IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 27, Issue 1, Ser. 3 (Jan. – Feb. 2025), PP 15-20
www.iosrjournals.org

Cyber Threat Detection And Response: Analyzing Network Intrusions Using Decision Tree, KNN, And Logistic Regression

Ankit Shukla
Virendra Kumar
Department Of Computer Science And Application, School Of Engineering And Technology, Sharda University, Greater Noida, UP, India

Abstract
Network attacks pose a major security threat in today's society. From small mobile devices to large cloud platforms, nearly all computing products we use daily are interconnected and vulnerable to network intrusions. As the number of network users rapidly grows, these intrusions have become more frequent, volatile, and sophisticated. Detecting these intrusions in real-time across such extensive networks is critical yet highly challenging. Consequently, machine learning-based Network Intrusion Detection (NID) systems, recognized for their intelligent capabilities, have gained significant attention in recent years. Unlike traditional signature-based methods, AI-driven approaches are more effective at identifying advanced and evolving network attack variants. However, despite achieving high detection rates, existing systems often face a trade-off with elevated false alarm rates, which can impact the overall effectiveness of the intrusion detection system. This paper proposes the most effective machine learning algorithm for NID, specifically Decision Tree, applied to a dataset to enhance detection efficiency.

Keywords: *Network security, intrusion detection system, classifiers, decision tree, machine learning.*

Date of Submission: 27-01-2025 Date of Acceptance: 07-02-2025

I. Introduction

With the rapid advancement of digital infrastructures, network security has become a critical concern. From personal mobile devices to extensive cloud platforms, almost every digital product integrated into daily life is connected to networks, making them susceptible to cyber threats. Network intrusions, in particular, have become an increasingly significant issue, with the frequency and complexity of attacks rising in parallel with the expansion of network usage. The dynamic nature of these intrusions highlights the necessity for robust and adaptable Network Intrusion Detection (NID) systems that can effectively identify and counteract threats in real time.

Traditional signature-based NID approaches, while effective against known attacks, often fall short in recognizing novel or evolving threats. This limitation has prompted the adoption of machine learning (ML) techniques, which provide a more flexible solution by detecting patterns and anomalies in network traffic indicative of intrusions. Machine learning-based NID systems have shown considerable potential in not only identifying known attack types but also in adapting to emerging attack patterns. However, challenges persist, as many current ML-driven NID systems achieve high detection rates but also exhibit elevated false positive rates, which can undermine the overall effectiveness of intrusion detection systems.

This study presents the Decision Tree algorithm as the most efficient method for detecting network intrusions. By analyzing its performance in comparison to other machine learning techniques, the findings underscore the Decision Tree's effectiveness in accurately identifying threats while minimizing false alarms.

II. Related Work

Network Intrusion Detection (NID) has evolved significantly from traditional methods to more sophisticated machine learning (ML) techniques, driven by the increasing complexity and frequency of cyber threats. Traditional signature-based NID systems primarily identify known attacks by matching observed patterns against a database of predefined signatures. While these systems are effective for recognized threats, they struggle with new or unknown attack types, which limits their utility in dynamic network environments.

To address the rising threat of unknown cyber-attacks, significant research has been devoted to the development of intrusion detection systems (IDS) that effectively identify and mitigate malware, vulnerabilities, and exploits. Since the early 2000s, various data mining (DM) and machine learning (ML) methods have been successfully applied to IDS, leading to the creation of algorithms tailored for enhancing detection performance. Specifically, DM techniques have been instrumental in uncovering trends and associations within extensive datasets, as shown in the works of Hodo et al. [1,2]. Artificial neural networks (ANNs) have also been widely applied in multiclass intrusion detection tasks, utilizing feed-forward neural networks trained via back-propagation to predict intrusions [3].

In addition to ANNs, other ML methods, such as Support Vector Machines (SVM) and Random Forests, have been utilized in IDS frameworks. Research by Roopadevi et al. demonstrates the advantages of SVM with feature reduction for intrusion detection [4], while Zhang and Zulkernine proposed a hybrid IDS model employing Random Forests for enhanced detection in network-based systems [5]. These studies indicate that certain classification algorithms excel in detecting specific types of attacks, though some are less effective in other scenarios. Multiclassifier models have been suggested to address these limitations, particularly with approaches like Random Forests in network intrusion detection systems (NIDS).

