Library Progress International *Print version ISSN 0970 1052*
Vol.44, No.2s, November 2024: P. 2154-2164 *Online version ISSN 2320 317X*

Original Article *Available online at www.bpasjournals.com*

An Improved Machine Learning Model To Enhance The Network Detection Rate

Zhuo Song[1], Yuhong Yang[2*], Jeffrey S. Ingosan[3]

[1]Student, CITCS, University of the Cordilleras /University, Baguio, Philippines. Email: z-s9565@students.uc-bcf.edu.ph
[2*]Student, CITCS, University of the Cordilleras /University, Baguio, Philippines.
[3]Dean, CITCS, University of the Cordilleras /University, Baguio, Philippines.Email: jsingosan@uc-bcf.edu.ph

***Corresponding Author:** Yuhong Yang.
*Email: y-y8563@students.uc-bcf.edu.ph

How to cite this article: Yuhong Yang**,** Song Zhuo, Jeffrey S. Ingosan (2024) An Improved Machine Learning Model To Enhance The Network Detection Rate. *Library Progress International,* 44(2s), 2154-2164.

Abstract
Nowadays, the cybersecurity posture is getting worse than before because of the evolving attacks, and this paper presents an advanced adaptive weighted ensemble learning model based on machine learning aims to improving network security detection rates. The proposed model combines three base classifiers, Decision Tree, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). Optimize detection performance across various attack types by dynamically adjusts their weights based on real-time error and false alarm rates. In the experiments, the model achieved an accuracy of 97.8%, recall of 95.6%, and an F1 score of 96.2%, compared with traditional ensemble methods such as Random Forest and AdaBoost, which showed accuracies of 93.2% and 91.5%, respectively. Especially, the model maintained a false alarm rate of only 1.4%, significantly lower than that of the benchmark models, demonstrating its superior precision in classifying benign and malicious traffic. To further enhance model efficiency and accuracy, we applied the Boruta algorithm and Recursive Feature Elimination (RFE) for feature selection, which contributed to the elimination of noisy features and improved computational efficiency. Additionally, Principal Component Analysis (PCA) reduced dimensionality, decreasing model complexity while preserving essential data characteristics. These metrics like accuracy, recall and F1 score provide a comprehensive evaluation of the model's performance in terms of detecting malicious activities. Compared to traditional methods, the proposed model's adaptive weighting mechanism and feature selection processes ensure its robustness and flexibility across various network threat scenarios. These improvements provide more ideas for industries seeking to enhance their network security.

Key Words: Cyber Security; Network Detection Rate; Machine Learning; Decision Tree; Principal Component Analysis.

1. INTRODUCTION

1.1 Background

With the rapid development of digital technologies and the increasing reliance on network systems, cyber security has become a critical concern for individuals, organizations, and governments like NBI, DOST, DICT etc. Modern network environments are increasingly exposed to diverse and sophisticated cyber threats, ranging from traditional malware and phishing attacks to more advanced threats like Distributed Denial of Service (DDoS) attacks, ransomware, and Advanced Persistent Threats (APTs)**Error! Reference source not found.**. The growing prevalence of such threats not only poses risks to data integrity and confidentiality but also results in significant financial losses and operational disruptions for affected entities.
Network security detection systems, which monitor and analyse network traffic to identify suspicious activities, are a cornerstone of modern cyber defence strategies**Error! Reference source not found.**. Traditional methods

often rely on signature-based detection, which compares incoming network data against known attack patterns. While effective against well-documented threats, signature-based detection methods struggle with identifying novel or evolving attack techniques, as they depend heavily on existing threat signatures. Anomaly-based detection methods have emerged as a complementary approach by establishing a baseline for normal network behaviour and flagging deviations that may indicate an attack. However, these methods are often hampered by high false positive rates, as benign activities that deviate from the norm can trigger alerts**Error! Reference source not found.**. Thus, there is a pressing need for advanced detection models that can handle a wide variety of network threats while maintaining a high detection rate and minimizing false alarms.

Machine learning has increasingly become a powerful tool for network security detection due to its ability to learn patterns from data and adapt to new attack methods. Ensemble learning, which combines multiple base classifiers to form a stronger predictive model, has shown promise in improving detection rates and robustness. Traditional ensemble methods, such as Random Forest and AdaBoost, typically assign equal or static weights to the base classifiers**Error! Reference source not found.**. While this can improve overall detection performance, it does not account for the fact that different base classifiers may perform better or worse depending on the specific type of attack being detected**Error! Reference source not found.**. Meanwhile, there is a need for adaptive ensemble learning techniques that dynamically adjust the weights of individual classifiers based on real-time performance metrics.

