Düzce University
Journal of Science & Technology

Research Article

Nelder-Mead Optimized Weighted Voting Ensemble Learning for Network Intrusion Detection

Mustafa Burak ÜRÜN [a,*], Yusuf SÖNMEZ [b]

[a] *Graduate School of Natural and Applied Sciences, Gazi University, Ankara, TURKEY*
[b] *Department of Computer Engineering, Faculty of Technology, Gazi University, Ankara, TURKEY*
***** *Corresponding author's e-mail address: mustafaburak.urun@gazi.edu.tr*
DOI: 10.29130/dubited.1440640

ABSTRACT

The rise in internet usage and data transfer rates has led to numerous anomalies. Hence, anomaly-based intrusion detection systems (IDS) are essential in cybersecurity because of their ability to identify unknown cyber-attacks, especially zero-day attacks that signature-based IDS cannot detect. This study proposes an ensemble classification for intrusion detection using a weighted soft voting system with KNN, XGBoost, and Random Forest base models. The base model weights are optimized using the Nelder-Mead simplex method to improve the overall ensemble performance. In this study, a robust intrusion detection framework that uses soft-voting classifier-level weights optimized using the Nelder-Mead algorithm and feature selection is proposed. The system's performance was evaluated using the KDD99 and UNSW-NB15 datasets, which demonstrated that the proposed approach exceeded other existing methods in respect of accuracy and provided comparable results with fewer features. The proposed system and its hyperparameter optimization technique were compared with other cyber threat detection and mitigation systems to determine their relative effectiveness and efficiency.

Keywords: *Intrusion Detection Systems, Ensemble Learning, Soft Voting, Hyperparameter Optimization, Nelder-Mead Algorithm.*

Saldırı Tespiti İçin Nelder-Mead Algoritması ile Optimize Ağırlıklı Oylama Topluluk Öğrenmesi

ÖZET

İnternet kullanımı ve veri aktarım hızlarındaki artış çok sayıda anomaliye yol açmıştır. Bu nedenle, anomali tabanlı saldırı tespit sistemleri (IDS), bilinmeyen siber saldırıları, özellikle de imza tabanlı IDS'lerin tespit edemediği sıfırıncı gün saldırılarını belirleme yetenekleri nedeniyle siber güvenlikte çok önemlidir. Bu çalışmada, KNN, XGBoost ve Rastgele Orman temel modelleri ile ağırlıklı bir yumuşak oylama sistemi kullanarak saldırı tespiti için bir topluluk sınıflandırması önerilmektedir. Temel modellerin ağırlıkları, genel topluluk performansını iyileştirmek için Nelder-Mead simpleks yöntemi kullanılarak optimize edilmiştir. Bu çalışmada, Nelder-Mead algoritması ve özellik seçimi kullanılarak optimize edilen yumuşak oylama sınıflandırıcı seviyesi ağırlıklarını kullanan sağlam bir saldırı tespit çerçevesi önerilmektedir. Sistemin performansı KDD99 ve UNSW-NB15 veri setleri kullanılarak değerlendirilmiş ve önerilen yaklaşımın doğruluk açısından mevcut diğer yöntemleri aştığını ve daha az özellik ile karşılaştırılabilir sonuçlar sağladığı tespit edilmiştir. Önerilen sistem ve hiperparametre optimizasyon tekniği, göreceli etkinlik ve verimliliğini belirlemek için diğer siber tehdit tespit ve sınırlama sistemleriyle karşılaştırılmıştır.

Received: 25/02/2024, Revised: 01/07/2024, Accepted: 10/07/2024

Anahtar Kelimeler: *Saldırı Tespit Sistemleri, Topluluk Öğrenme, Yumuşak Oylama, Hiperparametre Optimizasyonu, Nelder-Mead Algoritması*

I. INTRODUCTION

The widespread use of information technology and the expansion of cyberspace have made significant contributions to economic and societal progress. However, these advancements have also brought new security risks and challenges. Ensuring the security of cyberspace is critical to the mutual interests of humanity, global peace and development, and the national security of all nations. With the advent of the digital age, interconnectedness has become the norm, and the world now heavily depends on software. As a result, the presence of software inevitably leads to the existence of vulnerabilities, which in turn results in increasingly complex network infrastructure and a constantly expanding attack surface. Network security attacks are becoming more sophisticated and widespread on scale. Ultimately, it can be claimed that all financial and technological conflicts are essentially being carried out in cyberspace.

According to SonicWall threat researchers, there has been a notable increase in overall intrusion attempts globally during the first half of 2023. Specifically, they have recorded a staggering 3.7 trillion intrusion attempts, which represents a 21% rise compared to the same period in 2022 [1]. Cybersecurity breaches continue to increase, heightening uncertainty and risks and causing widespread concern globally due to their impact on national security and the economy. The cost of preventing cybercrime is rapidly mounting, generating financial losses including organizational damages, compensation and legal fees [2]. This trend is creating a global impact given its consequences to national security and the economy.

The rise in internet usage and data transfer rates has led to numerous anomalies. Institutions and organizations are continually upping their investment in cybersecurity technologies to ensure a secure and stable service for their users. The market value for cybersecurity was approximately $164 billion in 2019, and it is projected to reach $637 billion by 2030. Figure 1 [3] provides a depiction of the estimated size of the cybersecurity market over time. For these reasons, it is clear that there should be a greater focus on technology optimization for effective cyber defense.

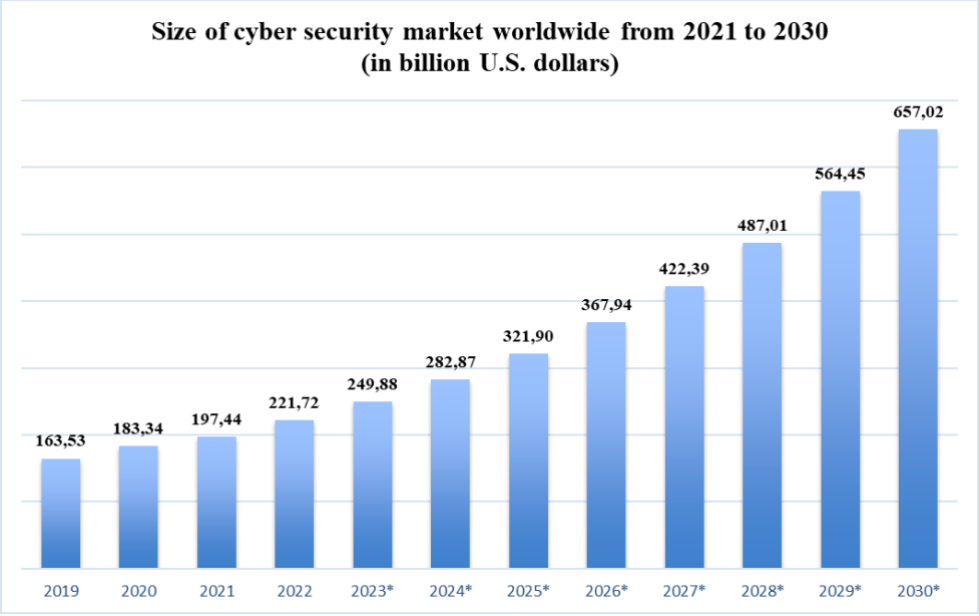

Figure 1. *Size of cyber security market worldwide from 2021 to 2030*

The aim of using machine learning techniques is to develop an intelligent intrusion detection system (IDS) as a secondary defense mechanism for securing computer networks. IDS works by keeping track of activity on a computer network or system. The system analyzes events to identify potential security breaches or unauthorized access attempts. Anomaly-based intrusion detection systems play a key role in the field of cybersecurity due to their ability to identify unknown cyberattacks, specifically zero-day attacks, which cannot be detected by signature-based IDSs. These systems employ the analysis of network traffic and/or resource usage to identify abnormal states, classifying them as anomalies. When malicious activity is detected, an alert is raised, enabling early intrusion and preventing potential attacks from causing substantial damage. Anomaly-based IDSs are also effective in detecting suspicious activity or policy violations in real-time, providing an opportunity for timely response and mitigation to mitigate potential harm caused by cyber threats [4]. In today's Information and Communication Technology (ICT) era, IDSs are crucial to strive for comprehensive protection to ensure the confidentiality, integrity, and availability of information.

