



A MACHINE LEARNING APPROACH FOR INTRUSION DETECTION

V. Bhavya Lahari¹, T. Amrutha², S.N.B. Tanuja Reddy³, P. Deepika Leela⁴,
Y. Venkata Narayana⁵

^{1,2,3,4}Students, ⁵Assistant Professor

¹Department of Information Technology

¹Vasireddy Venkatadri Institute of Technology, Guntur, India

Article DOI: <https://doi.org/10.36713/epra14771>

DOI No: 10.36713/epra14771

ABSTRACT

In the rapidly evolving digital landscape, ensuring the security of computer systems and networks has become a paramount concern. Traditional methods often struggle to keep pace with the increasing complexity and diversity of cyber threats. In the context of cybersecurity and network analysis, the crucial tools for identifying and responding to unauthorized access, malicious activities, and potential threats within a network or system cannot take place. The significance of intrusion detection is underscored by the escalating frequency and sophistication of cyberattacks that can lead to data breaches, service disruptions, and financial losses. The XGBoost algorithm's ability to handle complex, imbalanced datasets and its exceptional performance in various domains make it a promising candidate for improving intrusion detection accuracy. To evaluate the efficacy of the XGBoost algorithm, extensive experiments are conducted using the KDDCup99 dataset. This dataset represents diverse network traffic scenarios and intrusion types, providing a comprehensive evaluation environment. The Random Forest algorithm, known for its robustness in handling classification tasks, is selected as a baseline for comparison. The performance of the XGBoost algorithm is assessed using multiple performance metrics, including accuracy, precision, recall, F1-score. The outcomes highlight the potential advantages of utilizing XGBoost for intrusion detection.

KEYWORDS: Cybersecurity, Network Analysis, Intrusion Detection, XGBoost Algorithm, Random Forest Algorithm, KDDCup99 Dataset.

1. INTRODUCTION

Cybercrimes refer to a broad category of illegal activities committed in the digital realm, often leveraging technology and the internet. Cyber-crime can be defined as any crime which happens online where an electronic device is considered as a tool to perform such crimes [1][2]. These crimes encompass a wide range of malicious actions, including hacking, identity theft, phishing, malware distribution, and various forms of online fraud. CyberCriminals could potentially take advantage of weaknesses in computer systems, networks, or software to illicitly enter, pilfer sensitive information, or disrupt online operations. Due to the growing interconnectivity of our digital world, cybercrimes now pose a substantial threat to individuals, businesses, and governments alike.

Cybersecurity attacks are malicious activities that target computer systems, networks, or individuals with the intent of stealing, damaging, or otherwise compromising the integrity, confidentiality, or availability of information. Some common types of cybersecurity attacks include as shown in Fig.1: Malware, Phishing, Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks, Man-in-the-Middle (MitM) attacks, Password attacks, SQL Injection, Cross-Site Scripting (XSS), Zero-Day Exploits, DNS Tunneling, Crypto jacking, Ransomware. These are just a few examples of the many types of cybersecurity attacks that exist. Organizations must employ a comprehensive security strategy, including firewalls, intrusion detection and prevention systems, encryption, antivirus software, employee training, and regular security audits, to protect against these and other threats [3].



Fig.1: Types of Cyber Attacks [4]

It is vital to design a powerful intrusion detection system in order to prevent computer hackers and other intruders from effectively getting into computer networks or systems [5]. Machine learning based intrusion detection system is particularly useful in effectively processing the enormous data, detecting any harmful behaviour, and efficiently controlling and promptly identifying any attacks of such sorts [6]. IDS systems based on ML (Machine Learning), as well as DL (Deep Learning), have recently been deployed as feasible procedures for swiftly identifying network intrusions [7].

2. OBJECTIVES

- **Evaluate Intrusion Detection Accuracy:** Assess the performance of intrusion detection using the XGBoost algorithm and determine its accuracy in identifying unauthorized access, malicious activities, and potential threats within computer systems and networks.
- **Benchmark with Random Forest:** Conducting a comparative analysis with the Random Forest algorithm to measure the efficacy of XGBoost in intrusion detection. Determine whether XGBoost outperforms or complements the existing Random Forest system.
- **Handling Diverse Scenarios:** Evaluating how well XGBoost adapts to diverse network traffic scenarios and intrusion types by using benchmark dataset like KDDCup99 which represent different real-world situations.
- **Strengthen Cybersecurity Defenses:** Ultimately, the primary objective is to enhance cybersecurity defenses by adopting a more effective intrusion detection approach, leading to reduced data breaches, service disruptions, and financial losses.