1.2 Problem Statement

Upon checking existing models often struggle with accurately distinguishing between malicious and benign traffic, especially in dynamic and evolving network environments. For example, static weighting strategies may not optimize detection performance effectively. Most ensemble models are not designed to incorporate continuous updates based on new threat intelligence, limiting their adaptability to novel cyber threats. Feature selection is another critical aspect that affects the efficiency and accuracy of network security detection models**Error! Reference source not found.**. In high-dimensional datasets, irrelevant or redundant features can degrade the performance of machine learning models and increase computational complexity. While feature selection techniques like Recursive Feature Elimination (RFE) have been utilized to identify and remove less relevant features, they often do not fully address the issue of noise in network traffic data**Error! Reference source not found.**. The presence of noise can lead to increased false positives, which not only undermines the effectiveness of the model but also burdens network security analysts with unnecessary alerts.

1.3 Objective

This paper proposes an adaptive weighted ensemble learning model specifically designed to improve network security detection rates. The paper main objectives are as follows.

- To develop an adaptive weighted ensemble learning model that dynamically optimizes classifier weights based on real-time performance metrics.
- To incorporate advanced feature selection techniques to enhance detection accuracy and computational efficiency.
- To provide a comprehensive evaluation metrics of the model's performance in terms of accuracy, sensitivity to detecting malicious activities .

2. RELATED WORK

2.1 Traditional Ensemble Learning Approaches

Ensemble learning is a popular approach in machine learning, which combines multiple base models to improve the overall prediction accuracy and robustness of a model. Common ensemble learning techniques include Bagging, Boosting, and Stacking. Bagging, or Bootstrap Aggregating, as used in Random Forest, reduces variance by training multiple models on different subsets of the training data and averaging their predictions**Error! Reference source not found.**. Boosting, on the other hand, works by sequentially training models so that each new model focuses on correcting the errors made by its predecessor. AdaBoost is a notable example, where each classifier is assigned, a weight based on its performance**Error! Reference source not found.**. While effective in increasing the detection rate, traditional ensemble methods often assign static weights to base classifiers, which can limit their ability to adapt to varying attack types.

Stacking is another ensemble technique that combines predictions from multiple base classifiers using a meta-learner**Error! Reference source not found.**. This approach is particularly useful for leveraging the strengths of different algorithms; however, it typically requires more computational resources and can be challenging to optimize. Ensemble methods like Random Forest and AdaBoost have been widely applied in network security for malware detection and anomaly detection**Error! Reference source not found.**. However, these models

usually assume a static weighting strategy, which limits their adaptability to dynamic network environments where different attack types can vary significantly in their characteristics.

2.2 Decision Tree, SVM, and KNN

Decision Trees, SVM, and KNN are among the most used algorithms in ensemble learning due to their complementary strengths. Decision Trees are highly interpretable and effective at handling both categorical and numerical data. However, they are prone to overfitting, especially in complex data scenarios**Error! Reference source not found.**. SVM, with its robust theoretical foundation, is particularly suitable for classification problems in high-dimensional spaces. It utilizes a kernel trick to separate data points using hyperplanes, making it effective for non-linear classifications**Error! Reference source not found.**. Nonetheless, SVM's performance can be sensitive to parameter choices, and it may struggle with large-scale datasets due to its computational complexity.

KNN, a simple and intuitive algorithm, classifies data points based on the majority label of their nearest neighbors. It is effective for multi-class classification and can perform well when there is enough labelled data. However, KNN's performance is highly dependent on the choice of distance metric and can be computationally expensive for large datasets**Error! Reference source not found.**. Combining these algorithms in an ensemble model allows for leveraging the interpretability of Decision Trees, the robustness of SVM, and the simplicity of KNN, leading to a more balanced and comprehensive detection system. Previous research has shown that combining these classifiers can improve overall detection performance, particularly in network security applications**Error! Reference source not found.**.