Main contributions of this paper are as follows:

- In the domain of intrusion detection, an ensemble classification approach is proposed, which leverages a weighted soft voting mechanism in conjunction with KNN, XGBoost, and Random Forest as base models. To optimize the performance of the soft voting, the weights associated with each base model are accurately determined through the Nelder-Mead algorithm.
- To evaluate the performance of the proposed system, two datasets, namely KDD99 and UNSW-NB15, were used. These datasets served as the foundation for assessing the effectiveness and efficiency of the suggested system in achieving its objectives.
- The performance of the proposed system has been compared with other systems in order to determine its relative efficiency and effectiveness in detecting and mitigating cyber threats.

II. RELATED WORK

In the domain of intrusion detection, a diverse set of machine learning algorithms is employed to detect and classify patterns and anomalies in network traffic and system behavior. These algorithms play a crucial role in identifying and preventing cyber-attacks. By using machine learning algorithms, intrusion detection systems can adapt and learn from new data, enhancing their capacity to detect emerging threats and new attack patterns.

Ensemble methods, on the other hand, are machine learning techniques that involve combining multiple base models to improve the overall performance of a prediction or classification system. Ensemble methods have gained significant attention in machine learning research due to their effectiveness in reducing false positive rates and generating more accurate solutions compared to using a single model alone. By aggregating the predictions of multiple models, ensemble methods aim to leverage the diversity and complementary strengths of individual models, leading to improved overall performance.

Yao et al. [5] conducted experiments using the UNSW-NB15 dataset to evaluate the performance of Soft Voting with three baseline models: XGBoost, LightBGM, and Random Forest. The results demonstrate that the Soft Voting method achieves higher accuracy rates compared to the individual models, both in binary and multiclass classification scenarios. This finding suggests that using Soft Voting is both feasible and efficient for improving the accuracy of classification models.

Shen et al. [6] employed Class Level Soft Voting as an ensemble learning technique, using Support Vector Machines (SVM), k-Nearest Neighbors (KNN), and Decision Trees (DT) as base models. In the Soft Voting approach, the weights of the base models were optimized using the Chaos Bat algorithm, which draws inspiration from the behavior of bats and incorporates concepts from chaos theory. This

optimization process aims to find the ideal combination of weights for the base models, thereby enhancing the overall performance of the ensemble. The experimental results showcased that the Soft Voting ensemble method achieved accuracy levels comparable to other ensemble learning algorithms.

Swami et al. [7] introduced a voting-based ensemble model for predicting final results in the detection of Distributed Denial of Service (DDoS) attacks against Software Defined Networks (SDN). The paper specifically focuses on analyzing three ensemble models: Voting-CMN (CART, MLP, NB), Voting-RKM (RF, KNN, MLP) and Voting-CKM (CART, KNN, MLP). The experimental results indicated that the proposed voting models were more successful in terms of accuracy when compared to the existing models.

Zhou et al. [8] developed a model for detecting network attacks by using feature selection and voting ensemble learning techniques. The model incorporates Correlation Based Feature Selection, which is a heuristic algorithm used for reducing dimensionality by selecting the optimal subset of features based on their correlation. The proposed approach adopts a voting ensemble learning method that includes Forest PA (Forest by Penalizing Attributes), Random Forest, and C4.5 algorithms. The voting mechanism combines the probability estimates from the base learners to make the ultimate prediction. The experimental results demonstrated that the proposed model outperforms other existing methods in terms of various evaluation metrics.

Gu et al. [9] presented a framework for intrusion detection that is based on an ensemble of Support Vector Machine (SVM) models. To address the issue of imbalanced datasets commonly encountered in intrusion detection, the authors applied the ratio transformations technique, which aims to improve the distribution of data by adjusting the ratio between the minority and majority classes. By enhancing the quality of the training data, this technique helps improve the effectiveness of the intrusion detection system. To further enhance the diversity of the SVM classifiers, the authors employed the Fuzzy C-Means (FCM) method. This method contributes to capturing diverse knowledge within the ensemble by introducing additional variability in the classifiers. Finally, the authors applied a non-linear combination method to aggregate the individual SVM classifiers. This aggregation technique enables the ensemble model to benefit from the diverse knowledge captured by each SVM classifier, resulting in improved intrusion detection performance.

Gao et al. [10] proposed an adaptive ensemble method for intrusion detection. The approach involved combining base classifiers, namely Decision Trees (DT), Random Forest (RF), k-Nearest Neighbors (KNN), and deep neural networks, using the voting method with class weights. To determine the class weights, the authors calculated the training accuracy of each algorithm for several attack types. These weights were then used in the voting process to assign more importance to the predictions of the base classifiers with higher accuracy. By employing this adaptive ensemble method, Gao et al. aimed to enhance the general performance of intrusion detection systems by leveraging strengths of multiple base classifiers and adjusting their contributions based on their individual accuracies for different attack types.

Seth et al. [11] put forward a model that aims to construct an ensemble for identifying multiple attack types. The approach involves evaluating the detection abilities of various base classifiers and ranking them accordingly. The ranking matrix is computed based on the F1-score of each algorithm for various attack types. When making the final prediction, the output of an algorithm for a particular attack is only taken into account if it has the highest F1-score in the ranking matrix for that specific attack category.

Zhang et al. [12] introduced a dynamic weighted voting classifier (DWC) as a method for network intrusion detection. The DWC classifier offers improved performance compared to fixed weighted voting and simple majority rule voting. To implement the DWC classifier, an ensemble of classifiers is first trained using a dataset. During the prediction phase, the DWC classifier calculates the weighted sum of the individual classifiers' predictions. The weights assigned to each classifier are updated dynamically based on their performance in previous predictions. This dynamic updating ensures that classifiers with better performance receive more weight in the final prediction. The results showed that the DWC classifier surpassed both fixed weighted voting and simple majority rule voting in terms of evaluation metrics.

Previous studies in the field have primarily focused on optimizing machine learning algorithms to improve the overall effectiveness of intrusion detection systems. Feature selection and ensemble learning have been the main methods used for optimization. Despite these efforts, there is still substantial potential for improvement in the results of these studies. More study and investigation are required to improve the ability to spot security breaches effectively.

III. MATERIALS AND METHODS

A. DATASETS

KDD99 and UNSW-NB15 datasets were used in this research.

