier algorithms, parameters such as 'criterion', 'max_depth', and 'min_samples_split' were finetuned in order to balance the trade-off between accuracy and computational performance. For maximizing performance, tuning such factors such as square, 'n_neighbors', 'weights', and 'algorithm' for the KNN were considered. The EEO parameters, including population size, max iterations, and update strategies, were fine-tuned to ensure effective feature selection. In the final selected values, we established a totally empirical approach to balance accuracy, computational cost, and stability. With this selection in place, the application framework faced advanced classifiers and outperformed the best results on the datasets. The system configuration for experimental set up of the proposed methodology is shown in Table 6.

Cybersecurity dataset
Malware, DoS, SQL injections, and phishing attacks are common in cybersecurity datasets, showing how cyber threats change. Researchers use popular datasets like NSL-KDD, UNSW-NB15, and CICIDS2017 to test their algorithms' real-world threat detection capabilities. This lets them analyze their IDS's accuracy and efficiency. We tested the UNSW-NB15 and NSL-KDD datasets in our research. NSL-KDD and UNSW-NB15 were chosen as test datasets because they were popularly used as benchmark datasets in IDS research, so that the results could be compared directly[11,18,50–53]. NSL-KDD is an improved version of the KDD'99 dataset, which tackles redundancy and class-imbalance issues while accommodating four primary attack categories, namely DoS, Probe, R2L, and U2R. This highlights that it is a well-structured dataset to assess IDS effectiveness; on the other hand, UNSW-

Model	Hyperparameter	Search space	Selected value
J48	Criterion	{Gini, entropy}	Entropy
	Splitter	{Best, random}	Best
	max_depth	{None, 10, 20, 30}	20
	min_samples_split	{2, 5, 10}	5
	min_samples_leaf	{1, 2, 4}	2
ExtraTreeClassifier	n_estimators	{50, 100, 200}	50
	max_features	{Auto, sqrt, log2}	sqrt
	max_depth	{None, 10, 20, 30}	10
	min_samples_split	{2, 5, 10}	5
	min_samples_leaf	{1, 2, 4}	4
KNN	n_neighbors	{3, 5, 7, 9}	5
	Weights	{Uniform, distance}	Distance
	Algorithm	{Auto, ball_tree, kd_tree, brute}	Auto
	Leaf_size	{10, 20, 30, 40}	30
	p	{1 (Manhattan), 2 (Euclidean)}	2 (Euclidean)
EEO	Population size	{20, 50, 100}	50
	Max iterations	{50, 100, 200}	100
	Adaptive_factor (α)	{0.1, 0.5, 0.9}	0.5
	Progress_factor (β)	{0.05, 0.1, 0.2}	0.1
	Update_strategy	{Linear, exponential, adaptive}	Adaptive

Table 5. Hyperparameter grid for tuning using GridSearchCV.

Component	Specification
Hardware configuration	
Processor	Intel Core i7-12700K @ 3.6 GHz (12 cores, 20 threads)
RAM	32 GB DDR4
GPU	NVIDIA RTX 3080 (10GB VRAM)
Storage	1 TB NVMe SSD
Software configuration for proposed model	
Operating system	Ubuntu 20.04 LTS
Programming language	Python 3.8
ML frameworks	TensorFlow 2.4.1, Scikit-Learn 0.24
Libraries	NumPy, Pandas, Matplotlib, Seaborn

Table 6. Experimental setup configuration.

NB15 is a relatively recent and realistic dataset that encompasses modern cyber threats, such as Generic, Exploits, Fuzzers, DoS, Reconnaissance, Analysis, Backdoor, Shellcode, Worms, reflecting contemporary attack patterns. Additionally, NSL-KDD provides a discriminant and balanced dataset; in contrast, UNSW-NB15 incorporates many different features extracted from real-world network traffic to make sure that the IDS is tested under both artificial and real-world conditions. The use of these two datasets guarantees a thorough evaluation of the proposed model across various kinds of attack scenarios, with a nutshell aim of reducing overfitting to the single dataset structures. Such an approach ensures strength to the generalizability and robustness of our results by substantiating the model, which happens to be effective across different networking environments.

NSL-KDD dataset
The NSL-KDD dataset is an improved version of the original KDD Cup 1999 dataset, which was used for evaluating network IDS. It is available publicly and has become one of the most widely used datasets in IDS research. The NSL-KDD database contains 125,973 cases in the training set and 22,544 cases in the testing set. In total, there are 41 characteristics which signify different attributes of the network traffic including protocol type, service type, flag and multiple calculated network statistics like the count of failed login attempts and connection duration. The dataset is divided into four attack categories: Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R), alongside a class for normal traffic. The NSL-KDD dataset addresses some of the issues in the original KDD dataset, such as redundant records and class imbalance, which improves the evaluation of IDS models.

UNSW-NB15 dataset
The UNSW-NB15 dataset has been developed by the University of New South Wales in the context of a cybersecurity research project in support of IDS development. This data set is considered a public domain and has been specifically designed for evaluating ML models to detect network intrusions of various attack classes. The dataset contains data streams of real-world traffic and attacks through the hybridization of modern attack techniques. Of the many characteristics exhibited by the traffic types, the dataset consists of 2,540,044 instances, each associated with 49 features. For example, both categorical and continuous data describe a large variety of these network traffic attributes. Among these features are flow-based statistics, basic packet features, and statistical characteristics like average packet size and connection duration. The dataset is divided into ten classes: nine attack classes (analysis, worm, exploits, reconnaissance, backdoor, shellcode, DoS, fuzzers, and generic) and one for normal traffic. The wide variety of attacks present should enable a comprehensive evaluation of IDS models.

Pre-processing of NSL-KDD and UNSW-NB15
Missing value imputation: The 'service' attribute of UNSW-NB15 dataset got KNN impute by calculating the distance between that incomplete instance and all complete instances. Having found the 'k' nearest neighbors, we then impute the absent value by averaging neighbor values in case of numerical features or utilizing majority voting in categorical attributes. Table 3 provides the filling results for the 'Service' attributes in the UNSW-NB15 dataset. There are no missing values in the NSL-KDD dataset.

Label encoding: The UNSW-NB15 and NSL-KDD datasets include categorical features with nominal values: "service", "proto" and "state", for the first and "Protocol_Type", "Service", and "Flag", for the second. To convert these categorical variables into a usable form for ML algorithms, label encoding was performed upon them, so that each individual nominal value could be expressed in a binary format that may then be numerically processed by algorithms concerned with categorical information.

Normalization of datasets: Normalization with a standard scaler adjusts the features of the UNSW-NB15 and NSL-KDD datasets to have a mean of zero and a standard deviation of one. This ensures that the proposed model does not remain bias to larger values.

Performance metrics
Accuracy, precision, recall, and F1 score are essential metrics for assessing the effectiveness of anomaly detection in IDS.

$$\text{Accuracy} = \frac{\text{Count of accurately categorized instances}}{\text{Total number of instances}} \times 100\%$$

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = v\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Results and discussion
In this study, we initially applied the EO and the proposed EEO on raw datasets to assess the impact of the optimization techniques on the results. Then we applied SMOTE-IPF for data balancing. We evaluated multiple ML algorithms, including KNN, DT, J48, and ExtraTreeClassifier, as classification models for both the UNSW-NB15 and NSL-KDD datasets. We conducted experiments across four data scenarios: non-optimized imbalanced data (Case 1), non-optimized balanced data (Case 2), optimized imbalanced data (Case 3), and optimized balanced data (Case 4). The results indicated an improvement in accuracy when using EEO for feature selection compared to EO, as demonstrated in Tables 7 and 8. Consequently, EEO was selected as the feature selection method for all subsequent experiments. Before applying SMOTE-IPF, the NSL-KDD and UNSW-NB15 datasets exhibited severe class imbalance, leading to poor classification performance for minority attack classes, particularly U2R, R2L, Backdoor, Shellcode, and Worms. Tables 9 and 10, shown below presents the accuracy of different attack classes before and after applying SMOTE-IPF for UNSW-NB15 and NSL-KDD dataset respectively.