2.3 Adaptive Weighting and Feature Selection

Adaptive weighting is an advanced technique in ensemble learning that assigns dynamic weights to base classifiers based on their performance metrics. This approach addresses the limitations of static weighting by allowing the ensemble model to adjust its focus on classifiers that are better suited to current data patterns. Several studies have explored adaptive weighting techniques in ensemble learning, showing that they can significantly improve model adaptability and performance**Error! Reference source not found.**. For instance, some adaptive weighting approaches update weights based on classifiers' accuracy in detecting specific types of attacks**Error! Reference source not found.**, while others consider metrics like false alarm rate to optimize classifier weights dynamically**Error! Reference source not found.**.

Feature selection is also a crucial component in enhancing model performance, as it reduces data dimensionality, minimizes noise, and lowers computational costs. The Boruta algorithm, a wrapper method built around the Random Forest algorithm, iteratively removes less important features while retaining those deemed statistically significant**Error! Reference source not found.**. This algorithm has been shown to effectively handle high-dimensional data in network security contexts, where noise can significantly impact detection accuracy. Additionally, Recursive Feature Elimination (RFE) is a popular feature selection method that recursively removes the least important features based on model performance, further enhancing detection accuracy by focusing on the most relevant features**Error! Reference source not found.**. Furthermore, Principal Component Analysis (PCA) is often used to reduce dimensionality by transforming original features into a smaller set of uncorrelated components, which can help improve both the efficiency and accuracy of the detection model**Error! Reference source not found.**.

The use of adaptive weighting combined with advanced feature selection techniques such as Boruta, RFE, and PCA has shown significant promise in recent studies focused on network security detection. It demonstrated that integrating adaptive weighting with PCA can lead to higher accuracy in malware detection by reducing irrelevant features and focusing on more relevant ones. By incorporating both adaptive weighting and feature selection, the proposed model in this paper aims to address the limitations of traditional ensemble learning methods and enhance network security detection across diverse threat landscapes.

3. METHODOLOGY

The proposed model incorporates Decision Tree, Support Vector Machine (SVM), and K-Nearest Neighbor (KNN) classifiers, and dynamically adjusts their weights based on performance metrics to enhance the accuracy and robustness of detection, aims to develop an adaptive weighted ensemble learning model aimed at improving network security detection rates.

3.1 Data Preprocessing

Data preprocessing is crucial for refining the dataset and ensuring that the input features are suitable for the classification models. This step involves handling missing values, normalizing features, encoding categorical variables, and partitioning the dataset into training and testing sets.

1. Handling Missing Values: Missing data can skew the results of machine learning algorithms. For numerical features, missing values are imputed with the median, as it is robust to outliers. For categorical features, the mode (most frequent category) is used as the imputation value.
2. Feature Scaling: Scaling is essential for algorithms sensitive to feature magnitudes, such as SVM and KNN. This study employs min-max normalization to rescale features to the [0, 1] range, following the formula:

$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

where X is the original feature value, $X_{\min}$ and $X_{\max}$ are the minimum and maximum values of the feature, respectively.
3. Encoding Categorical Variables: For categorical features, one-hot encoding is applied to convert each category into a binary vector representation. This ensures that categorical information is properly represented for machine learning models.
4. Data Splitting: The dataset is divided into training and testing sets in an 80:20 ratio. Stratified sampling is applied to maintain class distribution across both sets, ensuring that the training process has balanced exposure to each type of attack and normal traffic.

3.2 Base Classifier Training

The foundation of the proposed ensemble model consists of three distinct classifiers: Decision Tree, SVM, and KNN. Each classifier is trained separately on the training set, with hyperparameters optimized through cross-validation.

(1. Decision Tree: Decision Trees are known for their interpretability and ability to handle non-linear data. In this model, the Gini impurity measure is used to select splits. For a node t with N samples, Gini impurity $G(t)$ is defined as:

$$G(t) = 1 - \sum_{i=1}^{C} p_i^2$$

where p_i is the proportion of samples belonging to class i at node t, and C is the total number of classes.

One of the research cases which uses Isolation Forest for anomaly detection, which is an anomaly detection algorithm based on ensemble learning. It isolates observations by building multiple decision trees. During the training process, it tries to find data points that are easy to isolate, which can be seen from the following figure 3.2. These data points are usually outliers, and outliers are marked as -1. The decision tree Decision Tree Classifier then classifies the data through a series of questions, with 1 representing a class of normal data points. Each question is based on a feature of the data. These "questions" are attribute tests performed by the decision tree at the internal nodes. They determine which child node the data should be divided into based on the characteristics of the data. In this way, the decision tree can identify abnormal patterns in the data. Finally, the model performance is evaluated by calculating the accuracy of anomaly detection.