3. METHODOLOGY

The methodology employed in this research project for enhancing intrusion detection using the XGBoost algorithm consists of several key steps. It begins with the collection of benchmark datasets, such as KDDCup99 which represent diverse network traffic scenarios and intrusion types. Intrusion detection is one of the looms to resolve the problem of network security [8]. The data undergoes thorough preprocessing, including data cleansing and normalization. The XGBoost algorithm is selected for its capabilities in handling complex, imbalanced datasets. The model is trained, and its performance is evaluated using established metrics like accuracy, precision, recall, and F1-score [9]. To provide a benchmark, the Random Forest algorithm is also implemented and tested. Controlled experiments are conducted, and the results are analyzed to identify where XGBoost excels and how it can potentially enhance intrusion detection. XGBoost or extreme Gradient Boosting helps in exploiting every bit of memory and hardware resources for tree boosting algorithms [10]. The research culminates in conclusive findings and actionable recommendations for the adoption or adaptation of XGBoost in intrusion detection systems.

Here's a detailed methodology broken down into steps:

3.1 Data Collection

- Acquire relevant datasets, including the KDDCup99 which encompass different network traffic scenarios and intrusion types.

- Conduct data quality checks to ensure the datasets are reliable and free from errors. Address any inconsistencies or missing values.

3.2 Data Preprocessing

- Perform data cleansing to handle issues like missing data, outliers, and irrelevant features. Ensure that the data is consistent and suitable for analysis.
- Normalize or scale the data, if necessary, to bring it into a consistent format for model training.

3.3 Algorithm Selection

- Choose the XGBoost algorithm as the primary tool for intrusion detection enhancement.
- The selection of the XGBoost algorithm as the primary tool for enhancing intrusion detection is justified by its exceptional capabilities that align with the specific challenges posed by its task:
 - **Handling Complex, Imbalanced Datasets:** XGBoost is renowned for its robustness in handling complex and imbalanced datasets. In the context of intrusion detection, data often includes rare intrusion types that can be overshadowed by more common benign traffic. XGBoost's ability to manage imbalanced data ensures that these rare intrusions are not overlooked, thereby improving overall detection accuracy.
 - **Ensemble Learning and Gradient Boosting:** XGBoost employs an ensemble learning approach that combines multiple weak learners to create a strong, accurate model as shown in the fig.2. It uses gradient boosting techniques to iteratively improve the model's performance, making it well-suited for capturing complex data relationships and patterns within the network traffic data.

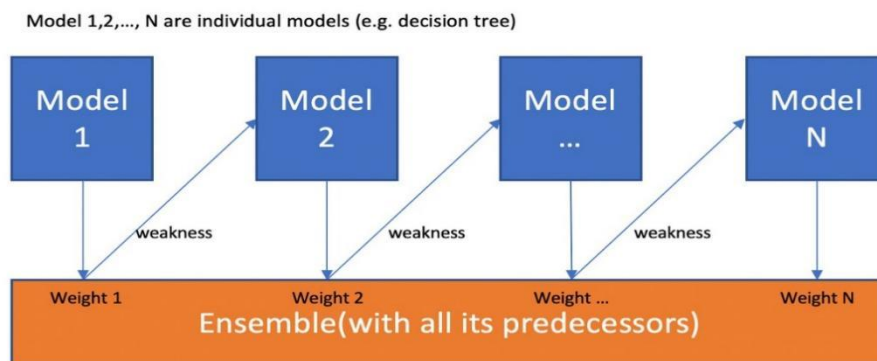


Fig.2: Boosting [11]