Cases	EO				EEO			
Models	DT	KNN	ExtraTree classifier	J48	DT	KNN	ExtraTree classifier	J48
Case 1	91.8	90.2	93.9	92.8	93.1	91.6	94.7	93.5
Case 2	92.8	91.5	96.4	93.8	94.5	93.3	97.0	94.8
Case 3	89.7	88.2	92.2	90.5	90.8	89.7	92.9	91.0
Case 4	90.7	88.9	95.3	92.3	92.6	90.1	96.1	92.9

Table 7. Accuracy (%) using EO and EEO for different models on various cases on UNSW-NB15 dataset. Case 1: Non-optimized imbalanced data; Case 2: Non-optimized balanced data; Case 3: Optimized imbalanced data; Case 4: Optimized balanced data

Cases	EO				EEO			
Models	DT	KNN	ExtraTree classifier	J48	DT	KNN	ExtraTree classifier	J48
Case 1	92.7	90.9	93.5	92.8	93.3	91.6	94.7	94.7
Case 2	93.9	93.0	95.7	95.5	94.8	93.6	97.2	97.2
Case 3	98.2	96.5	98.1	97.6	98.7	97.4	98.7	98.7
Case 4	93.1	93.1	98.8	97.7	99.3	93.5	99.6	99.6

Table 8. Accuracy (%) using EO and EEO for different models on various cases on NSL-KDD dataset. Case 1: Non-optimized imbalanced data; Case 2: Non-optimized balanced data; Case 3: Optimized imbalanced data; Case 4: Optimized balanced data

Type of attack	Accuracy before SMOTE	Accuracy after SMOTE-IPF	Improvement
Normal (Benign)	98.39	98.67	+ 0.28
Generic	98.16	99.12	+ 0.96
Exploits	97.95	98.02	+ 0.07
Fuzzers	97.23	89.45	− 7.78
DoS	96.52	99.47	+ 2.95
Reconnaissance	96.13	97.34	+ 1.21
Analysis	86.19	94.53	+ 8.34
Backdoor	84.83	97.42	+ 12.59
Shellcode	78.64	93.45	+ 14.81
Worms	30.77	95.62	+ 64.85

Table 9. Comparison of accuracy before and after data balancing on UNSW-NB15. We can observe that the minority class like Backdoor, Shellcode, Worms have shown significant improvement once the proposed architecture is applied. All values are in (%)

Type of attack	Accuracy before SMOTE	Accuracy after SMOTE-IPF	Improvement
Normal	98.3	99.7	+ 1.4
U2R	45.6	98.5	+ 52.9
R2L	52.1	98.7	+ 46.6
DoS	97.2	99.74	+ 2.54
Probing	96.8	99.64	+ 2.84

Table 10. Comparison of accuracy before and after sampling on NSL-KDD. We can observe that the minority class like U2R, R2L have shown significant improvement once the proposed architecture is applied. All values are in (%)

We constructed six stacking models by combining the four base machine learning models—DT (Model 1), KNN (Model 2), ExtraTreeClassifier (Model 3), and J48 (Model 4)—in various configurations. The first stacking model, referred to as Model 5, used KNN as the base classifier and DT as the meta-classifier. Model 6 utilized DT as the base classifier and ExtraTreeClassifier as the meta-classifier. In Model 7, DT was used as the base classifier and J48 as the meta-classifier. Model 8 employed KNN as the base classifier and ExtraTreeClassifier as the meta-classifier. For Model 9, KNN was used as the base classifier and J48 as the meta-classifier. Finally, Model 10 combined J48 as the base classifier with ExtraTreeClassifier as the meta-classifier. The highest performance for both datasets was achieved with the combination of EEO for feature selection, SMOTE-IPF for data balancing, and the sixth stacking model (Model 10), which used J48 as the base classifier and ExtraTreeClassifier as the meta-classifier. The results of our analysis for all models across the four cases (Case 1, Case 2, Case 3, and Case 4) are illustrated in Figs. 2 and 3 for the UNSW-NB15 and NSL-KDD datasets, respectively. Among all the evaluated cases, Model 10 showed the best performance in Case 4, which involved optimized and balanced datasets, as highlighted in Fig. 4. The results for the stacking models are provided in Table 13. Tables 11 and 12 shows the classwise perfromence for NSL-KDD and UNSW-NB15 dataset where Table 13 shows overall accuracy, precision, recall and F1 score for both datasets.

Many ML algorithms have been employed for intrusion detection on the NSL-KDD dataset, which have provided excellent results on accuracy, precision, recall, and F1-score. Abbas et al.[54] used a hybrid ensemble model with random forest-recursive feature elimination. They did not consider balancing the data before the classification process, which might induce bias against the model, particularly in identifying minority attack classes. As a result, this could possibly lower the detection efficacy on less frequent but crucial attack types.

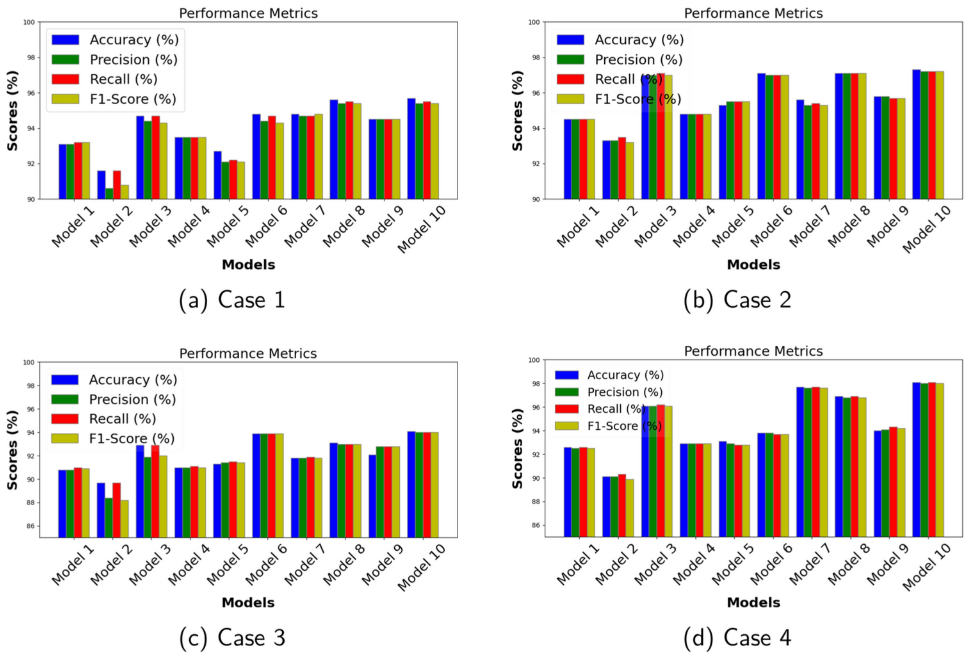


Fig. 2. Performance analysis of different models on different cases for UNSW-NB15 dataset. Model 1: DT; Model 2: KNN; Model 3: ExtraTreeClassifier; Model 4: J48; Model 5: stacking model with KNN as base classifier and DT as meta classifier; Model 6: stacking model with DT as base classifier and ExtraTreeClassifier as meta classifier; Model 7: stacking model with DT as base classifier and J48 as meta classifier; Model 8: stacking model with KNN as base classifier and ExtraTreeClassifier as meta classifier; Model 9: stacking model with KNN as base classifier and J48 as meta classifier; Model 10: stacking model with J48 as base classifier and ExtraTreeClassifier as meta classifier. Case 1: non-optimized imbalanced data; Case 2: non-optimized balanced data; Case 3: optimized imbalanced data; Case 4: optimized balanced data.