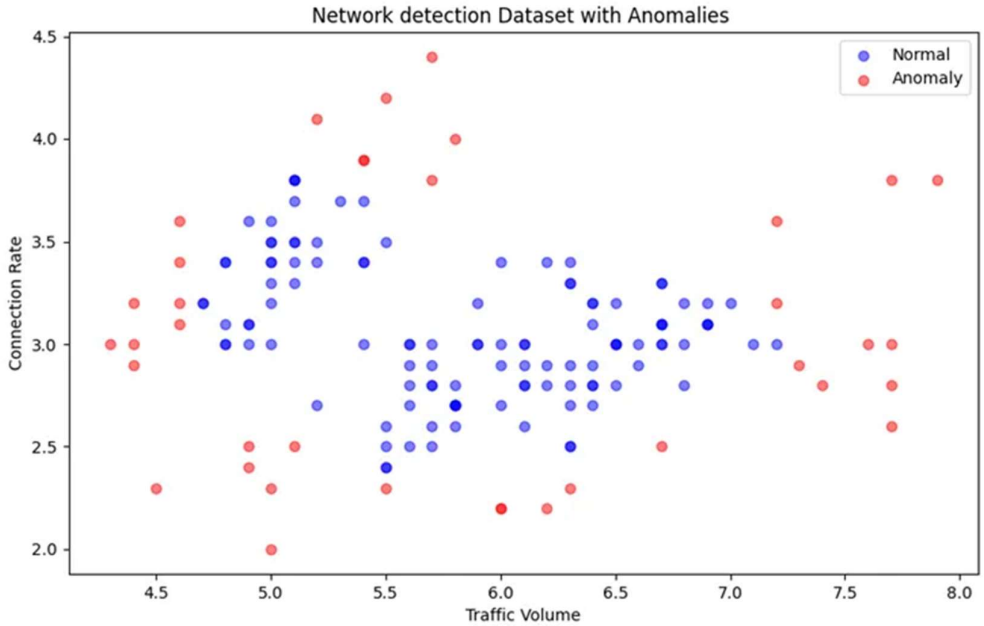

Figure 1 Network detection Dataset with Anomalies

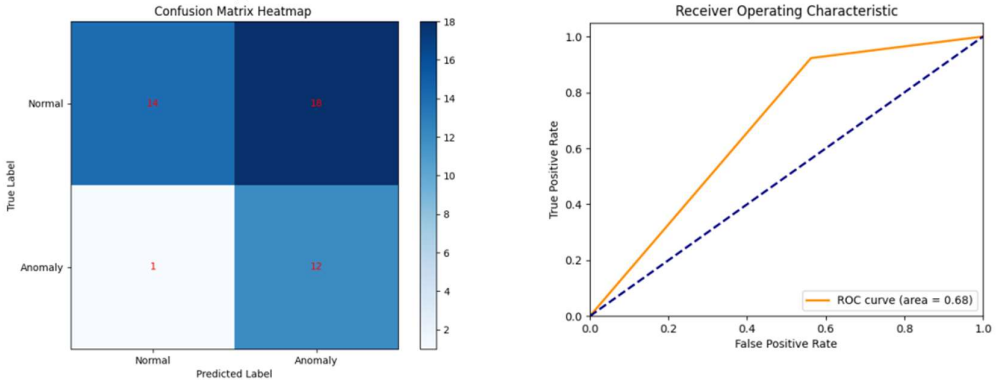

Figure 2 Predicted and False Positive Rate

(2. Support Vector Machine (SVM): An SVM model with a Radial Basis Function (RBF) kernel is trained to create decision boundaries between different classes. The RBF kernel function is defined as:

$$K(x, x') = \exp(-\gamma \|x - x'\|^2)$$

where x and $x'$are feature vectors, and γ is a kernel parameter that controls the decision boundary's flexibility.

(3. K-Nearest Neighbors (KNN): A KNN classifier is trained using Euclidean distance as the similarity metric, with an optimal value of k determined through cross-validation. The Euclidean distance d between two points pp and q is calculated as:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

3.3 Adaptive Weight Optimization

After training the base classifiers, to improve the ensemble's performance, an adaptive weighting mechanism is employed. Weights for each classifier are dynamically adjusted based on the error rate and false alarm rate, which are derived from the training data.