Moreover, Farid et al. investigated hybrid DM algorithms to boost the classification accuracy of Naïve Bayes in multiclass intrusion detection scenarios [6]. Building on this, Koc et al. introduced an IDS framework based on data mining, utilizing the KDD'99 dataset to demonstrate the effectiveness of the Hidden Naïve Bayes (HNB) model in addressing challenges such as correlated features and high data volumes within IDS applications [7].

III. Methodology

The following methodology outlines the process used to develop a Decision Tree classifier for detecting network intrusions. The approach combines data preprocessing, feature engineering, and model evaluation to achieve high accuracy and reliability.

Data Collection

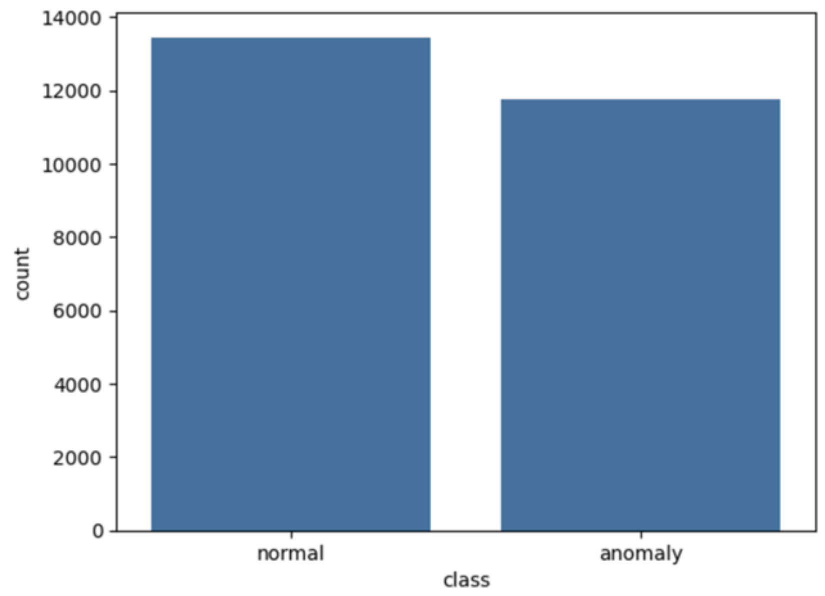
The dataset used in this study simulates a real-world military network environment with diverse intrusions, mimicking a US Air Force Local Area Network (LAN). It contains labeled records, where each connection is identified as either normal or as a specific type of attack (e.g., DoS, Probe, R2L, U2R).

Each record consists of **41 features** (38 quantitative and 3 qualitative) and a target label with two categories:

- **Normal**
- **Anomalous (Attack)**

Figure 1: Graph Represents Anomalous and Normal network connection

```
<Axes: xlabel='class', ylabel='count'>
```