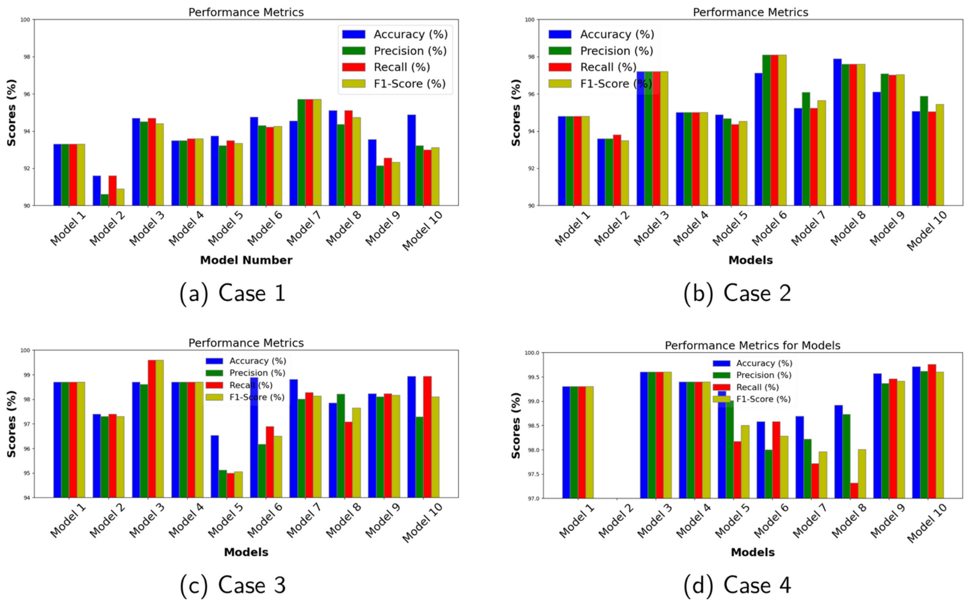


Fig. 3. Performance analysis of different models on different cases for NSL-KDD dataset. Model 1: DT; Model 2: KNN; Model 3: ExtraTreeClassifier; Model 4: J48; Model 5: stacking model with KNN as base classifier and DT as meta classifier; Model 6: Stacking model with DT as base classifier and ExtraTreeClassifier as meta classifier; Model 7: stacking model with DT as base classifier and J48 as meta classifier; Model 8: stacking model with KNN as base classifier and ExtraTreeClassifier as meta classifier; Model 9: stacking model with KNN as base classifier and J48 as meta classifier; Model 10: stacking model with J48 as base classifier and ExtraTreeClassifier as meta classifier. Case 1: non-optimized imbalanced data; Case 2: non-optimized balanced data; Case 3: optimized imbalanced data; Case 4: optimized balanced data.

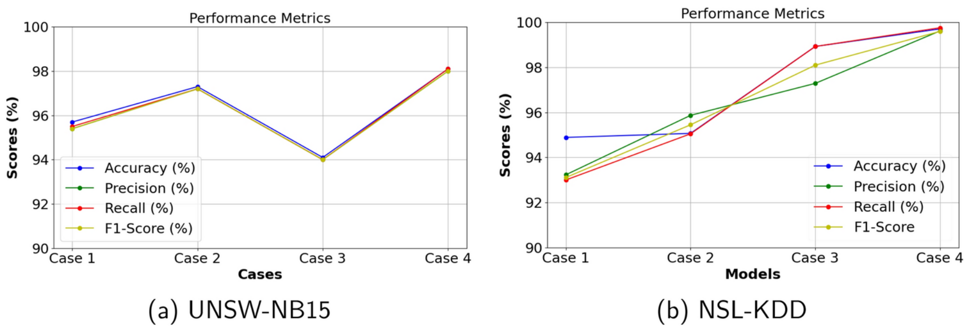


Fig. 4. Performance analysis of proposed model (stacking model of J48 as base classifier and ExtraTreeClassifier as meta classifier) on different cases for UNSW-NB15 and NSL-KDD datasets. Case 1: non-optimized imbalanced data; Case 2: non-optimized balanced data; Case 3: optimized imbalanced data; Case 4: optimized balanced data.

Pokharel et al.[55] employed the Naive Bayes classifier and SVM for attack detection inside a network. Their model had a long training time, mainly focusing on reducing false positives rather than maximizing detection speed; consequently, their method cannot be applied in real-time threat-monitoring scenarios. Bong et al.[56] used Gaussian NB models. Their analysis was limited to a small portion of the NSL-KDD dataset, looking at only a limited number of selected attack types from the dataset, consequently limiting the exposure of the model to the entire range of possible network intrusions and possibly risking generalizability against a broader threat landscape. Das et al.[44] have executed a supervised Ensemble ML framework (SupEnML) by uniting various ML classifiers belonging to different classification families like DT, Logistic regression, Naive Bayes, Neural Network, and SVM for IDS. While the classifiers performed well, their training time was high and also the consumption of memory was huge due to the complexity of implementing multiple classifiers, which generate a computational overhead on practical deployment in resource-starved network environments. In addition, Meliboev et al.[57] put their weight on DL through CNN-LSTM for the extraction of hierarchical features for any other type. They only addressed SMOTE, which did not seem to tackle the minor class imbalance dilemma effectively, and their approach included heavy training of 100 epochs with a fixed learning rate, which might prove inefficient for swiftly adapting to diverse network traffic patterns and fresh attack vectors.

The development of intrusion detection in the UNSW-NB15 dataset has been the focus of many works. Dora et al.[58] have employed the K Best for feature selection. For classification, they have used the ML algorithms such as RF, XGBoost, and DT. However, the usage of traditional DL and insider-optimized LSTM tends to put heavy burdens on computational resources for training and detections at unchecked rates, thereby possibly invalidating its greater accuracies in constrained-resource environments. They also gave importances on finding some specific type of attacks. Kasongo et al.[18] implemented XGBoost for feature selection, then applied the SVM and ANN algorithms to a small binary dataset possessing an accuracy of 0.77%. Their oversimplified binary classification approach impended the reality of the multi-class dimensionality of most modern network attack classes, with far less correct classification to illustrate the feature representation or model selection firm as a limitation. Yin et al.[17] presented a hybrid feature selection technique in which the hybrid ranking relates Information Gain and RF feature selection to the respective classifier, followed later by MLP, showing much higher classification accuracy. Although the feature selection fared well in this context, the MLP model scored only 84.24% accuracy in multi-class intrusion detection, pointing to the severe challenge in distinguishing between attack types that look very much the same and may overfit due to feature reduction.

