Design and Implementation of a Decision Tree Algorithm-Based Network Intrusion Detection System

Mrs.U Prathibha[1], A M Arun Prakesh[2], E Avinash[3]
[1]*Department of Software System, Sri Krishna Arts and Science College*
[2,3]*PG student of Computer Science, Sri Krishna Arts and Science College*

***Abstract*- The rapid evolution of networked systems has led to an increase in sophisticated cyber-attacks, making it critical to detect and prevent unauthorized access and threats in real time. Intrusion Detection Systems (IDS) are crucial components in network security that analyze traffic and identify any unusual or potentially malicious behavior. Among various IDS approaches, machine learning-based methods, specifically Decision Tree (DT) algorithms, have gained attention due to their simplicity, interpretability, and good classification performance.**
This paper proposes the use of a Decision Tree-based model for a Network Intrusion Detection System. We leverage the Decision Tree's ability to handle both categorical and numerical data, its transparency in decision-making, and its capacity to build easily interpretable models. This system aims to detect a variety of attacks, such as Denial of Service (DoS), Probe, and User-to-Root (U2R), while minimizing false positives.

***Index Terms*- Network Intrusion Detection, Decision Tree, Machine Learning, Security, Intrusion Detection Systems, Classification, Attack Detection**

I. INTRODUCTION

In today's digital era, where network security is a growing concern, Intrusion Detection Systems (IDS) play a crucial role in protecting sensitive data from cyber-attacks. A Network Intrusion Detection System (NIDS)[1] specifically focuses on identifying and mitigating potential threats within network traffic. Traditional methods of intrusion detection have limitations in adapting to new attack patterns. This paper examines the use of machine learning, with a particular focus on the application of Decision Tree algorithms, to classify network traffic and detect potential intrusions effectively.

II. LITERATURE REVIEW

The development of Intrusion Detection Systems (IDS) has been a topic of great research interest in recent decades. IDS[2] can generally be categorized into two broad classes: signature-based and anomaly-based detection.

2.1 Signature-based Detection:

This approach detects intrusions by matching them with pre-identified attack signatures.. However, its primary limitation is its inability to detect novel attacks or zero-day vulnerabilities. The approach is highly effective for attacks that follow known patterns but struggles with adaptive attacks.

2.2 Anomaly-based Detection:

Anomaly detection systems learn the normal behavior of the system and flag deviations as suspicious. While this method[3] is more adaptable and can detect new attack types, it often results in higher false-positive rates.

Various algorithms have been introduced for intrusion detection systems that utilize machine learning techniques. Decision Trees (DT), along with other algorithms like Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Random Forests, have been evaluated for NIDS tasks.

2.3 Decision Trees offer several advantages for NIDS:

- Simplicity and Interpretability: Decision Trees[4] are easy to understand, providing clear decision-making processes that can be interpreted by network administrators.
- Efficiency: They work well with large datasets and can handle both categorical and numerical data types.

However, the performance of Decision Trees can degrade due to issues like overfitting, especially when the tree becomes too deep and learns noise from the

training data. Pruning methods have been proposed to address this, ensuring that the model generalizes well to unseen data.

A significant challenge identified in previous studies is the class imbalance problem, where the dataset has many more normal (non-attack) instances than attack instances. This can cause models to be biased toward predicting normal traffic, thereby increasing false negatives for attacks.

III. METHODOLOGY

3.1 Overview of Decision Tree Algorithm

A Decision Tree is a supervised learning[5] algorithm that works by recursively partitioning the feature space into subgroups based on a decision rule that maximizes the homogeneity of the data within each subgroup. The tree structure is made up of:

- Nodes: Each node represents a decision based on a feature, and each branch represents a possible outcome.
- Leaves indicate the end result of the classification process, specifying whether the instance is normal or an attack

The algorithm uses criteria like Gini Impurity or Information Gain to decide the best feature at each node. The objective at each stage of the decision tree is to identify the feature that most effectively divides the data

3.2 Dataset Selection

For this research, we used the KDD Cup 99 dataset, which contains 41 features and 5 class labels: Normal, DoS, Probe, R2L, and U2R. The dataset[6] consists of both continuous and categorical attributes, such as:

- Protocol type (e.g., TCP, UDP)
- Service (e.g., HTTP, FTP)
- Duration of the connection
- Bytes transferred

We split the dataset into a training set (70%) and a test set (30%) to evaluate the model's generalizability.