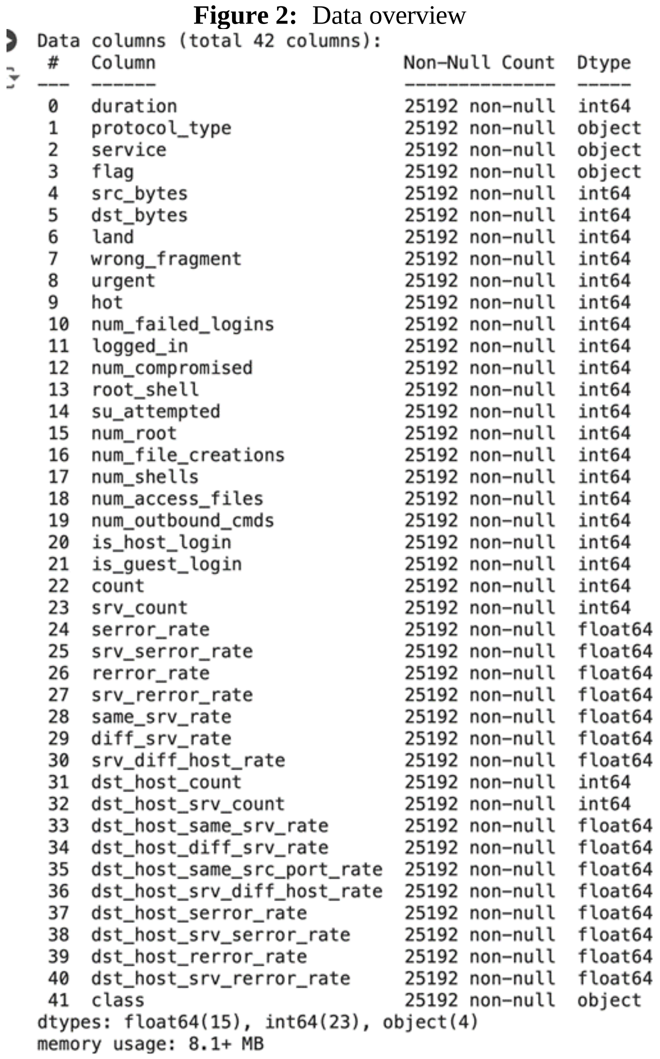
Data Preprocessing

Given the complexity of the dataset, preprocessing was critical. The preprocessing steps included:

- **Handling Missing Values**: The dataset contained non-null counts for each column, confirming that no data imputation was required.

- **Encoding Categorical Variables**: Three features, including protocol_type, service, and flag, were categorical. These were transformed using label encoding to prepare the data for the Decision Tree classifier.
- **Feature Scaling**: To optimize model performance, numerical features were scaled, ensuring that no feature disproportionately influenced the model's learning process.

Figure 2: Data overview

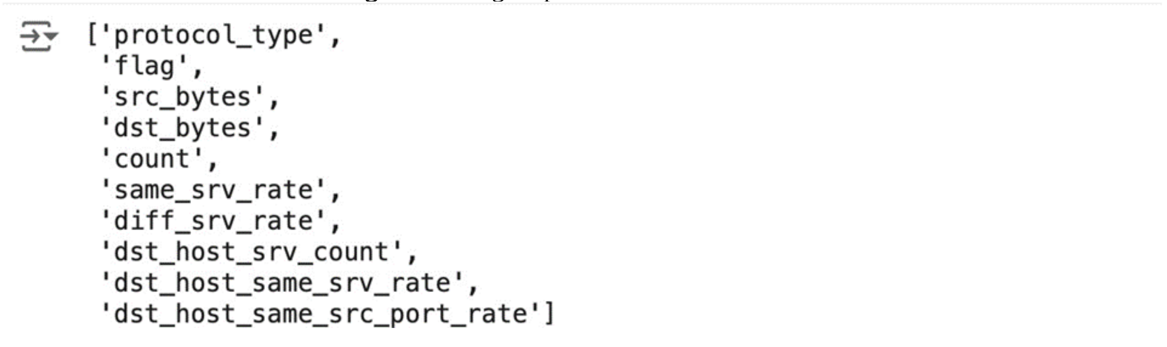


```
Data columns (total 42 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   duration                     25192 non-null  int64
 1   protocol_type                25192 non-null  object
 2   service                      25192 non-null  object
 3   flag                         25192 non-null  object
 4   src_bytes                    25192 non-null  int64
 5   dst_bytes                    25192 non-null  int64
 6   land                         25192 non-null  int64
 7   wrong_fragment               25192 non-null  int64
 8   urgent                       25192 non-null  int64
 9   hot                          25192 non-null  int64
 10  num_failed_logins            25192 non-null  int64
 11  logged_in                    25192 non-null  int64
 12  num_compromised              25192 non-null  int64
 13  root_shell                   25192 non-null  int64
 14  su_attempted                 25192 non-null  int64
 15  num_root                     25192 non-null  int64
 16  num_file_creations           25192 non-null  int64
 17  num_shells                   25192 non-null  int64
 18  num_access_files             25192 non-null  int64
 19  num_outbound_cmds            25192 non-null  int64
 20  is_host_login                25192 non-null  int64
 21  is_guest_login               25192 non-null  int64
 22  count                        25192 non-null  int64
 23  srv_count                    25192 non-null  int64
 24  serror_rate                  25192 non-null  float64
 25  srv_serror_rate              25192 non-null  float64
 26  rerror_rate                  25192 non-null  float64
 27  srv_rerror_rate              25192 non-null  float64
 28  same_srv_rate                25192 non-null  float64
 29  diff_srv_rate                25192 non-null  float64
 30  srv_diff_host_rate           25192 non-null  float64
 31  dst_host_count               25192 non-null  int64
 32  dst_host_srv_count           25192 non-null  int64
 33  dst_host_same_srv_rate       25192 non-null  float64
 34  dst_host_diff_srv_rate       25192 non-null  float64
 35  dst_host_same_src_port_rate  25192 non-null  float64
 36  dst_host_srv_diff_host_rate  25192 non-null  float64
 37  dst_host_serror_rate         25192 non-null  float64
 38  dst_host_srv_serror_rate     25192 non-null  float64
 39  dst_host_rerror_rate         25192 non-null  float64
 40  dst_host_srv_rerror_rate     25192 non-null  float64
 41  class                        25192 non-null  object
dtypes: float64(15), int64(23), object(4)
memory usage: 8.1+ MB
```