Ethala et al.[19] also pointed out a proposed model for hybrid approach combining SMO with HPSO to handle the high data size more efficiently in an IDS. Their quite complex optimization heavily relies on high processor resources during the training phase and such effect could heavily restrict real-time implementations and adaptability in addressing evolved network traffic patterns. Similarly, with the concept of RF, Ahmad et al.[59] corroborated somewhat of the contributions their RF-based cluster-based model was able to offer-regrettably with inconsistencies across different feature subsets, for their classifier to achieve just 91.4% accuracy with the TCP features, hence, largely speaking to the challenge posed by the need for different traffic-type resolutions apart from attack vectors the model has ever perceived. Our model outperforms all the state of the art approaches and introduces a novel approach to IDS research domain that addresses several critical limitations identified in previous studies. Feature selection, which was done with EEO, resulted in a reduced multi-dimensionality without losing the necessary attack features. By reducing the number of irrelevant and redundant features, it helps to extend beyond classic ML models due to different accurate selections, overcoming the feature redundancy issues faced in Abbas et al.'s[54] work and the suboptimal feature selection seen in Bong et al.'s[56] limited dataset analysis. Unlike Kasongo et al.'s[18] XGBoost selection that achieved only 0.77% accuracy on UNSW-NB15, EEO's nature-inspired optimization capability identifies optimal feature subsets that capture the essential characteristics of various attack patterns.

The data balancing problem for minor attack categories, completely ignored by Abbas et al.[54] and inadequately addressed by Meliboev et al.[57] using only SMOTE, is comprehensively resolved in our proposed method through SMOTE-IPF. Before applying SMOTE-IPF, the NSL-KDD and UNSW-NB15 datasets exhibited severe class imbalance, leading to poor classification performance for minority attack classes, particularly

Dataset		UNSW-NB15				NSL-KDD			
Model	**Case No.**	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Model 1 (DT)	Case 1	93.1	93.1	93.2	93.2	93.3	93.3	93.3	93.3
	Case 2	94.5	94.5	94.5	94.5	94.8	94.8	94.8	94.8
	Case 3	90.8	90.8	91.0	90.9	98.7	98.7	98.7	98.7
	Case 4	92.6	92.5	92.6	92.5	99.3	99.3	99.3	99.3
Model 2 (KNN)	Case 1	91.6	90.6	91.6	90.8	91.6	90.6	91.6	90.9
	Case 2	93.3	93.3	93.5	93.2	93.6	93.6	93.8	93.5
	Case 3	89.7	88.4	89.7	88.2	97.4	97.3	97.4	97.3
	Case 4	90.1	90.1	90.3	89.9	93.5	94.4	93.5	93.4
Model 3 (ExtraTree classifier)	Case 1	94.7	94.4	94.7	94.3	94.7	94.5	94.7	94.4
	Case 2	97.0	97.0	97.1	97.0	97.2	97.2	97.2	97.2
	Case 3	92.9	91.9	92.9	92.0	98.7	98.6	99.6	99.6
	Case 4	96.1	96.1	96.2	96.1	99.6	99.6	99.6	99.6
Model 4 (J48)	Case 1	93.5	93.5	93.5	93.5	94.7	94.5	94.7	94.4
	Case 2	94.8	94.8	94.8	94.8	97.2	97.2	97.2	97.2
	Case 3	91.0	91.0	91.1	91.0	98.7	98.6	99.6	99.6
	Case 4	92.9	92.9	92.9	92.9	99.6	99.6	99.6	99.6
Model 5 (Stacking: KNN base, DT meta)	Case 1	92.7	92.1	92.2	92.1	93.75	93.22	93.5	93.35
	Case 2	95.3	95.5	95.5	95.5	94.88	94.68	94.37	94.52
	Case 3	91.3	91.4	91.5	91.4	96.54	95.12	95.0	95.05
	Case 4	93.1	92.9	92.8	92.8	99.21	99.01	98.17	98.5
Model 6 (Stacking: DT base, ExtraTree classifier meta)	Case 1	94.8	94.4	94.7	94.3	94.75	94.3	94.22	94.25
	Case 2	97.1	97.0	97.0	97.0	97.13	98.09	98.09	98.09
	Case 3	93.9	93.9	93.9	93.9	98.89	96.17	96.89	96.5
	Case 4	93.8	93.8	93.7	93.7	98.58	98.0	98.58	98.28
Model 7 (Stacking: DT base, J48 meta)	Case 1	94.8	94.7	94.7	94.8	94.55	95.71	95.71	95.71
	Case 2	95.6	95.3	95.4	95.3	95.23	96.09	95.23	95.65
	Case 3	91.8	91.8	91.9	91.8	98.81	98.01	98.27	98.13
	Case 4	97.7	97.6	97.7	97.6	98.69	98.22	97.72	97.96
Model 8 (Stacking: KNN base, ExtraTreeClassifier meta)	Case 1	95.6	95.4	95.5	95.4	95.11	94.37	95.11	94.73
	Case 2	97.1	97.1	97.1	97.1	97.9	97.61	97.61	97.61
	Case 3	93.1	93.0	93.0	93.0	97.86	98.22	97.09	97.65
	Case 4	96.9	96.8	96.9	96.8	98.92	98.73	97.32	98.01
Model 9 (Stacking: KNN base, J48 meta)	Case 1	94.5	94.5	94.5	94.5	93.55	92.14	92.56	92.34
	Case 2	95.8	95.8	95.7	95.7	96.1	97.08	97.01	97.04
	Case 3	92.09	92.8	92.8	92.8	98.23	98.1	98.23	98.16
	Case 4	94.0	94.1	94.3	94.2	99.57	99.37	99.46	99.41
Model 10 (Stacking: J48 base, ExtraTreeClassifier meta) (Proposed model)	Case 1	95.7	95.4	95.5	95.4	94.89	93.23	93.0	93.11
	Case 2	97.3	97.2	97.2	97.2	95.07	95.87	95.05	95.45
	Case 3	94.1	94.0	94.0	94.0	98.93	97.29	98.93	98.1
	Case 4	**98.1**	**98.0**	**98.1**	**98.0**	**99.71**	**99.62**	**99.76**	**99.6**

Table 13. Performance metrics for models 1–10 on UNSW-NB15 and NSL-KDD datasets. Where, **Case 1**: Non-optimized imbalanced data; **Case 2**: Non-optimized balanced data; **Case 3**: Optimized imbalanced data; **Case 4**: Optimized balanced data. And **Model 1**: DT; **Model 2**: KNN; **Model 3**: ExtraTreeClassifier; **Model 4**: J48; **Model 5**: Stacking model of KNN as base classifier and DT as meta classifier; **Model 6**: Stacking model of DT as base classifier and ExtraTreeClassifier as meta classifier; **Model 7**: Stacking model of DT as base classifier and J48 as meta classifier; **Model 8**: Stacking model with KNN as base classifier and ExtraTreeClassifier as meta classifier; **Model 9**: Stacking model with KNN as base classifier and J48 as meta classifier; **Model 10**: Stacking model with J48 as base classifier and ExtraTreeClassifier as meta classifier. All values are in (%). Significant values are in bold.