Boosting Algorithm

- **Initialize Weight:** Assign equal weights to all observations in the dataset. If we have N observations, each observation's initial weight is $1/N$ as shown in the Fig.2.
- **Build Weak Learners:** Fit a weak learner to the data. A model that outperforms random chance by a small margin is called a weak learner.
- **Compute Error:** Calculate the error of the weak learner. This can be the sum of the weights of the misclassified observations.
- **Compute Learner Weight:** Calculate the weight of the weak learner in the ensemble. This is usually based on the error; a lower error will result in a higher weight.
- **Update Weights:** Increase the weights of the misclassified observations and decrease the weights of the correctly classified observations. This makes the misclassified observations more important for the next weak learner.
- **Normalize Weights:** Normalize the weights so that they sum up to 1.
- **Repeat:** Repeat steps 2 to 6 for a predefined number of boosting rounds or until the error is minimized.

3.4 Model Training

- Implement the XGBoost algorithm and configure its hyperparameters, including learning rate, maximum depth, and the number of boosting rounds.
- Train the XGBoost model using the preprocessed datasets, ensuring that the data is divided into training and testing sets for evaluation

3.5 Performance Evaluation

- Define a set of performance metrics, including accuracy, precision, recall, and F1-score, to quantitatively assess the effectiveness of the XGBoost model.
- Apply these metrics to the model's predictions on both benchmark datasets to evaluate its performance.

3.6 Baseline Comparison

- Implement the Random Forest algorithm as a baseline for performance comparison.
- Train the Random Forest model using the same preprocessed datasets and evaluate its performance using the same metrics.

3.7 Data Analysis

- Interpret the results to identify where XGBoost excels or complements the Random Forest algorithm in intrusion detection.
- Determine the specific scenarios in which XGBoost demonstrates superior intrusion detection capabilities.

4. ARCHITECTURE

The below Picture Fig.3 represents the Architecture of the System.

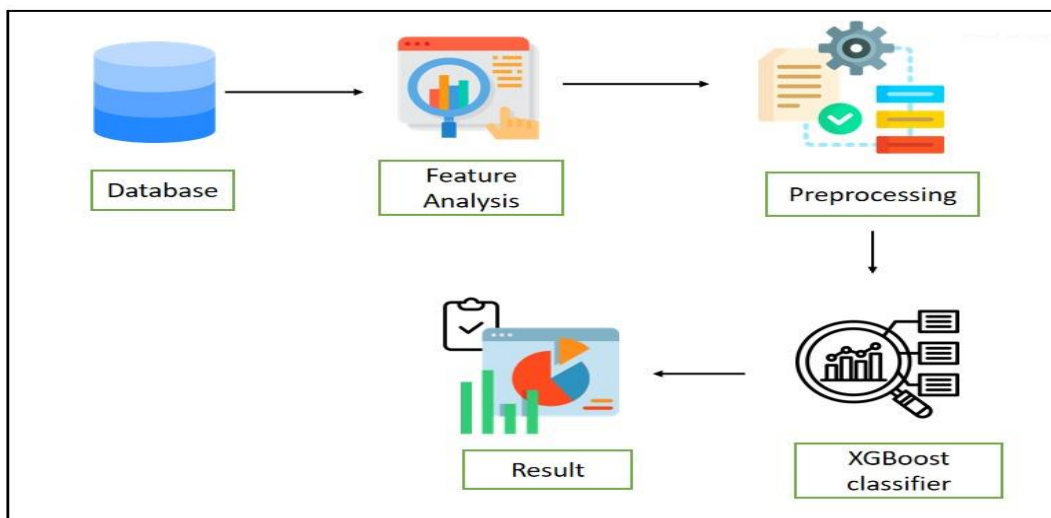


Fig.3: Architecture of System

5. ADVANTAGES OF THE SYSTEM

5.1 Handling Imbalanced Datasets: XGBoost excels in handling imbalanced datasets commonly encountered in intrusion detection. It ensures that rare intrusion types are not overlooked, thereby improving detection accuracy.

5.2 Exceptional Performance Across Domains: XGBoost has demonstrated exceptional performance in various domains, making it a promising choice for enhancing intrusion detection accuracy. Its adaptability to different contexts enhances its effectiveness.