Feature Selection

Feature selection was conducted to reduce dimensionality and improve the Decision Tree's efficiency. Using correlation analysis and feature importance scores calculated by an initial Decision Tree model, irrelevant or redundant features were identified and excluded. This approach allowed for a more streamlined model that focused on high-impact features, such as src_bytes, dst_bytes, and other connection-related metrics.

Figure 3. Image represents Selected Features



```
['protocol_type',
 'flag',
 'src_bytes',
 'dst_bytes',
 'count',
 'same_srv_rate',
 'diff_srv_rate',
 'dst_host_srv_count',
 'dst_host_same_srv_rate',
 'dst_host_same_src_port_rate']
```

Decision Tree Classifier Setup

The Decision Tree algorithm was selected for its interpretability and ability to handle both categorical and numerical data efficiently. The Decision Tree was configured with the following parameters:

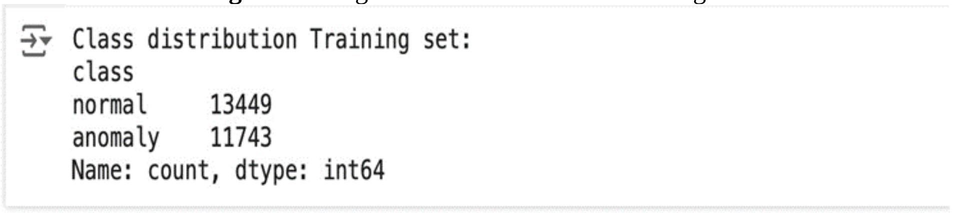
- Criterion: "Gini index" was used as the splitting criterion to measure node impurity.
- Max Depth: Depth of the tree was optimized based on validation results, balancing accuracy and overfitting.
- Min Samples Split: The minimum number of samples required to split an internal node was adjusted to control the tree's growth, helping to prevent overfitting.

These parameters allowed the model to create a robust structure that could identify patterns associated with different network intrusions effectively.

Model Training and Testing

Data Split: The dataset was divided into an 80:20 ratio for training and testing. The 80% training set was used to build and train the model, while the remaining 20% served as a test set to evaluate the model's performance on unseen data.

Figure 4. Image of Class Distribution Training Set

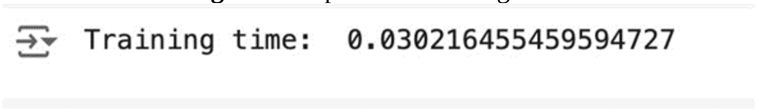


```
Class distribution Training set:
class
normal     13449
anomaly    11743
Name: count, dtype: int64
```

Training Process:

- During training, the Decision Tree classifier iteratively divided the dataset based on feature values, forming branches where nodes represented feature-based decisions and leaf nodes represented the final classification (normal or anomalous).
- The Gini index criterion was employed at each node to assess impurity, splitting nodes in a way that maximized the purity of each branch. This allowed the tree to efficiently classify patterns within the dataset.
- Hyperparameters such as maximum tree depth and minimum samples per split were adjusted through cross-validation, ensuring a balance between learning capacity and generalizability.