U2R, R2L, Backdoor, Shellcode, and Worms. This dual-phase technique balances the dataset and prevents synthetic samples from creating noise that degrades the model. According to empirical research, SMOTE-IPF outperforms both SMOTE-ENN and regular SMOTE. Our stacking model architecture, utilizing J48 as the base classifier and ExtraTreeClassifier as the meta classifier, creates a powerful ensemble that outperforms the simpler approaches like Pokharel et al.'s[55] hybrid classifier and Yin et al.'s[17] MLP implementation. This multi-level

Attack class	Accuracy	Precision	Recall	F1-Score
Normal	99.69	99.7	99.44	99.5698
U2R	98.45	98.22	84.37	90.7697
R2L	98.68	98.02	89.11	93.3529
DoS	99.69	99.64	99.35	99.4948
Probing	99.61	99.65	99.00	99.3239

Table 11. Performance metrics for different attack classes of NSL-KDD Dataset where we used EEO for feature selection, SMOTE-IPF for data balancing and Stacking ML (J48 as base classifier and ExtraTreeClassifier as meta classifier) for classification. All values are in (%)

Type of attack	Accuracy	Precision	Recall	F1-Score
Normal	98.67	98.2	97.9	98.0498
Generic	99.12	99.62	99.76	99.6899
Exploits	98.02	97.1	97.2	97.1499
Fuzzer	89.45	88.55	86.09	87.3027
DoS	99.47	98.44	98.6	98.5199
Reconnaissance	97.34	95.5	96.8	96.1456
Analysis	94.53	94.5	94.1	94.2996
Backdoor	97.42	96.63	96.4	96.5149
Shellcode	93.45	93.22	93.52	93.3698
Worm	95.62	94.6	93.7	94.1478

Table 12. Performance metrics for different attack classes of UNSW-NB15 Dataset where we used EEO for feature selection, SMOTE-IPF for data balancing and Stacking ML (J48 as base classifier and ExtraTreeClassifier as meta classifier) for classification. All values are in (%)

Models	Accuracy	Precision	Recall	F1-Score
RF-RFE_ML ensemble[54]	99.53	99.79	99.78	99.29
Naive Bayes classifier + SVM[55]	85.51	97.62	68.90	80.78
Gaussian NB[56]	94.529	84.631	92.026	88.17
SupEnML[44]	88.1	95.9	82.6	88.7
Combination of CNN and LSTM[57]	82.6	94.9	68.9	79.8
Proposed model	**99.7**	**99.6**	**99.6**	**99.6**

Table 14. Comparison of NSL-KDD dataset result with proposed and existing models. All values are in (%). Significant values are in bold.

learning framework leverages the complementary strengths of decision tree-based algorithms while mitigating their individual weaknesses. Unlike Das et al.'s[44] ensemble approach that suffered from high computational overhead, our stacking model achieves superior performance (99.7% accuracy on NSL-KDD and 98.1% on UNSW-NB15) with lower computational requirements by focusing on two strategically selected complementary classifiers rather than combining many different classification families.

These results improve our approach towards understanding the immense ensemble potential of these three components for feature selection, feature representation, error-driven classification, and decision-based mechanisms during data imbalance handling. Now, this work is instrumental in understanding a prominent ensemble interaction, it offers the full range of possibility or alternatives to create a better or more diversified learner for a wider range of divergent instances, especially under the umbrella of attacking network-like data in the domain of network security. Such considerations enforce us to look beyond seeking best opportunities toward developing diverse options with which to improve one's probability. The detailed performance comparison is presented in Tables 14 and 15, while Fig. 5 provides a comparative analysis with baseline models across both datasets.

The confusion matrix results obtained using the proposed methodology, which incorporates EEO for feature selection, SMOTE-IPF for data balancing, and a stacking model with J48 as the base classifier and ExtraTreeClassifier as the meta-classifier, demonstrate a significant improvement in classification accuracy as shown in Fig. 6. When applied to the NSL-KDD and UNSW-NB15 datasets, the approach effectively enhances the detection of cyberattacks, showing a better balance between precision, recall, and F1-score compared to baseline models. The AUC-ROC curves were obtained using the proposed methodology, which incorporates

Models	Accuracy	Precision	Recall	F1-Score
KBest_RF+XGBoost+ DT[58]	93.7	94.5	90.2	92.29
ANN-XGBoost[18]	77.51	79.5	77.53	77.28
IGRF-REE-MLP[17]	84.24	82.22	83.22	82.85
SMO-HPSO[19]	94.12	94.19	93.32	94.12
RF[59]	97.37	96.13	96.11	96.11
Proposed model	**98.1**	**98**	**98.1**	**98**

Table 15. Comparison of UNSW-NB15 dataset result with proposed and existing models. All values are in (%). Significant values are in bold.

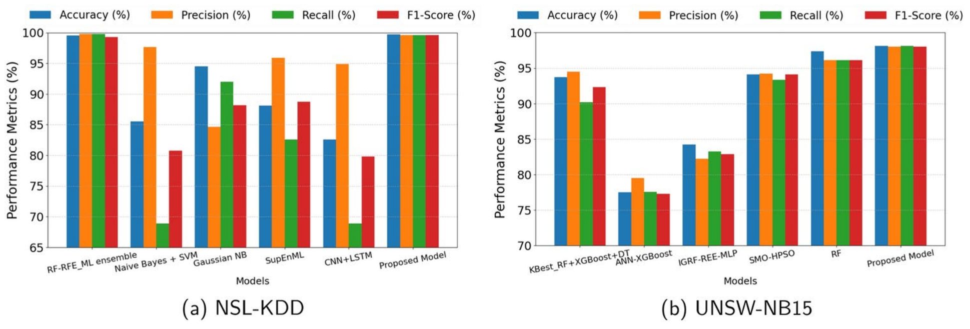


Fig. 5. Comparison of Proposed method with existing methods[44,54–57] over NSL-KDD dataset and state-of-the-art methods[17–19,58,59] over UNSW-NB15 dataset.

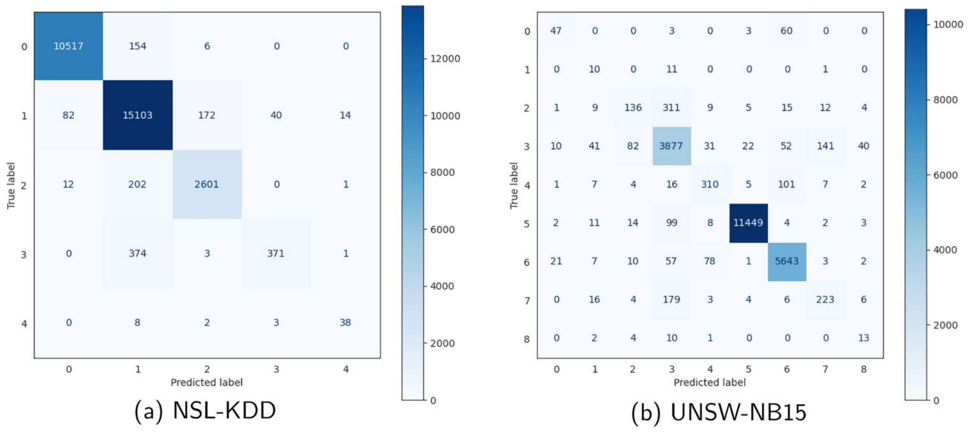
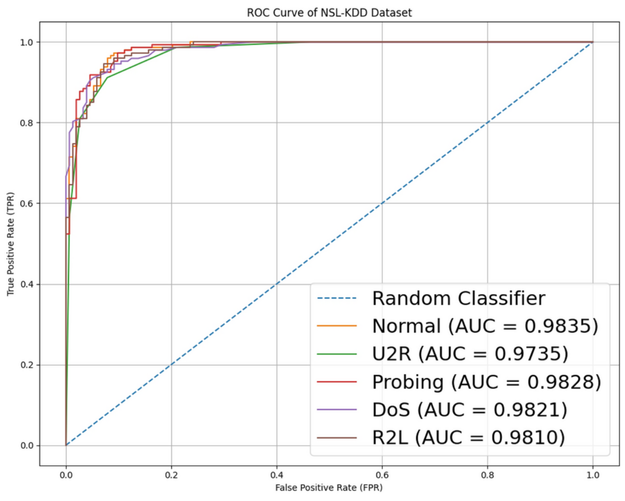