5.3 Automation for Efficiency: The automated nature of intrusion detection using machine learning and XGBoost allows security experts to focus on strategic tasks, threat response, and policy improvements rather than manual data monitoring. This automation streamlines the process and reduces human workload.

5.4 Optimized for Speed and Efficiency: XGBoost is optimized for speed and efficiency, making it suitable for real-time or near real-time intrusion detection. It can train models relatively quickly, enabling rapid response to potential threats.

6. ALGORITHM OF PROPOSED SYSTEM

Step-1: Importing Libraries

Step-2: Data Loading

Step-3: Data Exploration

Step-4: Pie Plot Function

Step-5: Data Visualization



Step-6: Data Preparation

Step-7: Train-Test Split

Step-8: Label Mapping

Step-9: Model Training

Step-10: Model Evaluation

- Based on the test results, make forecasts.
- Compute and present the matrix of confusion.
- Plot the confusion matrix using ConfusionMatrixDisplay as show in the Fig.4

| | | Predicted | |
|--------|---------------|--------------------------------------|-------------------------------------|
| | | Negative (N) - | Positive (P) + |
| Actual | Negative - | True Negative (TN) | False Positive (FP) Type I Error |
| | Positive + | False Negative (FN) Type II Error | True Positive (TP) |

Fig.4: Confusion Matrix [12]

- A confusion matrix is a tabular representation that helps to evaluate performance of classification model.
- The terms "TP," "FN," "TN," and "P" are commonly used in the context of binary classification to represent different outcomes in a confusion matrix.
- True Positive (TP): Both the observation and the prediction are positive.
- False Negative (FN): A positive observation is anticipated to be a negative one.
- True Negative (TN): Both the observation and the prediction are negative.
- False Positive (FP): A positive prediction is made based on a negative observation.
- Calculate and display accuracy, precision, and recall scores.

Accuracy: This tells how many times a model made a correct prediction across the entire data set. Mathematically, Accuracy is ratio between number of correct predictions to total predictions.

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

Precision: It is about correctness of +ve predictions and is a measure of quality. It is the ratio of correctly predicted +ve cases to total number of +ve predictions. $\text{Precision} = \frac{TP}{TP+FP}$

Recall: Recall measures how well the model captures all the +ve cases and is a measure of quantity. It is ratio of correctly predicted +ve cases to total number of actual positives. $\text{Recall} = \frac{TP}{TP+FN}$

F1-score: Precision and recall are balanced in the F1-score. F1-score is useful when you want a single metric that consider both FP and FN. $\text{F1 Score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

7. RESULTS

7.1 Random Forest using KDD(Existing)

The Below Picture Fig.5 represents the Confusion Matrix of Random Forest Using KDD.

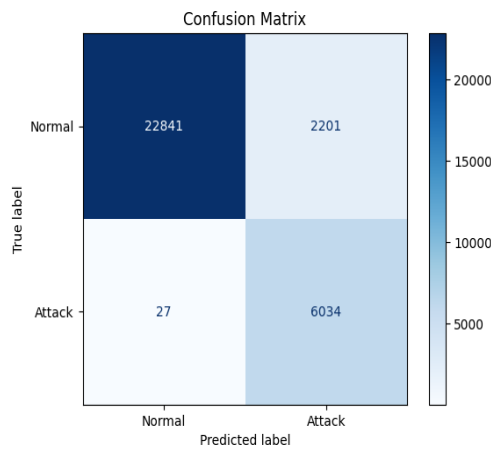


Fig.5: Confusion Matrix

Accuracy, Precision, Recall, F1-Score

```
#Calculating out the accuracy
Import accuracy, precision, recall, and f1 scores from sklearn.metrics
pos_label = 'attack.'
acc = accuracy_score(pred, labels_test)
precision = precision_score(pred, labels_test, pos_label=pos_label)
recall = recall_score(pred, labels_test, pos_label=pos_label)
f1 = f1_score(pred, labels_test, pos_label=pos_label)
print ("Accuracy is {}".format(round(acc,4)))
print("Precision is {:.4f}".format(precision))
print("Recall is {:.4f}".format(recall))
print("F1 Score is {:.4f}".format(f1))
Accuracy is 0.9284.
Precision is 0.9121
Recall is 0.9988
F1 Score is 0.9535
```