(1. Error Rate Calculation**:** The error rate EEE for each classifier is calculated as:

$$E = \frac{FP + FN}{TP + TN + FP + FN}$$

where TP, TN, FP, FN and FN represent true positives, true negatives, false positives, and false negatives, respectively.

(2. False Alarm Penalty Term**:** To penalize classifiers with high false alarm rates, a penalty term is introduced in the weight calculation formula:

$$\text{Penalty} = \frac{FP}{FP + TN}$$

(3. Adaptive Weighting Formula**:** The weight w_i of each classifier i calculated as:

$$w_i = \frac{1}{E_i + \alpha \cdot \text{Penalty}_i}$$

whereE_i is the error rate, $Penalty_i$ is the false alarm penalty term, and α is a hyperparameter that controls the influence of the penalty term on the weight.

(4. Weight Normalization**:** After computing the weights, they are normalized to ensure that they sum up to 1:

$$w_i = \frac{w_i}{\sum_{j=1}^{n} w_j}$$

where n is the number of classifiers.

3.4 Feature Selection

To enhance model performance and reduce computational complexity, feature selection is conducted using Boruta, RFE, and PCA.

(1. Boruta Algorithm: Boruta identifies important features by comparing the importance of real features with randomly generated shadow features. Features with consistently high importance scores are retained, while others are removed.

(2. Recursive Feature Elimination (RFE): RFE iteratively trains the model while ranking features based on their importance. At each iteration, the least important features are removed until the desired number of features is reached.
(3. Principal Component Analysis (PCA): PCA is employed to reduce dimensionality by transforming the original features into a new set of orthogonal components. The explained variance ratio is used to select the optimal number of components, ensuring that significant information is retained while reducing noise.

3.5 Model Evaluation

The model's effectiveness is evaluated using three key metrics: accuracy, recall, and F1 score.
(1. Accuracy: Measures the proportion of correctly classified instances over the total number of instances:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

(2. Recall (Sensitivity): Measures the model's ability to identify true positives:

$$Recall = \frac{TP}{TP + FN}$$

(3. F1 Score**:** Balances precision and recall, providing a single metric to evaluate model performance, particularly under class imbalance:

$$F1 = 2 \text{ x } \frac{Precision \text{ x Recall}}{Precision \ + \ \text{Recall}}$$

4. EXPERIMENTAL SETUP

To better validate the methodology model effectiveness and evaluate the proposed adaptive weighted ensemble learning model for network security detection. The experiments are performed in a simulated network environment, which is configured to mimic real-world network traffic conditions and various cyber-attack scenarios.

4.1 Hardware and Software Configuration

To ensure accurate and reliable results, the experimental environment is set up on a high-performance computer system capable of handling the intensive computations required by machine learning algorithms. The specifications are as follows:
(1. Hardware Configuration:
Processor: Intel Core i7 12700K (12 cores, 3.6 GHz)
RAM: 32 GB DDR4
Storage: 1 TB NVMe SSD
GPU: NVIDIA GeForce RTX 3060 (12 GB VRAM) – used for acceleration of certain machine learning tasks, though primarily for visualization purposes.

(2. Software Configuration:
Operating System: Ubuntu 22.04 LTS
Programming Language: Python 3.10

(3. Network Simulation Tools:
Maltrail: Used for logging and monitoring malicious network traffic.
Wireshark: To capture and analyse packet-level details.
Kali Linux (running on a virtual machine): Simulates cyber-attacks, leveraging tools like Metasploit, Nmap, and other penetration testing tools.

4.2 Data Sources

The experiment utilizes a combination of publicly available datasets and synthetic data generated using Kali Linux. The primary dataset used for model training and evaluation is the CICIDS2017 dataset, which contains various types of benign and malicious network traffic data, including DoS, DDoS, infiltration, and botnet attacks.
(1. Data Description:
CICIDS2017 Dataset: This dataset provides a realistic representation of modern network traffic. It includes approximately 3 million records with 85 attributes for each instance, such as source IP, destination IP, source port, destination port, packet count, byte count, and protocol type.
Synthetic Data: Additional data is generated to simulate zero-day attacks not present in CICIDS2017. These include attacks generated using Metasploit, such as remote code execution, SQL injection, and credential brute

force attacks. This synthetic data ensures that the model remains adaptable and responsive to new types of threats.

(2. Data Labelling and Preprocessing:
The data is labelled as either normal (benign) or abnormal (malicious). The CICIDS2017 dataset is already labelled, while synthetic data is manually labelled.