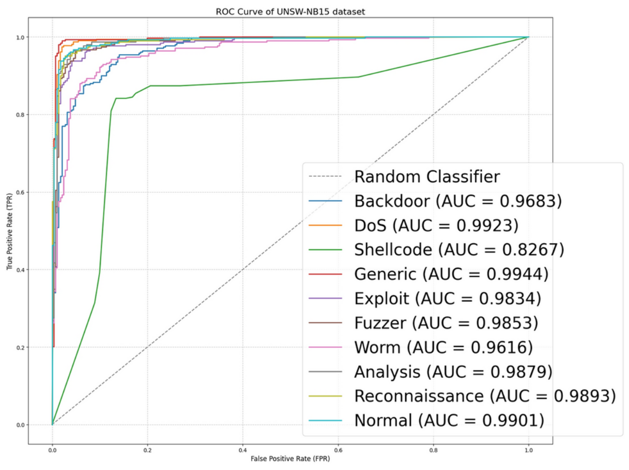
Fig. 6. Result of Confusion Matrix using Proposed Methodology where we used EEO for feature selection, SMOTE-IPF for data balancing and stacking model of J48 as base classifier and ExtraTreeClassifier as meta classifier for classification on NSL-KDD and UNSW-NB15 datasets.

EEO for feature selection, SMOTE-IPF for data balancing, and a stacking model comprising J48 as base classifier and ExtraTreeClassifier as the meta-classifier. This approach was applied to both the NSL-KDD and UNSW-NB15 datasets to evaluate the classification performance. Figure 7 represents the AUC-ROC curve for both datasets.

The findings of this study underscore the effectiveness of integrating the EEO for feature selection, SMOTE-IPF for data balancing, and stacking ML models in IDS. The EEO, which incorporates the fisher Score, outperformed the standard EO by ensuring that the selected features have strong discriminative capabilities, thereby enhancing the model's robustness and interpretability. The inclusion of accuracy scores from the KNN classifier further validates that the selected features positively influence the classifier's performance, leading to improved predictive accuracy. Additionally, EEO maintains a balance between exploration and exploitation by leveraging the adaptive nature of EO, allowing it to explore diverse feature subsets while concentrating on high-quality solutions. This balance facilitates faster convergence and helps avoid local optima, ultimately improving the overall performance of the model. The use of SMOTE-IPF addressed the significant class imbalance often



(a) NSL-KDD



(b) UNSW-NB15

Fig. 7. Result of AUC-ROC curve using Proposed Methodology where we used EEO for feature selection, SMOTE-IPF for data balancing and stacking model of J48 as base classifier and ExtraTreeClassifier as meta classifier for classification on NSL-KDD and UNSW-NB15 datasets.

present in IDS datasets by not only generating synthetic samples of the minority class but also employing noise-filtering techniques to remove misclassified instances. This dual approach ensures that the data is more balanced and less noisy, allowing ML models to better learn decision boundaries between normal and attack classes, thus reducing the rates of false positives and false negatives. The quality and balance of the data improved through SMOTE-IPF contribute to the robust training of the models, enhancing overall classification performance. The stacking of ML models further amplified the system's effectiveness by combining the strengths of different classifiers into a cohesive framework. By utilizing various combinations of base classifiers, such as KNN, DT, J48, and ExtraTreeClassifier, with a meta-classifier, the stacking approach captured diverse patterns in the data that individual models might miss. This ensemble method reduces the biases and variances associated with single classifiers, leading to a more stable and accurate system. The combined use of EEO for feature selection, SMOTE-IPF for data balancing, and stacking models with J48 as base classifier and ExtraTreeClassifier as meta classifier results in a comprehensive and adaptive framework that significantly improves IDS performance, offering a robust solution that outperforms traditional models in detecting intrusions with high accuracy.

Practical considerations of the proposed IDS

In real-world IDS applications, factors such as latency, model size, and computational cost play a crucial role in determining the feasibility of deployment. The proposed IDS, which integrates feature selection and a stacking-based ML model, is designed to optimize these aspects while maintaining high detection accuracy.

Latency: The usage of proposed EEO for feature selection has led to reduction of features from the original dataset thereby enhancing the computing operations. For the UNSW-NB15 dataset, the proposed model takes 1.6394 s for training purposes while the prediction time stands at 0.3525 s per instance. Similarly, for the NSL-KDD dataset, the training time computed is 0.8560 s while the prediction time is 0.2974 s per instance. Thus it can be perceived that the proposed model has lower latency while performing the intrusion detection.

Model size: We have used the ensemble of J48 combined with ExtraTreeClassifier in our proposed approach with EEO as feature selection. The final model size for the UNSW-NB15 dataset is 5.8905 MB while for the NSL-KDD it is 9.8779 MB. These model sizes are of uncompressed versions. Further compression can be done to make the model further smaller in size.

Computational cost: As a result of the ensemble process involving the feature selection and stacking ML method, our proposed model has the following compute costs for each dataset. The computation for the UNSW-NB15 dataset in terms of memory, CPU is 0.2031 MB, 0.6000% respectively. On the other hand the computation for the NSL-KDD dataset in terms of memory, CPU is 30.0156 MB, 0.3000% respectively. Thus the proposed system can be readily deployed on a system with limited resources also.

Overall complexity analysis

The EEO, a metaheuristic optimization algorithm, has a time complexity of $O(P \cdot I \cdot D)$, where *P* is the number of particles (agents), *I* is the number of iterations, and *D* is the number of features. For SMOTE-IPF, the time complexity is $O(N \cdot k)$ for the Synthetic Minority Over-sampling Technique (SMOTE), where *N* is the number of samples and *k* is the number of nearest neighbors, and $O(I \cdot N)$ for Iterative Partitioning Filter (IPF), where *I* is the number of iterations.

The time complexity for training a stacking model with J48 and ExtraTreeClassifier involves: J48, which has a time complexity of $O(N \cdot D \cdot \log(N))$, where *N* is the number of samples and *D* is the number of features. ExtraTreeClassifier, which has a time complexity of $O(T \cdot N \cdot D \cdot \log(N))$, where *T* is the number of trees.

Combining these complexities, the overall time complexity of our experiment can be approximated as:

$$\text{‰}(P \cdot I \cdot D) + O(N \cdot k) + O(I \cdot N) + O(N \cdot D \cdot \log(N)) + O(T \cdot N \cdot D \cdot \log(N)),$$

providing a comprehensive view of the computational complexity involved in the proposed IDS method. Therefore, the overall time complexity of the experiment is:

$$O(N \cdot D \cdot \log(N)),$$

which simplifies to $O(N \cdot \log(N))$.

Conclusion

This work introduced a novel way of finding intrusions by feature selection through EEO, balancing by SMOTE-IPF, and then applying a stacking ML algorithm with J48 as the base classifier and ExtraTreeClassifier as the meta-classifier. The proposed method was found to perform excellently on two widely used benchmark datasets. Our system has achieved an accuracy of 99.7% for the NSL-KDD dataset with precision, recall, and F1 all at 99.6%. Thus, the results validate our system's potential as an intrusion detector, providing nearly correct predictions with only a few false alarms. A further demonstration of the efficiency of our system can be shown from another evaluation on the UNSW-NB15 dataset, where we achieved an accuracy of 98.1%, precision of 98.0%, recall of 98.1%, and F1 score of 98.0%. These metrics indeed highlight the superiority of our system in diverse and convoluted intrusion scenarios.