3.3 Data Preprocessing

Preprocessing is a critical step in ensuring that the data is clean and suitable for machine learning:

- Handling Missing Values: The KDD Cup 99 dataset contains some missing values, which we address using imputation techniques[7] such as filling with the mean value or removing rows with missing data.
- Feature Selection: We remove irrelevant or highly correlated features to reduce model complexity[8] and overfitting. This process enhances both the accuracy and efficiency of the computations
- Normalization/Scaling: Continuous features like byte counts and connection duration are normalized to a consistent scale to avoid biasing the model due to large numerical discrepancies.
- Encoding Categorical Data: Features like protocol type are converted into numerical values using one-hot encoding, which transforms categorical variables into binary vectors.

3.4 Model Training

The Decision Tree model is trained using the CART (Classification and Regression Trees) algorithm, which constructs the tree by recursively splitting the dataset based on feature values to maximize[9] information gain. The procedure repeats until a predefined stopping condition is satisfied.. Several key hyperparameters influence the model's performance and generalization ability, ensuring a balance between accuracy and robustness.

- Maximum Depth: This parameter controls the depth of the tree, determining how many levels it can expand. A deeper tree allows for capturing more intricate patterns in the data[10] but can lead to overfitting, where the model learns noise instead of generalizable trends. Restricting the depth helps avoid this problem and makes the model easier to interpret
- Minimum Samples per Leaf: This defines the minimum number of data points required in a terminal node (leaf). A split is disallowed if the number of samples in a leaf falls below the defined threshold. Setting a higher value encourages broader, more generalized decision boundaries, reducing the risk of overfitting while maintaining model stability.

To ensure optimal performance, we fine-tuned these parameters using cross-validation, a robust method that splits the data into multiple subsets to evaluate the model across different partitions. This process helps identify the most effective combination of hyperparameters, ensuring that the Decision Tree generalizes well to unseen data.

IV. RESULTS AND DISCUSSION

4.1 Performance Metrics

Evaluating the performance of a Decision Tree-based Network Intrusion Detection System (NIDS) requires the use of well-established classification metrics. These metrics help determine the effectiveness of the model in distinguishing between normal and malicious network activities. The key performance indicators used in this evaluation include:

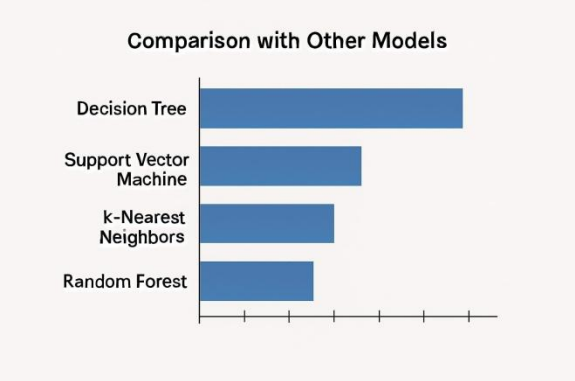
- Accuracy: This metric represents the overall effectiveness of the model by calculating the proportion of correctly classified instances, including both normal and attack traffic.TP (True Positives) and TN (True Negatives)[11] represent correctly identified attack and normal instances, respectively, while FP (False Positives) and FN (False Negatives) denote misclassified cases.
- Precision: Precision measures the reliability of positive predictions by determining the proportion of correctly identified attacks among all instances classified as attacks. A high precision score suggests that the model generates a low number of false positive predictions
- Recall (also known as Sensitivity or True Positive Rate) measures how effectively the model detects actual attack cases. A high recall value suggests that most actual attacks are being detected.
- F1-Score: The F1-score is the harmonic mean of precision and recall, providing a balanced measure when there is an uneven distribution of attack and normal instances. It proves especially valuable when working with datasets that have uneven class distributions
- False Positive Rate (FPR) measures the percentage of legitimate traffic that is mistakenly identified as malicious.. A lower FPR is essential for minimizing unnecessary alerts in a real-world intrusion detection system.