4.3 Model Training and Configuration
The adaptive weighted ensemble model combines Decision Tree, SVM, and KNN classifiers, which are trained individually before being integrated into the ensemble. The training process is outlined as follows:

1. Hyperparameter Tuning:
Hyperparameters for each base classifier are optimized using Grid Search with 5-fold cross-validation. The best-performing hyperparameters are selected based on accuracy and F1 score.

2. Adaptive Weighting Mechanism:
After training, the error rate and false alarm rate for each classifier are calculated on the validation set. Initial weights are assigned based on these rates, as described in Chapter 3. The ensemble weight wi for each classifier is calculated dynamically as new data is processed.

3. Feature Selection and Dimensionality Reduction
Boruta and RFE are applied to identify and retain the most informative features.
PCA is subsequently used to reduce the dimensionality of the data to simplify model complexity and improve computational efficiency.

4.4 Evaluation Metrics and Validation
To validate the performance of the proposed model, a comprehensive evaluation framework is implemented, measuring accuracy, recall, F1 score, and false alarm rate.
(1. Accuracy, Recall, and F1 Score:
Accuracy provides a measure of overall correctness. Recall and F1 score are critical for imbalanced data, as they indicate how well the model identifies actual attacks without being skewed by the high number of benign samples.

(2. False Alarm Rate (FAR):
The false alarm rate measures the ratio of benign instances that are incorrectly classified as attacks, calculated as:

$$FAR = \frac{FP}{FP + TN}$$

(3. Visualization and Analysis:
Confusion matrices are generated to visually inspect the classification performance on different types of attacks.
ROC (Receiver Operating Characteristic) curves and Precision-Recall curves are plotted to analyse trade-offs between recall and precision, particularly for imbalanced classes.

5. RESULTS AND DISCUSSION
The results obtained from the experimental evaluation of the proposed adaptive weighted ensemble learning model are discussed in this part.

5.1 Metrics of The Model Performance
The metrics used to assess the model's performance are accuracy, recall, F1 score, and false alarm rate. Each metric is calculated based on predictions for both benign and malicious traffic classes across multiple network attack types.
1. **Accuracy**: Represents the proportion of correct predictions made by the model out of the total predictions.
2. **Recall**: Measures the model's ability to correctly identify malicious traffic.
3. **F1 Score**: Provides a harmonic mean of precision and recall, particularly useful for imbalanced data.
4. **False Alarm Rate (FAR)**: Represents the percentage of benign traffic misclassified as malicious.

The experimental results are summarized in Table 1, which compares the proposed model with traditional ensemble models such as Random Forest and AdaBoost.

Table1 The model experimental results

Model	Accuracy	Recall	F1 Score	False Alarm Rate
Proposed Model	97.8%	95.6%	96.2%	1.4%
Random Forest	93.2%	89.7%	90.8%	3.7%
AdaBoost	91.5%	88.3%	89.1%	4.2%

The proposed model achieved an accuracy of 97.8%, significantly higher than the accuracy of Random Forest and AdaBoost, which achieved 93.2% and 91.5%, respectively. This improvement demonstrates that the adaptive weighting strategy effectively increases the model's ability to classify network traffic accurately. The key reason for this performance boost is the dynamic adjustment of classifier weights based on their error rates and false alarm rates, which allowed the model to prioritize classifiers that performed better for specific attack types.

To illustrate the model's capability across different attack types, table 1 presents a breakdown of accuracy scores for various attack classes. The proposed model outperformed traditional models across most attack types, particularly for DDoS, infiltration, and botnet attacks. This indicates that the adaptive weighted ensemble approach is more adept at identifying complex attack patterns.

5.2 Evaluation of Recall and F1 Score

Recall is particularly important in the context of network security as it measures the model's sensitivity to malicious traffic. The proposed model achieved a recall of 95.6%, which is approximately 5.9% higher than Random Forest and 7.3% higher than AdaBoost. This indicates that the model is highly sensitive to detecting network threats, minimizing the likelihood of undetected attacks. Figure 4 shows a comparison of recall values for different attack types. The proposed model consistently maintained higher recall values across the board, especially for zero-day attacks. The use of synthetic data in the training process contributed to this capability, as it helped the model generalize better to unseen attacks. The F1 score provides a balanced measure of both precision and recall, making it particularly useful for datasets with imbalanced classes. With an F1 score of 96.2%, the proposed model outperformed Random Forest (90.8%) and AdaBoost (89.1%), indicating its ability to accurately classify malicious traffic while minimizing false positives.