Furthermore, our model achieved this high-performance benchmark while reducing prediction time. As such, high-performance streaming IDS is the ideal infrastructure. Integration of EEO, SMOTE-IPF, and the stacking algorithm has found to be the efficacious solution in both efficiency and accuracy in IDS tasks. This research has been significant for cybersecurity by offering a scalable and dependable solution for intrusion detection, thus paving the way to its application in real-world security environments. Our study is focused on two major benchmark data sets, namely NSL-KDD and UNSW-NB15, as they are widely used in various studies and offer a variety of cyber-attacks.

Limitation and future work

Despite strong model performance in most respects, there are some drawbacks associated with the proposed approach. The foremost of them is their dependence on benchmark datasets referring to NSL-KDD and UNSW-NB15. While these are popular datasets, they cannot cover all the possible scenarios and intricacies posed by real-world network environments, thereby reducing the methodological approach's robustness through unknown network traffic patterns unseen in the dataset, differing intrusion signatures potentially unaccounted for. AI models could also work beautifully in the face of constantly changing attacks in shifting environments, though how well they perform remains an uncertainty. Intruders are always altering the attack patterns by coming up with new consolidation and improvement strategies. This would always require continuous learning with dynamic adaptation to provide consistency for top performance. Moving forward, a research question would be to evaluate on real-world network traffic data. These results set the base for research on IDS, making it more versatile for practical use. Another unexplored possibility is advanced optimization techniques that could be used to mother improve the feature selection efficiency and adaptability of the EEO. And finally, an adaptable IDS website should regulate and adjust the values of its variables, depending on the situation, to make the system resilient against advanced cyber-attacks. Therefore, these innovations will make the proposed IDS powerful in this wide spectrum of rapidly-changing security landscapes.

Data availability

Upon a reasonable request, the corresponding author will provide some or all of the data, models, or programs that support the study's conclusions.

Received: 9 December 2024; Accepted: 2 May 2025
Published online: 14 May 2025

References

1. US, D. Annual Cyber Threat Trends Report. https://www2.deloitte.com(2024).
2. Hussain, F., Hussain, R., Hassan, S. A. & Hossain, E. Machine learning in IoT security: Current solutions and future challenges. *IEEE Commun. Surv. Tutor.* **22**(3), 1686–1721 (2020).
3. Nie, L. et al. Intrusion detection for secure social internet of things based on collaborative edge computing: A generative adversarial network-based approach. *IEEE Trans. Comput. Soc. Syst.* **9**(1), 134–145 (2021).
4. Liao, H.-J., Lin, C.-H.R., Lin, Y.-C. & Tung, K.-Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **36**(1), 16–24 (2013).
5. Thakkar, A. & Lohiya, R. Role of swarm and evolutionary algorithms for intrusion detection system: A survey. *Swarm Evol. Comput.* **53**, 100631 (2020).
6. Al-Omari, M., Rawashdeh, M., Qutaishat, F., Alshira'H, M. & Ababneh, N. An intelligent tree-based intrusion detection model for cyber security. *J. Netw. Syst. Manag.* **29**(2), 20 (2021).
7. Sarhan, M., Layeghy, S., Moustafa, N. & Portmann, M. Cyber threat intelligence sharing scheme based on federated learning for network intrusion detection. *J. Netw. Syst. Manag.* **31**(1), 3 (2023).
8. Tsimenidis, S., Lagkas, T. & Rantos, K. Deep learning in IoT intrusion detection. *J. Netw. Syst. Manag.* **30**(1), 8 (2022).
9. Thakkar, A. & Lohiya, R. A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. *Arch. Comput. Methods Eng.* **28**(4), 3211–3243 (2021).
10. Pudjihartono, N., Fadason, T., Kempa-Liehr, A. W. & O'Sullivan, J. M. A review of feature selection methods for machine learning-based disease risk prediction. *Front. Bioinform.* **2**, 927312 (2022).
11. Türk, F. Analysis of intrusion detection systems in UNSW-NB15 and NSL-KDD datasets with machine learning algorithms. *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi* **12**(2), 465–477 (2023).
12. Wang, C., Sun, Y., Wang, W., Liu, H. & Wang, B. Hybrid intrusion detection system based on combination of random forest and autoencoder. *Symmetry* **15**(3), 568 (2023).
13. Zhour, R., Khalid, C., & Abdellatif, K. Hybrid intrusion detection system based on Random forest, decision tree and Multilayer Perceptron (MLP) algorithms. In *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)* 1–5 (IEEE, 2023).
14. Kasongo, S. M. A deep learning technique for intrusion detection system using a recurrent neural networks based framework. *Comput. Commun.* **199**, 113–125 (2023).
15. Khan, N. M., Madhav C. N., Negi, A., & Thaseen, I. S. Analysis on improving the performance of machine learning models using feature selection technique. In *Intelligent Systems Design and Applications: 18th International Conference on Intelligent Systems Design and Applications (ISDA 2018) Held in Vellore, India, December 6–8, 2018 Vol. 2, 69–77* (Springer, 2020).
16. Almomani, O. A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms. *Symmetry* **12**(6), 1046 (2020).
17. Yin, Y. et al. IGRF-RFE: A hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset. *J. Big Data* **10**(1), 1–26 (2023).
18. Kasongo, S. M. & Sun, Y. Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. *J. Big Data* **7**, 1–20 (2020).
19. Ethala, S. & Kumarappan, A. A hybrid spider monkey and hierarchical particle swarm optimization approach for intrusion detection on Internet of Things. *Sensors* **22**(21), 8566 (2022).
20. Abdulganiyu, O. H., Tchakoucht, T. A., Saheed, Y. K. & Ahmed, H. A. XIDINTFL-VAE: XGBoost-based intrusion detection of imbalance network traffic via class-wise focal loss variational autoencoder. *J. Supercomput.* **81**(1), 1–38 (2025).
21. Saheed, Y. K., Kehinde, T. O., Ayobami Raji, M. & Baba, U. A. Feature selection in intrusion detection systems: A new hybrid fusion of Bat algorithm and Residue Number System. *J. Inf. Telecommun.* **8**(2), 189–207 (2024).
22. Saheed, Y. K. & Hamza-Usman, F. E. Feature selection with IG-R for improving performance of intrusion detection system. *Int. J. Commun. Netw. Inf. Secur.* **12**(3), 338–344 (2020).
23. Saheed, Y. K., Abdulganiyu, O. H. & Tchakoucht, T. A. Modified genetic algorithm and fine-tuned long short-term memory network for intrusion detection in the internet of things networks with edge capabilities. *Appl. Soft Comput.* **155**, 111434 (2024).
24. Saheed, Y. K. & Chukwuere, J. E. Xaiensembletl-iov: A new explainable artificial intelligence ensemble transfer learning for zero-day botnet attack detection in the internet of vehicles. *Results Eng.* **24**, 103171 (2024).
25. Saheed, Y. K., Misra, S., & Chockalingam, S. Autoencoder via DCNN and LSTM models for intrusion detection in industrial control systems of critical infrastructures. In *2023 IEEE/ACM 4th International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS)* 9–16 (IEEE, 2023)
26. Vakili, A. et al. A new service composition method in the cloud-based internet of things environment using a grey wolf optimization algorithm and MapReduce framework. *Concurr. Comput. Pract. Exp.* **36**(16), 8091 (2024).
27. Heidari, A., Jafari Navimipour, N., Dag, H. & Unal, M. Deepfake detection using deep learning methods: A systematic and comprehensive review. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **14**(2), 1520 (2024).
28. Heidari, A., Navimipour, N. J., Dag, H., Talebi, S. & Unal, M. A novel blockchain-based deepfake detection method using federated and deep learning models. *Cogn. Comput.* **16**(3), 1073–1091 (2024).
29. Amiri, Z., Heidari, A., Navimipour, N. J., Esmaeilpour, M. & Yazdani, Y. The deep learning applications in IoT-based bio-and medical informatics: A systematic literature review. *Neural Comput. Appl.* **36**(11), 5757–5797 (2024).
30. Heidari, A., Shishehlou, H., Darbandi, M., Navimipour, N. J. & Yalcin, S. A reliable method for data aggregation on the industrial internet of things using a hybrid optimization algorithm and density correlation degree. *Clust. Comput.* **27**(6), 7521–7539 (2024).
31. Heidari, A., Navimipour, N. J. & Unal, M. A secure intrusion detection platform using blockchain and radial basis function neural networks for internet of drones. *IEEE Internet Things J.* **10**(10), 8445–8454 (2023).
32. Heidari, A., Amiri, Z., Jamali, M. A. J. & Jafari, N. Assessment of reliability and availability of wireless sensor networks in industrial applications by considering permanent faults. *Concurr. Comput. Pract. Exp.* **36**(27), 8252 (2024).
33. Amiri, Z., Heidari, A., Zavvar, M., Navimipour, N. J. & Esmaeilpour, M. The applications of nature-inspired algorithms in Internet of Things-based healthcare service: A systematic literature review. *Trans. Emerg. Telecommun. Technol.* **35**(6), 4969 (2024).
34. Zanbouri, K. et al. A GSO-based multi-objective technique for performance optimization of blockchain-based industrial Internet of Things. *Int. J. Commun. Syst.* **37**(15), 5886 (2024).
35. Heidari, A., Navimipour, N. J., Zeadally, S. & Chamola, V. Everything you wanted to know about ChatGPT: Components, capabilities, applications, and opportunities. *Internet Technol. Lett.* **7**(6), 530 (2024).
36. Amiri, Z., Heidari, A., & Navimipour, N. J. Comprehensive survey of artificial intelligence techniques and strategies for climate change mitigation. Energy 132827 (2024)
37. Asadi, M., Jamali, M. A. J., Heidari, A. & Navimipour, N. J. Botnets unveiled: A comprehensive survey on evolving threats and defense strategies. *Trans. Emerg. Telecommun. Technol.* **35**(11), 5056 (2024).
38. Heidari, A., Jamali, M. A. J. & Navimipour, N. J. Fuzzy logic multicriteria decision-making for broadcast storm resolution in vehicular Ad Hoc networks. *Int. J. Commun. Syst.* **38**(5), 6034 (2025).