The KDD99 dataset has gained significant popularity in the domains of intrusion detection and machine learning. It played a vital role in the Third International Knowledge Discovery and Data Mining Tools Competition (KDD-99), which assessed different data mining and knowledge discovery techniques. The main objective of the competition was to develop a network intrusion detection system that could accurately classify between malicious connections, known as attacks, and normal connections. The dataset comprises network traffic data, including normal and attack segments, to simulate a standard network. These attacks are classified into four categories: DOS (Denial of Service), R2L (Remote-to-Local), U2R (User-to-Root), and probing. Each record in the dataset is assigned a class label and is associated with 41 fixed feature attributes [13]. Within the 41 fixed feature attributes of the KDD99 dataset, nine attributes are of the discrete type, while the rest are continuous. This study focuses on the "kddcup.data_10_percent" dataset, with a total of 494021 records.

The UNSW-NB15 dataset was created by the University of New South Wales (UNSW) to provide a comprehensive collection of raw network packets. This dataset combines genuine modern normal activities with synthetic contemporary attack behaviors. The data was captured using the tcpdump tool, resulting in a total of 100 GB of raw traffic stored in Pcap files. The UNSW-NB15 dataset includes nine distinct attack types, namely Fuzzers, Analysis, Backdoors, DoS (Denial of Service), Exploits, Generic, Reconnaissance, Shellcode, and Worms. To generate the UNSW-NB15 dataset, the researchers utilized the IXIA PerfectStorm tool. This process resulted in the creation of a dataset with a total of 49 features, including the class label [14]. The dataset is split into two subsets: the training set (UNSW_NB15_training-set.csv) and the testing set (UNSW_NB15_testing-set.csv). The training set consists of 175341 records, while the testing set contains 82332 records. In total, the UNSW-NB15 dataset comprises 257673 records.

B. THE BASE LEARNERS

B. 1. K-Nearest Neighbor (KNN)

The K-Nearest Neighbor (KNN) algorithm is a non-parametric, supervised learning classifier that applies the proximity of data points to make classifications or predictions. It operates by examining the class labels of the nearest neighbors of a given data point to designate a class label to that point. The algorithm begins by calculating the distance between the query point and all other data points in the dataset. The most commonly used distance metric is Euclidean distance, which measures the straight-line distance between two points. Other distance metrics, such as Manhattan and Minkowski distances, can also be used depending on the data and problem domain [15].

Once the distances are calculated, the algorithm selects the k nearest neighbors based on the chosen distance metric. The parameter k represents the number of neighbors to count when making a classification. A higher value of k considers more neighbors, while a lower value of k focuses on fewer neighbors [16]. Finally, the algorithm assigns a class label to the query point based on the majority class

of its k-nearest neighbors. Fundamentally, the query point is assigned the class label that is most commonly found among its k neighbors.

The KNN algorithm is simple and often used for sorting and prediction. It gives a new data point a category from the most common class among its closest neighbors. While it has advantages such as its simplicity and ability to learn non-linear decision boundaries, it can be computationally intensive for large training sets and choosing the right value for k can be a balancing act.

B. 2. Xgboost

The XGBoost algorithm is a technique that combines weak learners, usually decision trees, in a sequential manner to create a robust predictive model. It follows the boosting principle, where each new model in the sequence focuses on correcting the errors made by the previous models, thereby improving the overall predictive performance.

One of the advantages of XGBoost is its capacity to execute faster through parallel processing. This allows for quicker model training and prediction. Additionally, XGBoost offers portability, meaning it can be easily implemented on different platforms and systems.

XGBoost, a popular gradient boosting algorithm, employs decision trees as its base learners. Decision trees are relatively straightforward models that partition the input space into distinct regions and assign a specific value to each region. This partitioning allows decision trees to capture complex relationships and make predictions based on the selected region. Multiple decision trees are sequentially built and combined to improve the overall predictive performance. In XGBoost, these trees are often shallow, meaning they have a limited depth. This helps prevent overfitting and makes the algorithm more efficient [17].

The XGBoost algorithm operates within the gradient boosting frame, which aims to optimize a specified loss function. It achieves this by reiteratively appending new models that correct the errors made by existing models. The accomplishment of this task is done by evaluating the slope of the loss function in relation to the projections made by the model and using this information to update the model in a way that minimizes the loss. To control the complexity of the model and prevent overfitting, XGBoost incorporates regularization techniques. Regularization terms are added to the objective function, penalizing complex models and encouraging simpler, more generalizable solutions.

XGBoost is built for quick and effective performance, utilizing simultaneous tasks and networked computers to manage big data and speed up the training of models. Additionally, XGBoost employs tree pruning, which eliminates unnecessary splits that do not contribute significantly to the model's predictive power. This results in more compact and efficient trees, reducing the risk of overfitting.

The XGBoost algorithm follows a series of steps to build its predictive model:

1. Initial Prediction and Residual Calculation: The algorithm begins by making an initial prediction based on the available features. To evaluate the performance of a predictive model, the difference between the predicted value and the observed value is calculated, which gives rise to residuals.
2. Building XGBoost Trees: Each tree in the XGBoost model starts with a single leaf, and all the residuals are assigned to that leaf. The algorithm then calculates a similarity score for this leaf, which measures how well the residuals are explained by the leaf's predicted value.
3. Pruning the Tree: The tree is pruned to optimize its structure and prevent overfitting. This involves evaluating the impact of removing each leaf and its corresponding branches and removing those that do not contribute significantly to improving the model's performance.
4. Output Calculation: The output value for each leaf is calculated based on the residuals assigned to that leaf. This value represents the contribution of this specific leaf to the final prediction.

5. Final Prediction: To make predictions using the trained XGBoost model, the output from each tree is multiplied by a learning rate (a hyperparameter that controls the contribution of each tree) and added to the initial prediction. This aggregation process merges the predictions from each tree in the model to arrive at a final value or classification.

By following these steps, the XGBoost algorithm iteratively improves the model's accuracy by sequentially adding trees that focus on reducing the residuals from previous iterations.

B. 3. Random Forest

The Random Forest algorithm is an ensemble learning method that leverages multiple decision trees to make predictions. During the training phase, the algorithm constructs many decision trees. Each decision tree is trained on a different subset of the data, using random feature subsets.

When making predictions, the Random Forest algorithm combines the predictions of all the individual decision trees. For classification tasks, the mode of the predicted classes across the trees is taken as the final prediction. In regression tasks, the average prediction across the trees is used. This aggregation of predictions from multiple decision trees helps to improve the accuracy and robustness of the Random Forest algorithm [18].

Random Forest leverages the collective wisdom of these individual decision trees to enhance overall accuracy. Random Forest utilizes two primary methods, bagging and feature randomness, to infuse variety into the group.

The process of bagging entails instructing individual decision trees using distinct, randomly selected subsets of the training data. This approach enables multiple trees to acquire knowledge from diverse data subsets, thereby mitigating the risk of overfitting and enhancing the overall generalization capacity of the model. Incorporating feature randomness ensures that each tree considers only a randomly selected subset of features when deciding how to split. By introducing this randomness, Random Forest prevents any single feature from dominating the decision-making process and promotes the exploration of different feature combinations [19].