Figure 5. Represents Training Time

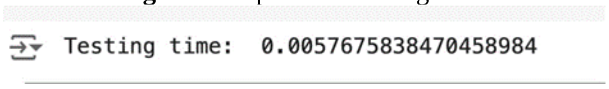


```
Training time:  0.030216455459594727
```

Testing Process:

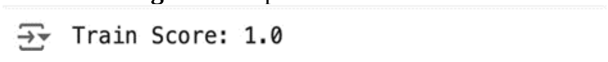
- Once the model was trained, it was evaluated on the test dataset. The Decision Tree used the learned structure to predict labels for the test data, classifying each connection as either normal or anomalous.
- Performance metrics such as accuracy, precision, recall, and F1-score were calculated to assess the model's effectiveness. These metrics helped determine not only how often the model correctly identified intrusions but also its ability to minimize false positives and false negatives.

Figure 6. Represents Testing Time



```
Testing time:  0.0057675838470458984
```

Figure 7. Represents Train Score



```
Train Score: 1.0
```

Evaluation Metrics and Analysis

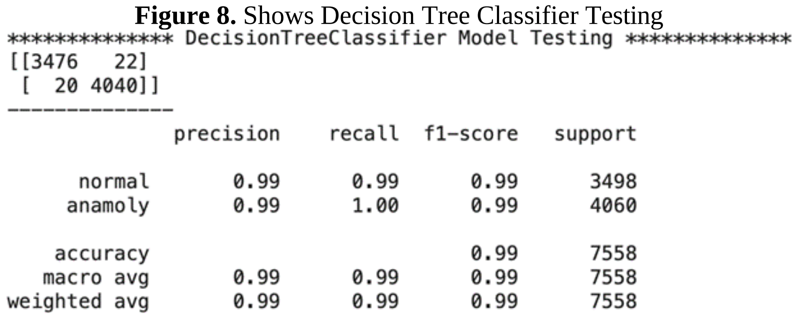
To thoroughly evaluate the Decision Tree model's performance, the following metrics were used:

- Accuracy: Defined as the ratio of correctly classified instances (both normal and anomalous) to the total number of instances, accuracy was the primary metric for evaluating overall performance.
- Precision and Recall: Precision (the ratio of true positives to the sum of true and false positives) and Recall

(the ratio of true positives to the sum of true positives and false negatives) provided insights into the model’s ability to correctly identify intrusions (attack class) while minimizing false alarms.

- F1-Score: This metric, the harmonic mean of precision and recall, balanced the two measures, providing a comprehensive evaluation of the model’s detection accuracy.
- Confusion Matrix: The confusion matrix illustrated the model's true positive, false positive, true negative, and false negative counts, offering a visual summary of classification results and identifying potential misclassification patterns.

Figure 8. Shows Decision Tree Classifier Testing

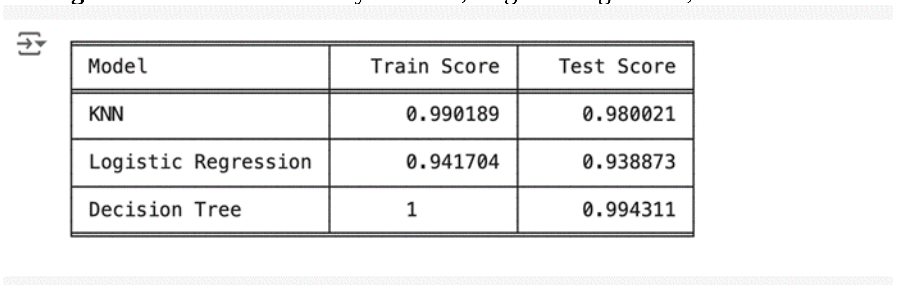


```
************** DecisionTreeClassifier Model Testing **************
[[3476   22]
 [  20 4040]]
--------------
              precision    recall  f1-score   support

      normal       0.99      0.99      0.99      3498
     anamoly       0.99      1.00      0.99      4060

    accuracy                           0.99      7558
   macro avg       0.99      0.99      0.99      7558
weighted avg       0.99      0.99      0.99      7558
```