39. Heidari, A., Amiri, Z., Jamali, M. A. J. & Navimipour, N. J. Enhancing solar convection analysis with multi-core processors and GPUs. *Eng. Rep.* 7(1), 13050 (2025).
40. Toumaj, S., Heidari, A., Shahhosseini, R. & Jafari Navimipour, N. Applications of deep learning in Alzheimerâ€™s disease: A systematic literature review of current trends, methodologies, challenges, innovations, and future directions. *Artif. Intell. Rev.* **58**(2), 44 (2024).
41. Heidari, A., Jafari Navimipour, N. & Unal, M. The history of computing in Iran (Persia)â€”since the achaemenid empire. *Technologies* **10**(4), 94 (2022).
42. Jabraeil Jamali, M. A., Bahrami, B., Heidari, A., Allahverdizadeh, P., Norouzi, F., Jabraeil Jamali, M. A., Bahrami, B., Heidari, A., Allahverdizadeh, P. & Norouzi, F. The IoT landscape. In *Towards the Internet of Things: Architectures, Security, and Applications* 1–8 (2020)
43. Yao, W., Hu, L., Hou, Y. & Li, X. A two-layer soft-voting ensemble learning model for network intrusion detection. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)* 155–161 (IEEE, 2022).
44. Das, S. et al. Network intrusion detection and comparative analysis using ensemble machine learning and feature selection. *IEEE Trans. Netw. Serv. Manag.* **19**(4), 4821–4833 (2021).
45. Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. KNN model-based approach in classification. In *On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3–7, 2003. Proceedings* 986–996 (Springer, 2003).
46. Faramarzi, A., Heidarinejad, M., Stephens, B. & Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl. Based Syst.* **191**, 105190 (2020).
47. Sáez, J. A., Luengo, J., Stefanowski, J. & Herrera, F. SMOTE-IPF: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Inf. Sci.* **291**, 184–203 (2015).
48. Sahu, S., & Mehtre, B. M. Network intrusion detection system using J48 decision tree. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* 2023–2026 (IEEE, 2015).
49. Ao, H. Using machine learning models to detect different intrusion on NSL-KDD. In *2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE)* 166–177 (IEEE, 2021).
50. Al-Daweri, M. S., Zainol-Ariffin, K. A., Abdullah, S. & Md. Senan, M. F. E. An analysis of the KDD99 and UNSW-NB15 datasets for the intrusion detection system. *Symmetry* **12**(10), 1666 (2020).
51. Choudhary, S. & Kesswani, N. Analysis of KDD-Cupâ€™99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT. *Procedia Comput. Sci.* **167**, 1561–1573 (2020).
52. Janarthanan, T., & Zargari, S. Feature selection in UNSW-NB15 and KDDCUP'99 datasets. In *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)* 1881–1886 (IEEE, 2017).
53. Masoodi, F. et al. Machine learning for classification analysis of intrusion detection on NSL-KDD dataset. *Turk. J. Comput. Math. Educ. (TURCOMAT)* **12**(10), 2286–2293 (2021).
54. Abbas, Q., Hina, S., Sajjad, H., Zaidi, K. S. & Akbar, R. Optimization of predictive performance of intrusion detection system using hybrid ensemble model for secure systems. *PeerJ Comput. Sci.* **9**, 1552 (2023).
55. Pokharel, P., Pokhrel, R., & Sigdel, S. Intrusion detection system based on hybrid classifier and user profile enhancement techniques. In *2020 International Workshop on Big Data and Information Security (IWBIS)* 137–144 (IEEE, 2020).
56. Bong, K., & Kim, J. Analysis of intrusion detection performance by smoothing factor of Gaussian NB model using modified NSL-KDD dataset. In *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)* 1471–1476 (IEEE, 2022).
57. Meliboev, A., Alikhanov, J. & Kim, W. Performance evaluation of deep learning based network intrusion detection system across multiple balanced and imbalanced datasets. *Electronics* **11**(4), 515 (2022).
58. Dora, V. R. S. & Lakshmi, V. N. Optimal feature selection with CNN-feature learning for DDoS attack detection using meta-heuristic-based LSTM. *Int. J. Intell. Robot. Appl.* **6**(2), 323–349 (2022).
59. Ahmad, M. et al. Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set. *EURASIP J. Wirel. Commun. Netw.* **2021**, 1–23 (2021).

Author contributions
Neha Pramanick: Formal analysis, Writing—original draft, Data curation, Data Processing and Analysis, Original draft preparation, and Writing. Jimson Mathew: Reviewing, and Revising. Shitharth Selvarajan: Reviewing, and Revising. Mayank Agarwal: Research profile design, Writing, Reviewing, and Revising.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to S.S.

Reprints and permissions information is available at www.nature.com/reprints.