Random Forest offers several advantages. It tends to achieve high accuracy due to the collective decision-making of multiple trees. It is also robust to noise in the data, as the majority vote or average prediction helps mitigate the impact of outliers. The use of Random Forest can offer valuable insights regarding the importance of features, enabling analysts to comprehend which features have the greatest impact on the predictions. Furthermore, Random Forest is effective in handling high-dimensional data with numerous features.

However, there are some considerations when using Random Forest. Constructing multiple decision trees means that training a Random Forest model can require extensive computing, especially for large amounts of data. The complex inner workings of Random Forest can also make it challenging to interpret compared to simpler models. Additionally, tuning the hyperparameters of Random Forest requires careful experimentation to achieve optimal performance.

C. VOTING METHOD

A voting classifier represents a form of ensemble learning method where multiple machine learning models are combined to make a final prediction. In this technique, each model is trained independently on a subset of the dataset, and their respective predictions are then aggregated to derive a final decision. Two categories of voting classifiers are distinguished: hard voting and soft voting.

The *hard voting* combination rule for class labels means picking what most classifiers agree on the most, which is a mere count of votes. [20]. This rule is commonly used in bagging. In binary settings, this

approach is like the typical Condorcet method—whichever class wins head-to-head against all others becomes the chosen one.

Soft voting combination rules in machine learning refer to a method of combining the predictions of multiple base classifiers. Each base classifier provides a continuous output, which represents the probability distribution over the possible outcomes for a given input. The soft voting combination rule involves polling the continuous outputs generated by each base classifier through the application of functions such as average, maximum, or minimum.

To determine the final prediction, the function is applied to the predicted probabilities, and the class label chosen being the one that maximizes the function's value [21]. Among the different functions, the average is considered as the most potent in terms of predictive efficacy. When using soft voting, the continuous outputs of the base classifiers are first converted into probability distributions. These probability distributions are then averaged, and the definitive prediction is made on the basis of averaged probabilities. This approach allows for a more robust prediction by considering the collective knowledge of multiple models. It can help improve the accuracy and reliability of predictions in machine learning tasks.

Weighted voting is a voting technique that allows for the adjustment of the weights assigned to different base models. Unlike majority voting, where all models have equal weight, weighted voting assigns varying levels of importance to different models [22]. As a result, the predictions of models with higher weights are considered multiple times and have a more significant impact on the final decision. This approach takes into account the unique strengths and weaknesses of each model, providing a more nuanced approach to decision-making. By assigning variable weights, the most reliable and accurate predictions can have a greater influence on the final outcome. This is particularly advantageous when certain models have demonstrated superior accuracy or expertise in specific domains. Weighted voting offers a flexible and adaptable framework for prediction, allowing for a more personalized and efficient approach.

D. NELDER-MEAD ALGORITHM

The Nelder-Mead algorithm, also referred to as the downhill simplex method, is a widely used optimization algorithm employed to locate the minimum of a function in a multidimensional space. The algorithm was developed by John Nelder and Roger Mead in 1965 and is particularly advantageous for scenarios where the objective function lacks differentiability or gradient information is unavailable.

This algorithm functions by maintaining a simplex, which is a geometric shape that extends the concept of a triangle to higher dimensions. It then iteratively adjusts the vertices of the simplex to gradually converge towards the minimum of the objective function. During each reiteration, the algorithm evaluates the objective function at the vertices of the simplex and subsequently applies operations like reflection, expansion, contraction, or shrinkage to update simplex [23].

The Nelder-Mead algorithm is relatively straightforward to implement and does not necessitate the computation of derivatives, making it suitable for a wide array of optimization problems. Nevertheless, its efficiency may not be on par with certain other optimization algorithms, particularly in high-dimensional spaces or for functions with intricate geometries.

Figure 2 illustrates the methodology of the original Nelder-Mead algorithm and its effectiveness in achieving the optimal solution. The algorithm utilizes four parameters, -specifically, reflection, expansion, contraction, and simplex size- to manipulate the simplex within the design space. This manipulation is dependent on the values present at both the vertices and the center of the triangle [24].

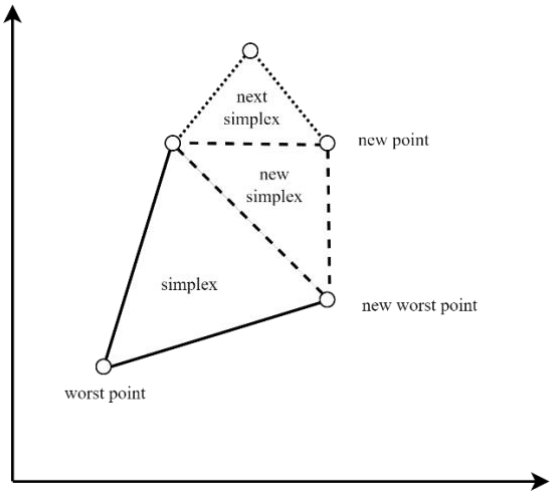

Figure 2. *Nelder-Mead Simplex Method [24]*

As depicted in Figure 2, the shape is a closed N-dimensional object in space, with line edges intersecting at N + 1 vertices. The object's motion consists of a reflection, creating a new vertex and simplex. The direction of the reflection and the selection of the new vertex are contingent on the worst point's placement in the simplex. The new vertex is referred to as the complement of the worst vertex. If a newly added vertex is the worst point of the new simplex, the algorithm oscillates between it and the previous worst point. To find the next new point, the second worst point is used in this case. The simplex moves through the design space and its center moves towards the endpoint. The edges of the simplex are allowed to expand and contract, moving towards the optimum [25].

E. EVALUATION METRICS

The four essential attributes of the confusion matrix, which clearly illustrates the actual and predicted classes as shown in Table 1, form the basis of the evaluation metrics we typically use.

Table 1. *Confusion Matrix*

		Predicted Class	
		Attack (0)	Normal (1)
Actual Class	Attack (0)	TN	FP
	Normal (1)	FN	TP

In this context, we establish the confusion matrix as represented in Table 1, where the abbreviations TN and TP correspond to true negatives and true positives, respectively. These terms denote the correct classification of attack and normal samples. Conversely, a false negative (FN) denotes a misclassification of a normal sample as an attack, while a false positive (FP) indicates the erroneous classification of an attack sample as normal [26].

In this paper, we evaluate various methods using three essential metrics: Accuracy, F1 Score, and Matthews Correlation Coefficient (MCC) [27]. We will now delve into the exact meanings of these metrics.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (4)$$

$$MCC = \frac{(TP \times TN)-(FP \times FN)}{\sqrt{(TP+FP) \times (TP+FN) \times (TN+FP) \times (TN+FN)}} \quad (5)$$

F. PROPOSED METHODOLOGY

The study involved three phases: first, we conducted data preprocessing to prepare the dataset for binary classification. The dataset was divided into two sets: the test set and the training set. The test set was created to evaluate the performance of the models and measure their generalization ability. It consisted of 20% of the total dataset. On the other hand, the training set was composed of the remaining 80% of the data and served as the input for model training and parameter optimization. In the second phase, we utilized base classifiers for classification purposes. Based on their accuracy rates, we selected the top three algorithms from the outcomes. Finally, in the third stage, we analyzed accuracy performance by combining results from base classifiers using soft voting. Moreover, we optimized hyperparameters at the classifier-level to further improve accuracy performance.

The *first phase* involved data cleaning, feature extraction and feature selection.