7.2 XGBoost Using KDD (comparison to random forest)

The below Picture Fig.6 represents the Confusion Matrix of XGBoost Algorithm Using KDD

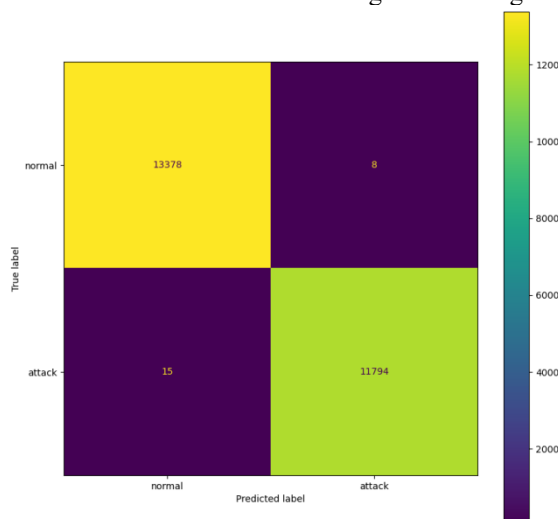


Fig.6: Confusion Matrix



Accuracy, Precision, Recall, F1-Score

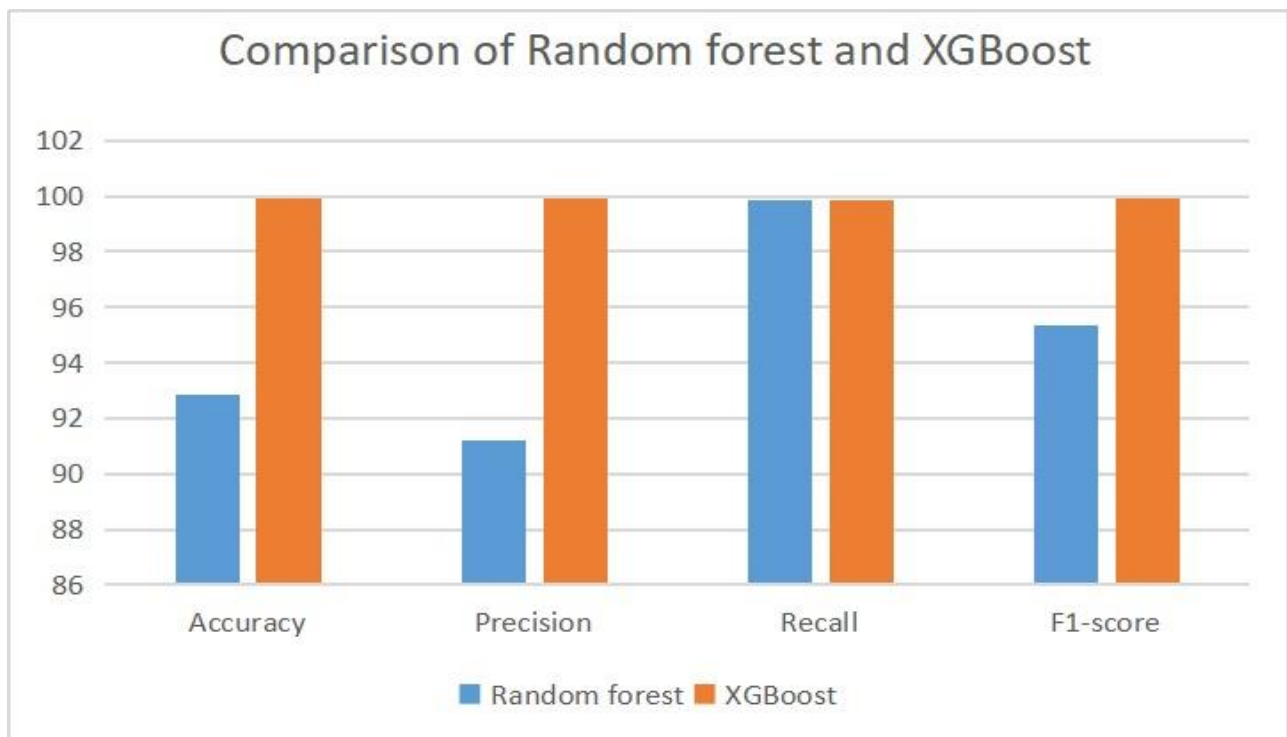
```
import xgboost as xgb
# Train the XGBoost classifier
xgb_classifier = xgb.XGBClassifier(objective='binary:logistic', n_estimators=100, random_state=42)
xgb_classifier.fit(x_train, y_train)
# Evaluate the XGBoost classifier
evaluate_classification(xgb_classifier, "XGBoost", x_train, x_test, y_train, y_test)
```