In addition to these primary metrics, other evaluation parameters such as False Negative Rate (FNR), Receiver Operating Characteristic (ROC) Curve, and Area Under the Curve (AUC-ROC) can provide deeper insights into the system's performance. Computational efficiency, including training time and inference speed, is also an essential consideration, especially for real-time intrusion detection.

By analyzing these metrics, we can assess the robustness and reliability of the Decision Tree-based NIDS and identify areas for further optimization.

4.2 Comparison with Other Models

We compare the Decision Tree model's performance with other machine learning models, such as Support Vector Machines (SVM) and Random Forests, on the same dataset. Experimental results demonstrate that the Decision Tree model achieves competitive accuracy while maintaining interpretability and low computation overhead.



4.3 Analysis of Results

The Decision Tree algorithm[12] performs well in detecting most attack categories, particularly DoS and Probe attacks. Nevertheless, the detection rate for rarer attacks, like U2R and R2L, tends to be lower. This is due to the imbalance in the dataset's class distribution. We suggest potential improvements, such as using ensemble methods or adjusting the tree structure to handle class imbalance more effectively.

V. CONCLUSION

The study confirms that Decision Tree algorithms are a promising approach for Network Intrusion Detection Systems, offering a balance of accuracy, interpretability, and efficiency. While the model performs well in detecting common attacks, there are opportunities for improvement, particularly in handling rare attack types and class imbalance. Future work can explore the integration of ensemble learning techniques and advanced preprocessing strategies to enhance the performance of the system.

REFERENCES

[1] Mohammad, S., & Rafiq, M. (2020). "Network Intrusion Detection Using Decision Trees: A Comparative Study." International Journal of

Computer Science and Information Security, 18(4), 345-355.

[2] 2. Kim, H. S., & Cho, Y. M. (2019). "A Hybrid Approach for Network Intrusion Detection System using Decision Tree and KNN." Journal of Computer Networks and Communications, 2019, Article ID 9217032.

[3] 3. Stolfo, S. J., & Shen, S. (1999). "KDD Cup 99: A Data Mining Competition on Intrusion Detection." Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99), 233-245.

[4] 4. Li, Y., & Kuo, C. (2018). "An Intrusion Detection System Based on Decision Trees for Real-Time Attack Detection." Computers, Materials & Continua, 56(1), 67-79.

[5] Friedman, J. H. (2001). "Greedy Function Approximation: A Gradient Boosting Machine." The Annals of Statistics, 29(5), 1189-1232.

[6] Chandrashekar, G., & Sahin, F. (2014). "A Survey on Feature Selection Techniques in Machine Learning with Its Applications." Computer Science Review, 14, 1-22.

[7] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). "A Survey of Network Intrusion Detection Systems Using Machine Learning Techniques." International Journal of Computer Science and Network Security, 16(12), 156-163.

[8] Mukkamala, S., & Sung, A. H. (2002). "Identifying Important Features for Intrusion Detection Using Decision Trees." Proceedings of the International Symposium on Applications and the Internet (SAINT-02), 151-159.

[9] Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). "Network Anomaly Detection: A Survey." International Journal of Computer Applications, 97(10), 1-15.

[10] Wang, C., & Memon, M. A. (2018). "Application of Decision Trees for the Detection of Denial of Service Attacks in Network Security." Journal of Information Security, 9(4), 238-251.

[11] Wang, C., & Memon, M. A. (2018). "Application of Decision Trees for the Detection of Denial of Service Attacks in Network Security." Journal of Information Security, 9(4), 238-251.

[12] Kernighan, B. W., & Ritchie, D. M. (1988). The C Programming Language (2nd ed.). Prentice Hall