Data cleaning is the pivotal procedure of detecting and rectifying, or eliminating, discrepancies, disparities, and inaccuracies within datasets. This indispensable phase in data scrutiny holds paramount significance in guaranteeing the precision and dependability of the outcomes derived from the data. Data cleaning contains a range of activities, including identifying missing or duplicate data, correcting spelling and formatting errors, and removing outliers and irrelevant data [28]. The goal of data cleaning is to ensure that the data is consistent, complete, and accurate, and that it is suitable for analysis. In this study, we removed repetitive rows from the KDD99 dataset which reduced the number of rows to 145586. We also removed rows with an empty "service" column from the UNSW-NB15 dataset which resulted in a dataset with 116352 rows.

Feature extraction is critical to analyzing data, as it selects and transforms relevant information from raw data. This procedure generates a set of features that can be used effectively by machine learning algorithms. Its main objective is to identify and extract the most significant and related features from a dataset, which, in turn, eases the training of machine learning models. One of the primary advantages of feature extraction lies in its capacity to decrease data dimensionality. Through the selection of highly informative features, it simplifies the analysis and processing procedures. Furthermore, this approach aids in the elimination of noise and irrelevant information from the dataset, resulting in enhanced accuracy and performance of machine learning models by prioritizing the most crucial aspects of the data [29].

In this study, datasets are prepared for training and testing using Principal Component Analysis (PCA) to reduce dimensionality and one-hot encoding for categorical features. First, features of either floating-point or integer data types are selected using PCA, after which a minimum variance threshold is established to determine the number of principal components to retain. The chosen features are then subjected to standard scaling and PCA transformation. Subsequently, we identify categorical features and use CountVectorizer to one-hot encode them. We then combine the transformed numerical and encoded categorical features, converting them into a data frame.

Feature selection is the identification and selection of the most significant features from a given dataset that can be employed to establish a strong predictive model [30]. It is crucial to select only the pertinent features to avoid complexity and boost the model's accuracy. In the context of feature selection, the XGBoost model calculated feature importance to determine the usefulness of each feature in constructing the boosted decision trees. To optimize the model's effectiveness, features with a score below the threshold of 0.01 are eliminated from the dataset. By strategically discarding these less influential features, the model's performance is enhanced, ultimately leading to more accurate and reliable predictions. Following the process of feature selection, a total of 10 features have been selected in the KDD99 dataset, while the UNSW-NB15 dataset has undergone feature selection resulting in the selection of 11 features. The target label (normal and attack) is added to the dataset before data splitting. Then, the process of data splitting has then been initiated.

Stratified sampling is employed to uphold a balanced representation of each class in the target variable within both the training and test sets. This method serves to mitigate any potential bias towards the predominant class, hence fostering a more equitable model. In this particular scenario, 80% of the data is allocated for training purposes, while the remaining 20% is reserved for testing.

During the *second phase* of the study, binary classification was carried out by implementing several base classifiers, including Random Forest, K-Nearest Neighbor, and XGBoost. The hyperparameters utilized for these base classifiers are meticulously outlined in Table 2 for reference and transparency.

Table 2. *Base Learner Parameters*

Methods	Parameters
KNN	n_neighbours = 3
Random Forest	n_estimators = 10
XGBoost	max_depth=3, n_estimators=100, objective='binary: logistic', booster='gbtree', n_jobs=2, random_state=1

Moving on to the *third phase*, the outcomes of the base classifiers, namely Random Forest, K-Nearest Neighbor, XGBoost, were effectively combined using the soft voting technique. This approach allowed for the aggregation of the predictions from each classifier, ultimately yielding a consolidated and robust decision based on the collective wisdom of the individual classifiers.

In the context of soft voting, it is important to acknowledge that the base classifiers often exhibit varying levels of performance. This implies that assigning equal importance to each classifier during the aggregation of results may not yield the optimal outcome. Consequently, it becomes advantageous to assign weights to the classifiers based on their individual performance. However, it is crucial to accurately determine these weights, as they significantly influence the overall performance of the ensemble. To address this, the Nelder-Mead algorithm is proposed in this study to precisely adjust the weights of the base learners, thereby optimizing the performance of the ensemble.

Our objective is to settle the optimum weight combination for the ensemble classifier, maximizing its performance on a dataset. The proposed approach entails constructing a soft voting classifier that comprises pre-trained base classifiers and subsequently evaluating its performance on the dataset. Subsequently, the minimize function is implemented to improve the accuracy performance of the

ensemble classifier through minimizing the classification error. Only two weights are optimized due to their relative importance and the potential for overfitting with additional parameters. Optimization begins with initial weight values of 1.0, and the search space is constrained between 0.5 and 1.5. This process is iteratively refined through the Nelder-Mead algorithm, gradually converging towards the optimal weight configuration.

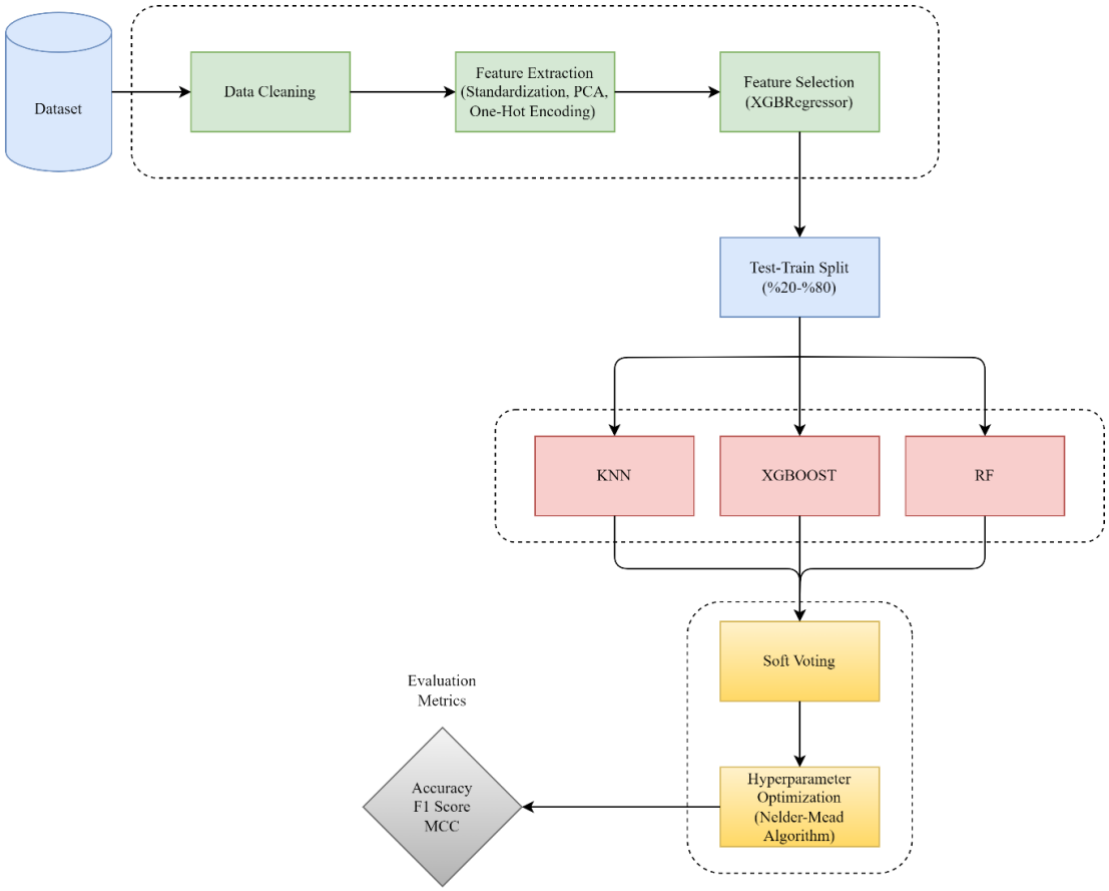

Figure 3. *Proposed Methodology*

IV. EXPERIMENTAL RESULTS

In this study, the experiments were conducted on a computer system consisting of an Intel® Core™ i7-8550U CPU running at a clock speed of 1.80 GHz. The system had 12 GB of RAM and was operating on the Windows 10 Enterprise Edition (64 bit) operating system.