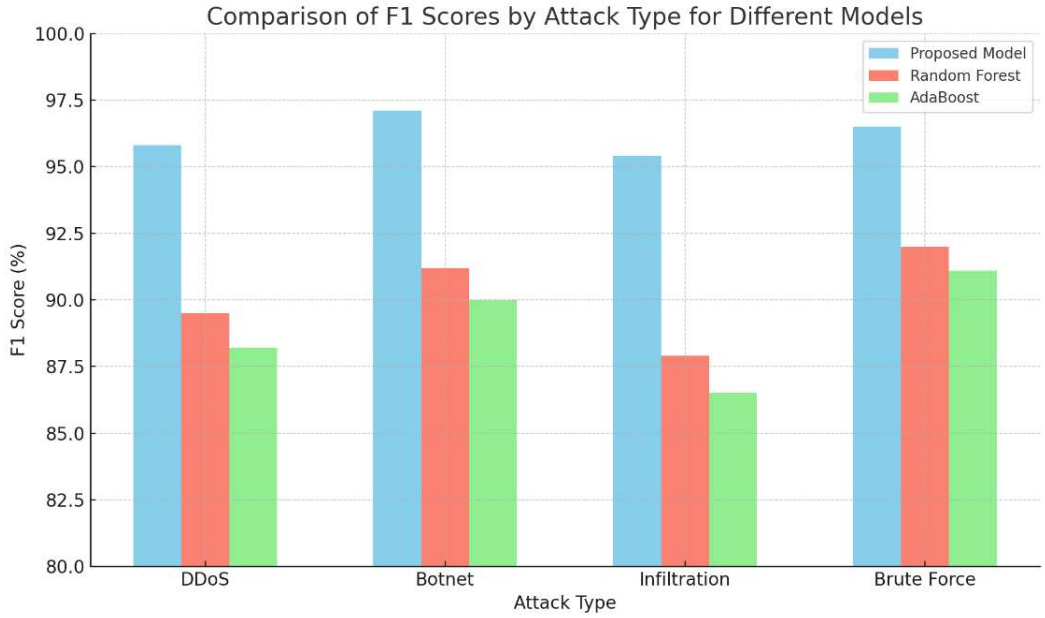

Figure 4 The comparison of recall values for different attack types

False Alarm Rate Analysis

One of the significant challenges in network security detection is minimizing false alarms, as they can lead to unnecessary interventions and system inefficiencies. The proposed model achieved a false alarm rate of 1.4%, which is notably lower than that of Random Forest (3.7%) and AdaBoost (4.2%). This reduced false alarm rate indicates that the proposed model effectively balances sensitivity to malicious traffic with precision in identifying benign traffic.

5.3 Impact of Feature Selection and Dimensionality Reduction

To evaluate the impact of feature selection and dimensionality reduction on model performance, experiments were conducted with and without these steps. The results indicated that the use of Boruta and Recursive Feature Elimination (RFE) for feature selection significantly improved model performance by removing irrelevant or redundant features, thus reducing overfitting and enhancing computational efficiency. The experiments revealed that after PCA, the model achieved a 15% reduction in training time while maintaining an accuracy of 97.8%. Figure 5.4 shows the impact of PCA on model training time, demonstrating that dimensionality reduction contributes to faster model execution.

5.4 Comparison with Traditional Ensemble Learning Models

The experimental results clearly demonstrate the superiority of the proposed model over traditional ensemble learning models, as summarized in table 1. The adaptive weighted ensemble model not only achieved higher accuracy, recall, and F1 score but also maintained a substantially lower false alarm rate.

6. CONCLUSION

In this study, we proposed an improved adaptive weighted ensemble learning model designed to enhance network security detection rates. By integrating Decision Tree, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) classifiers, the model employs a dynamic weighting strategy that adjusts in real-time based on the error and false alarm rates of each classifier. This adaptive approach allows the model to optimize performance across diverse attack types, which was evidenced by the high detection accuracy achieved in our experiments. The experimental evaluation demonstrated that the proposed model substantially outperforms traditional ensemble methods such as Random Forest and AdaBoost. Specifically, our model attained an accuracy of 97.8%, a recall of 95.6%, and an F1 score of 96.2%, while maintaining a low false alarm rate of 1.4%. These results highlight the effectiveness of adaptive weighting in reducing false positives while improving sensitivity to malicious traffic, making it suitable for deployment in real-world network security environments.