Training Accuracy XGBoost 99.98710023120356 Test Accuracy XGBoost 99.90871204604088
Training Precision XGBoost 99.98291474457544 Test Precision XGBoost 99.93221487883409
Training Recall XGBoost 99.98932103116123 Test Recall XGBoost 99.87297823693793
Training F1-score XGBoost 99.98611778525282 Test F1-score XGBoost 99.90258777688365

Comparison of Random Forest and XGBoost

| | Accuracy | Precision | Recall | F1-Score |
|---------------|----------|-----------|--------|----------|
| Random Forest | 92.84 | 91.21 | 99.88 | 95.35 |
| XGBoost | 99.9 | 99.93 | 99.87 | 99.9 |

Tabel.1: Comparison of Performance Metrics



8. CONCLUSION

we used the Random Forest algorithm with the KDD dataset to establish a baseline performance. Upon comparing the results, we found that the XGBoost algorithm outperformed Random Forest in terms of accuracy when applied to the same dataset. This demonstrates the effectiveness of XGBoost over Random Forest and its adaptability to newer datasets, making it a superior choice for our classification tasks.

9. REFERENCES

1. NARAYANA, Y. V., & SREEDEVI, M. (2023). Deep Neural System for Identifying Cybercrime Activities in Networks. *Journal of Theoretical and Applied Information Technology*, 101(16).
2. G. O. Boussi and H. Gupta, "A Proposed Framework for Controlling Cyber- Crime," 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 2020, pp. 1060-1063, doi: 10.1109/ICRITO48877.2020.9197975.



3. W. Khan, S. Zaib, F. Khan, I. Tarimer, J. T. Seo and J. Shin, "Analyzing and Evaluating Critical Cyber Security Challenges Faced by Vendor Organizations in Software Development: SLR Based Approach," in *IEEE Access*, vol. 10, pp. 65044-65054, 2022, doi: 10.1109/ACCESS.2022.3179822.
4. <https://www.wallarm.com/what/what-is-a-cyber-attack>
5. Kiran, S. W. Prakash, B. A. Kumar, Likhitha, T. Sameeratmaja and U. S. S. R. Charan, "Intrusion Detection System Using Machine Learning," 2023 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2023, pp. 1-4, doi: 10.1109/ICCCI56745.2023.10128363.
6. Mukesh Kumar Yadav, Mahaiyo Ningshen, 2023, Enhancement of Intrusion Detection System using Machine Learning, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 12, Issue 01 (January 2023)
7. S. Mirlekar and K. P. Kanojia, "A Comprehensive Study on Machine Learning Algorithms for Intrusion Detection System," 2022 10th International Conference on Emerging Trends in Engineering and Technology - Signal and Information Processing (ICETET-SIP-22), Nagpur, India, 2022, pp. 01-06, doi: 10.1109/ICETET-SIP-2254415.2022.9791586.
8. Dhaliwal, S., Nahid, A.-A., & Abbas, R. (2018). Effective Intrusion Detection System Using XGBoost. *Information*, 9(7), 149. <https://doi.org/10.3390/info90701>
9. Md. Alamgir Hossain, Md. Saiful Islam. "Ensuring network security with a robust intrusion detection system using ensemblebased machine learning", *Array*, 2023.
10. Chandrasekhar, A. M., & Raghuv eer, K. (2013, January). Intrusion detection technique by using k-means, fuzzy neural network and SVM classifiers. In 2013 International Conference on Computer Communication and Informatics (pp. 1-7). IEEE.
11. <https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30>
12. <https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5>