To implement the proposed methodology, Python version 3.10.9 was used. The machine-learning classifiers employed in this research were implemented using the scikit-learn library version 1.2.1 [31] and the scipy library version 1.10.0 [32].

Tables 3-4 show the remarkable effectiveness of our KDD99 binary classification model, as evidenced by its exceptional overall results. These results are characterized by their high accuracy, Matthews Correlation Coefficient (MCC) and F1 scores. Within the training set, the Random Forest (RF) model reigns supreme, achieving near flawless results across all metrics. However, there is a slight drop in performance for all models when moving to the test set. Nevertheless, the Nelder-Mead variants show commendable resilience and adaptability to unseen data, maintaining virtually identical scores even in the face of this decline.

Table 3. KDD99 Training Results

Classifier	Accuracy	MCC	F1
RF	99.992%	99.984%	99.992%
Nelder-Mead 3 (KNN - RF)	99.977%	99.952%	99.977%
Soft Voting	99.976%	99.950%	99.976%
Nelder-Mead 1 (KNN - XGB)	99.976%	99.950%	99.976%
Nelder-Mead 2 (RF - XGB)	99.976%	99.950%	99.976%
XGBoost	99.973%	99.944%	99.973%
KNN	99.882%	99.752%	99.882%

Table 4. *KDD99 Test Results*

Classifier	Accuracy	MCC	F1
Nelder-Mead 1 (KNN - XGB)	99.856%	99.699%	99.856%
Nelder-Mead 2 (RF - XGB)	99.856%	99.699%	99.856%
Nelder-Mead 3 (KNN - RF)	99.852%	99.691%	99.852%
Soft Voting	99.845%	99.677%	99.845%
RF	99.828%	99.641%	99.828%
XGBoost	99.818%	99.620%	99.818%
KNN	99.811%	99.605%	99.811%

Tables 5-6 demonstrate the commendable performance of our UNSW-NB15 binary classification model, exhibiting a substantial degree of accuracy, Matthews Correlation Coefficient (MCC), and F1 scores. Notably, the Random Forest (RF) algorithm emerged as the most proficient classifier within the training set. Despite a decline in overall performance, the Nelder-Mead variants exhibited a remarkable ability to maintain their supremacy across all metrics during the testing phase. This observation suggests that this particular ensemble possesses a remarkable level of generalizability, effectively translating its success in training to real-world scenarios involving previously unseen data. The relative ranking of the models remained largely consistent between the training and testing stages, with RF and the other Nelder-Mead variants continuing to exhibit strong performance. These results indicate that the Nelder-Mead algorithm effectively navigates the optimization landscape, allowing us to determine the weight combination that maximizes the performance of the ensemble classifier.

Table 5. *UNSW-NB15 Training Results*

Classifier	Accuracy	MCC	F1
RF	99.826%	99.536%	99.826%
Nelder-Mead 3 (KNN - RF)	99.635%	99.025%	99.634%
Nelder-Mead 1 (KNN - XGB)	99.601%	98.936%	99.601%
Nelder-Mead 2 (RF - XGB)	99.601%	98.936%	99.601%
Soft Voting	99.560%	98.824%	99.559%
XGBoost	99.003%	97.335%	99.001%
KNN	98.612%	96.295%	98.611%

Table 6. *UNSW-NB15 Test Results*

Classifier	Accuracy	MCC	F1
Nelder-Mead 1 (KNN - XGB)	97.662%	93.733%	97.653%
Nelder-Mead 3 (KNN - RF)	97.649%	93.699%	97.640%
Nelder-Mead 2 (RF - XGB)	97.637%	93.664%	97.627%
Soft Voting	97.602%	93.571%	97.593%
RF	97.439%	93.116%	97.421%
XGBoost	97.426%	93.089%	97.412%
KNN	97.164%	92.426%	97.161%

In the KDD99 dataset, the training time was typically quickest for KNN and RF, followed by XGBoost and Soft Voting. It's worth mentioning that XGBoost and RF had a very short estimation time for the training set, while there was a more noticeable difference in the case of KNN and Soft Voting. We noticed similar patterns when it came to estimating the test set time, with XGBoost and RF once again showing the fastest performance, while KNN performed relatively slower.

Table 7. *KDD99 Timing Results*

Classifier	Training Duration (mm:ss.000)	Training Set Estimation Time (mm:ss.000)	Test Set Estimation Time (mm:ss.000)
Soft Voting	00:16.429352	00:02.843084	00:01.359055
XGBoost	00:13.115712	00:00.176238	00:00.062485
RF	00:02.624386	00:00.187456	00:00.062484
KNN	00:00.718582	00:09.750892	00:01.873693
Nelder-Mead 1 (KNN-XGB)	N/A	00:03.577284	00:01.249709
Nelder-Mead 2 (RF-XGB)	N/A	00:02.452550	00:01.062249
Nelder-Mead 3 (KNN-RF)	N/A	00:03.742623	00:00.651294

Table 8 presents the timing analysis outcomes for the UNSWNB15 dataset. According to the provided table, the training duration for all classifiers was relatively short. The training times varied, with KNN taking the shortest time of 0.789 seconds, while Soft Voting had the longest training duration of 11.327 seconds. This suggests that all the classifiers are efficient to train. The training set estimation time was also relatively short for all classifiers, ranging from 0.137 seconds for XGBoost to 6.264 seconds for KNN. Similarly, the test set estimation time was also relatively short for all classifiers, ranging from 0.031 seconds for RF to 2.012 seconds for KNN. This suggests that all the classifiers are able to quickly make predictions on training and test sets.

Table 8. *UNSW-NB15 Timing Results*

Classifier	Training Duration (mm:ss.000)	Training Set Estimation Time (mm:ss.000)	Test Set Estimation Time (mm:ss.000)
Soft Voting	00:11.327153	00:02.272840	00:01.021486
XGBoost	00:10.436653	00:00.136623	00:00.049223
RF	00:01.996077	00:00.140589	00:00.031241
KNN	00:00.789365	00:06.264449	00:02.012055
Nelder-Mead 1 (KNN-XGB)	N/A	00:03.296877	00:00.497722
Nelder-Mead 2 (RF-XGB)	N/A	00:02.490687	00:01.028049
Nelder-Mead 3 (KNN-RF)	N/A	00:03.617227	00:00.519629

Table 9 shows that the Nelder-Mead optimization is much faster for a smaller number of features. In the KDD99 dataset, the search time for 10 features is only 41 seconds, while the search time for 41 features is over 58 minutes. The Nelder-Mead optimization technique likely serves to identify the lowest or highest point of a target function within a space with multiple dimensions. The method operates quickly and is applicable for multidimensional optimization tasks. However, it does not depend on a gradient, and therefore, convergence may be slow on datasets with a larger number of features. The convergence rate of the method can therefore be slow, especially for highly non-linear functions or functions with many local minima [33].