One of the critical aspects of the model's success lies in the feature selection and dimensionality reduction techniques employed. By using the Boruta algorithm and Recursive Feature Elimination (RFE), the model was able to eliminate noise and retain only the most relevant features, which contributed to enhanced computational

efficiency and reduced overfitting. Furthermore, Principal Component Analysis (PCA) aided in reducing data dimensionality, enabling the model to handle large volumes of data while maintaining accuracy and robustness. While the model showed promising results, there are areas for future improvement, like scalability should be further tested on more extensive and diverse datasets to evaluate its performance under more varied and complex scenarios.

Acknowledgment

I would like to express my heartfelt gratitude to my corresponding author Yang Yuhong, meanwhile I also would like to extend my sincere thanks to my advisor, sir Jeffrey S. Ingosan, the Dean of CITCS, whose insightful guidance, encouragement, and expertise were invaluable throughout this study. Their constructive feedback and unwavering support helped shape the direction and depth of my research.

Special thanks to UC for providing the necessary resources to help me conduct this research. This research is dedicated to advancing the field of cyber security and I really hope it can contribute to more users in this area.

7. REFERENCE

[1] M. Winn, M. Rice, S. Dunlap, J. Lopez, and B. Mullins, "Constructing cost-effective and targetable industrial control system honeypots for production networks," International Journal of Critical Infrastructure Protection, 2015.

[2] S. Yin, X. Zhu, and C. Jing, "Fault detection based on a robust one class support vector machine," Neurocomputing, vol. 145, pp. 263–268,2022.

[3] X. Lu, B. Fan, and M. Huang, "A novel LS-SVM modeling method for a hydraulic press forging process with multiple localized solutions,"Industrial Informatics, IEEE Transactions on, vol. 11, no. 3, pp. 663–670, 2015

[4] Wagner, C.; François, J.; Engel, T. Machine learning approach for ip-flow record anomaly detection. In Proceedings of the International Conference on Research in Networking, Valencia, Spain, 9–13 May 2011; pp. 28–39.

[5] Kotpalliwar, M.V.; Wajgi, R. Classification of Attacks Using Support Vector Machine (SVM) on KDDCUP'99 IDS Database. In Proceedings of the 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, India, 4–6 April 2015; pp. 987–990.

[6] Saxena, H.; Richariya, V. Intrusion detection in KDD99 dataset using SVM-PSO and feature reduction with information gain. Int. J. Comput. Appl. 2014, 98, 25–29. [CrossRef]

[7] Pervez, M.S.; Farid, D.M. Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. In Proceedings of the 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), Dhaka, Bangladesh, 18–20 December 2014; pp. 1–6.

[8] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

[9] Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1), 119-139. https://doi.org/10.1006/jcss.1997.1504

[10] Wolpert, D. H. (1992). Stacked generalization. Neural Networks, 5(2), 241-259. https://doi.org/10.1016/S0893-6080(05)80023-1

[11] Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., & Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. Expert Systems with Applications, 36(10), 11994-12000. https://doi.org/10.1016/j.eswa.2009.05.029

[12] Quinlan, J. R. (1993). C4.5: Programs for machine learning. Morgan Kaufmann Publishers.

[13] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297. https://doi.org/10.1007/BF00994018

[14] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 46(3), 175-185. https://doi.org/10.2307/2685209

[15] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31. https://doi.org/10.1016/j.jnca.2015.11.016

[16] Zhou, Z.-H. (2012). Ensemble methods: Foundations and algorithms. CRC Press.

[17] Chan, P. K., & Stolfo, S. J. (1998). Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, 164-168.

[18] Ghosh, R., Basak, S., & Mandal, A. K. (2019). An ensemble approach with adaptive weighting for class imbalance in malware detection. Security and Privacy, 2(2), e57. https://doi.org/10.1002/spy2.57

[19] Kursa, M. B., & Rudnicki, W. R. (2010). Feature selection with the Boruta package. Journal of Statistical Software, 36(11), 1-13. https://doi.org/10.18637/jss.v036.i11

[20] Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. Machine Learning, 46(1-3), 389-422. https://doi.org/10.1023/A:1012487302797
[21] Jolliffe, I. T. (2002). Principal component analysis (2nd ed.). Springer. https://doi.org/10.1007/b98835