Table 9. *Search Timing Results with Nelder-Mead Algorithm by Number of Features*

Dataset	Total Number of Features	Optimization Method	Search Duration (mm:ss.000)
KDD99	41	Nelder-Mead 1 (KNN - XGB)	58:29.617903
	10	Nelder-Mead 1 (KNN - XGB)	00:41.802781
UNSW-NB15	49	Nelder-Mead 1 (KNN - XGB)	31:14.104041
	11	Nelder-Mead 1 (KNN - XGB)	00:47.260621

Table 10 displays the confusion matrices for the Nelder-Mead optimized model on the KDD99 and UNSW-NB15 datasets. The model attained remarkable overall accuracy, with KDD99 achieving 99.856% and UNSW-NB15 reaching 97.662%. This level of accuracy confirms the effectiveness of the model in precisely classifying network traffic occurrences as normal or attack.

Our model demonstrated high rates of True Negatives (TN), indicating its ability to accurately detect actual attacks. It accurately classified 11525 attacks in KDD99 and identified 17244 attacks in UNSW-NB15. This indicates that our model is effective in identifying and distinguishing malicious network behavior. Despite registering low False Negative (FN) rates in two sets, our model displayed superior performance on KDD99. It identified only 16 normal instances as attacks, signifying remarkable control over false alarms. However, 340 normal instances were classified as attacks on UNSW-NB15, indicating misclassifications that require further investigation.

Our model demonstrated strong performance in accurately identifying normal network traffic, as indicated by its high True Positive (TP) rates. In the KDD99 dataset, it correctly detected 17551 instances of normal traffic, while in the UNSW-NB15 dataset, it accurately identified 5483 normal instances. This highlights our model's ability to distinguish normal network behavior effectively. Additionally, the model only identified 26 attack instances as normal in KDD99. However, it is worth noting that our model had a relatively higher False Positive (FP) rate of 204 on the UNSW-NB15 dataset. This suggests that there were instances where attacks were missed, falsely classified as normal traffic. Although the model performed well overall, there is room for improvement to decrease the false positive rate and accurately identify more attacks.

Table 10. *Test Dataset Confusion Matrices for Nelder-Mead 1 (KNN - XGB)*

Test Dataset Confusion Matrices		**Predicted Class**					
		KDD99			**UNSW-NB15**		
		Attack	Normal	Acc	Attack	Normal	Acc
Actual Class	Attack	11525	26	99.856%	17244	204	97.662%
	Normal	16	17551		340	5483	

Table 11 presents a comprehensive comparative analysis of the performance of intrusion detection models on 2 benchmark datasets: UNSW-NB15 and NSL-KDD. Accuracy, which measures the proportion of correctly classified intrusion events, is used as the evaluation metric. Our novel model demonstrates exceptional performance on both datasets, achieving an impressive accuracy rate of 97.66% on UNSW-NB15 and an impressive 99.85% on KDD99. These statistically significant results exceed the accuracy of all other models, which range from 85.2% to 99.81%.

In summary, Table 11 presents a comparative analysis that clearly demonstrates the superior performance of our model with fewer features. This indicates that our model can effectively detect intrusion using a smaller number of features, which helps to mitigate overfitting and enhance generalization, leading to more robust and reliable results. The model's accuracy on both datasets, coupled with its implementation of feature selection techniques, positions it as an effective solution for securing network environments.

Table 11. *Comparison with the Other Proposed Methods*

Authors	Datasets	Feature Selection	Number of Selected Features	Algorithms	Most Effective Method	Best Result (Accuracy)
Yao et al. [5]	UNSW-NB15	No	N/A	XGB, LGBM, RF, Soft Voting	Soft Voting	95.23%
Shen et al. [6]	NLS-KDD	Yes	20	SVM, KNN, DT, Soft Voting	Soft Voting	97.49%
	UNSW-NB15		19			94.62%
Swami et al. [7]	NSL-KDD	No	N/A	CART (Classification & Regression Tree), MLP, NB, RF, KNN, Soft Voting	Soft Voting	99.68%
	UNSW-NB15		N/A			89.29%
Zhou et al. [8]	NSL-KDD	Yes	10	DT, RF, ForestPA, Voting	Voting	99.81%
Gao et al. [10]	NSL-KDD	Yes	17	DT, RF, KNN, DNN, MultiTree, Voting	Voting	85.2%
Proposed Model	KDD99	Yes	10	XGB, RF, KNN, Soft Voting, Nelder-Mead Optimized Classifier Level Soft Voting	Nelder-Mead Optimized Classifier Level Soft Voting	99.85%
	UNSW-NB15		11			97.66%

V. CONCLUSION

This study identified the limitation of relying on a single classification algorithm to detect cyber-attacks. The objective of the study was to enhance the accuracy of attack detection by combining the detection capabilities of multiple classifiers. By leveraging the strengths and diversity of multiple classifiers, it is aimed to create an ensemble model that can provide more accurate and robust predictions in detecting attacks. This approach allows us to harness the collective knowledge and capabilities of different classifiers to improve the overall performance of the attack detection system.

In this study, ensemble classification for intrusion detection is proposed using weighted soft voting with KNN, XGBoost, and Random Forest base models. A powerful intrusion detection framework was proposed that utilizes soft voting classifier-level weights optimized by the Nelder-Mead algorithm and incorporates feature selection.

This study demonstrated that optimizing classifier-level weights through the Nelder-Mead algorithm could enhance combinatory performance, yielding comparable results with fewer features. The focus of this paper was to utilize a heuristic algorithm for fusing the base learners to enhance the performance of the ensemble. Future research will focus on refining the existing heuristic algorithms or developing new intelligent heuristic algorithms to further strengthen the ensemble's performance. Additionally, exploring alternative approaches for fusing base learners will be a key direction for future work.

VI. REFERENCES

[1] "Mid-Year Update: 2023 SonicWall Cyber Threat Report", Accessed: Sep. 30, 2023. [Online]. Available: https://www.sonicwall.com/2023-mid-year-cyber-threat-report/

[2] Md Haris Uddin Sharif and Mehmood Ali Mohammed, "A literature review of financial losses statistics for cyber security and future trend," World J. Adv. Res. Rev., vol. 15, no. 1, pp. 138–156, Jul. 2022, doi: 10.30574/wjarr.2022.15.1.0573.

[3] "Cyber Security Market Analysis Report | 2022 - 2030." Accessed: Nov. 05, 2023. [Online]. Available: https://www.nextmsc.com/report/cyber-security-market

[4] P. Spadaccino and F. Cuomo, "Intrusion Detection Systems for IoT: opportunities and challenges offered by Edge Computing and Machine Learning," 2020, doi: 10.48550/ARXIV.2012.01174.

[5] W. Yao, L. Hu, Y. Hou, and X. Li, "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), Baltimore, MD, USA: IEEE, Jun. 2022, pp. 155–161. doi: 10.1109/DSN-W54100.2022.00034.

[6] Y. Shen, K. Zheng, Y. Yang, S. Liu, and M. Huang, "CBA-CLSVE: A Class-Level Soft-Voting Ensemble Based on the Chaos Bat Algorithm for Intrusion Detection," Appl. Sci., vol. 12, no. 21, p. 11298, Nov. 2022, doi: 10.3390/app122111298.

[7] R. Swami, M. Dave, and V. Ranga, "Voting-based intrusion detection framework for securing software-defined networks," Concurr. Comput. Pract. Exp., vol. 32, no. 24, p. e5927, Dec. 2020, doi: 10.1002/cpe.5927.

[8] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," Comput. Netw., vol. 174, p. 107247, Jun. 2020, doi: 10.1016/j.comnet.2020.107247.

[9] J. Gu, L. Wang, H. Wang, and S. Wang, "A novel approach to intrusion detection using SVM ensemble with feature augmentation," Comput. Secur., vol. 86, pp. 53–62, Sep. 2019, doi: 10.1016/j.cose.2019.05.022.

[10] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An Adaptive Ensemble Machine Learning Model for Intrusion Detection," IEEE Access, vol. 7, pp. 82512–82521, 2019, doi: 10.1109/ACCESS.2019.2923640.

[11] S. Seth, K. K. Chahal, and G. Singh, “A Novel Ensemble Framework for an Intelligent Intrusion Detection System,” IEEE Access, vol. 9, pp. 138451–138467, 2021, doi: 10.1109/ACCESS.2021.3116219.

[12] R. Zhang, “Dynamic Weighted Voting Classifier for Network Intrusion Detection,” in 2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE), Guangzhou, China: IEEE, Aug. 2022, pp. 350–354. doi: 10.1109/MLISE57402.2022.00076.

[13] A. Harbola, J. Harbola, and K. S. Vaisla, “Improved Intrusion Detection in DDoS Applying Feature Selection Using Rank & Score of Attributes in KDD-99 Data Set,” in 2014 International Conference on Computational Intelligence and Communication Networks, Bhopal, India: IEEE, Nov. 2014, pp. 840–845. doi: 10.1109/CICN.2014.179.

[14] N. Moustafa and J. Slay, “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set,” Inf. Secur. J. Glob. Perspect., vol. 25, no. 1–3, pp. 18–31, Apr. 2016, doi: 10.1080/19393555.2015.1125974.

[15] A. I. Saleh, F. M. Talaat, and L. M. Labib, “A hybrid intrusion detection system (HIDS) based on prioritized k-nearest neighbors and optimized SVM classifiers,” Artif. Intell. Rev., vol. 51, no. 3, pp. 403–443, Mar. 2019, doi: 10.1007/s10462-017-9567-1.

[16] “A novel SVM-kNN-PSO ensemble method for intrusion detection system,” Appl. Soft Comput., vol. 38, pp. 360–372, Jan. 2016, doi: 10.1016/j.asoc.2015.10.011.

[17] S. Dhaliwal, A.-A. Nahid, and R. Abbas, “Effective Intrusion Detection System Using XGBoost,” Information, vol. 9, no. 7, p. 149, Jun. 2018, doi: 10.3390/info9070149.

[18] P. A. A. Resende and A. C. Drummond, “A Survey of Random Forest Based Methods for Intrusion Detection Systems,” ACM Comput. Surv., vol. 51, no. 3, pp. 1–36, May 2019, doi: 10.1145/3178582.

[19] N. Zhu, C. Zhu, L. Zhou, Y. Zhu, and X. Zhang, “Optimization of the Random Forest Hyperparameters for Power Industrial Control Systems Intrusion Detection Using an Improved Grid Search Algorithm,” Appl. Sci. Switz., vol. 12, no. 20, Oct. 2022, doi: 10.3390/app122010456.

[20] Md. Raihan-Al-Masud and H. A. Mustafa, “Network Intrusion Detection System Using Voting Ensemble Machine Learning,” in 2019 IEEE International Conference on Telecommunications and Photonics (ICTP), Dhaka, Bangladesh: IEEE, Dec. 2019, pp. 1–4. doi: 10.1109/ICTP48844.2019.9041736.

[21] A. Z. Kiflay, A. Tsokanos, and R. Kirner, “A Network Intrusion Detection System Using Ensemble Machine Learning,” in 2021 International Carnahan Conference on Security Technology (ICCST), Hatfield, United Kingdom: IEEE, Oct. 2021, pp. 1–6. doi: 10.1109/ICCST49569.2021.9717397.

[22] A. Mohammed and R. Kora, “A comprehensive review on ensemble deep learning: Opportunities and challenges,” J. King Saud Univ. - Comput. Inf. Sci., vol. 35, no. 2, pp. 757–774, Feb. 2023, doi: 10.1016/j.jksuci.2023.01.014.

[23] J. A. Nelder and R. Mead, “A Simplex Method for Function Minimization,” Comput. J., vol. 7, no. 4, pp. 308–313, Jan. 1965, doi: 10.1093/comjnl/7.4.308.

[24] P. C. Wang and T. E. Shoup, “Parameter sensitivity study of the Nelder–Mead Simplex Method,” Adv. Eng. Softw., vol. 42, no. 7, pp. 529–533, Jul. 2011, doi: 10.1016/j.advengsoft.2011.04.004.

[25] S.-K. S. Fan and E. Zahara, “A hybrid simplex search and particle swarm optimization for unconstrained optimization,” Eur. J. Oper. Res., vol. 181, no. 2, pp. 527–548, Sep. 2007, doi: 10.1016/j.ejor.2006.06.034.

[26] J.-O. Palacio-Niño and F. Berzal, “Evaluation Metrics for Unsupervised Learning Algorithms.” arXiv, May 23, 2019. Accessed: Nov. 21, 2023. [Online]. Available: http://arxiv.org/abs/1905.05667

[27] D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation,” BMC Genomics, vol. 21, no. 1, p. 6, Dec. 2020, doi: 10.1186/s12864-019-6413-7.

[28] C. P. Chai, “The Importance of Data Cleaning: Three Visualization Examples,” CHANCE, vol. 33, no. 1, pp. 4–9, Jan. 2020, doi: 10.1080/09332480.2020.1726112.

[29] S. Khalid, T. Khalil, and S. Nasreen, “A survey of feature selection and feature extraction techniques in machine learning,” in 2014 Science and Information Conference, London, UK: IEEE, Aug. 2014, pp. 372–378. doi: 10.1109/SAI.2014.6918213.

[30] M. Farajzadeh-Zanjani, R. Razavi-Far, and M. Saif, “A Critical Study on the Importance of Feature Extraction and Selection for Diagnosing Bearing Defects,” in 2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS), Windsor, ON, Canada: IEEE, Aug. 2018, pp. 803–808. doi: 10.1109/MWSCAS.2018.8623823.

[31] “Version 1.2.2,” scikit-learn. Accessed: Nov. 29, 2023. [Online]. Available: https://scikit-learn/stable/whats_new/v1.2.html

[32] “scipy: Fundamental algorithms for scientific computing in Python.” Accessed: Nov. 29, 2023. [MacOS, Microsoft :: Windows, POSIX, POSIX :: Linux, Unix]. Available: https://scipy.org/

[33] N. Pham and B. M. Wilamowski, “Improved Nelder Mead’s Simplex Method and Applications,” vol. 3, no. 3, 2011.