IV. Result

The performance evaluation of the Decision Tree classifier was conducted using a dataset designed to simulate network intrusion scenarios, consisting of both normal and anomalous network traffic. The Decision Tree achieved an impressive accuracy score of **99.4%**, positioning it as the most effective model among those evaluated, which included K-Nearest Neighbors (KNN) and Logistic Regression, with accuracy scores of **98.0%** and **93.8%**, respectively.

Figure 9. Shows the accuracy of KNN, Logistic Regression, Decision Tree



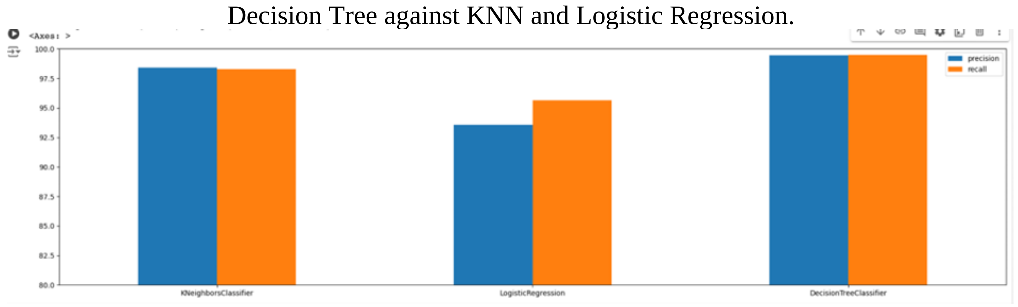
Model	Train Score	Test Score
KNN	0.990189	0.980021
Logistic Regression	0.941704	0.938873
Decision Tree	1	0.994311

A. Decision Tree Classifier Performance:

The Decision Tree model outperformed its counterparts, showcasing its ability to accurately classify network connections with minimal error. The high accuracy of **99.4%** reflects its capability to distinguish effectively between normal and anomalous traffic, resulting in a reliable intrusion detection system (IDS).

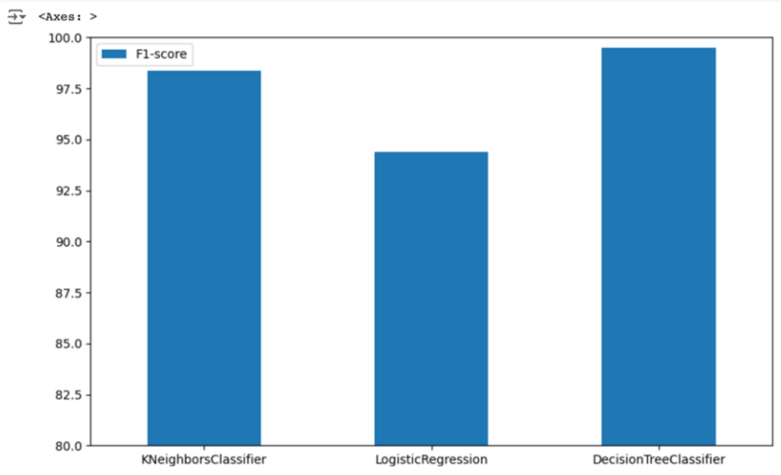
Precision Score and Recall: The graphical analysis of precision and recall demonstrated that the Decision Tree achieved the highest precision and recall scores among the models evaluated. This indicates that the model not only effectively identified intrusions (high recall) but also maintained a high standard in minimizing false positives (high precision). These attributes are crucial for practical IDS applications, where both sensitivity and specificity are required to ensure operational effectiveness.

Figure 10. The graph illustrating the precision and recall scores to visually compare the performance of the Decision Tree against KNN and Logistic Regression.



F1 Score: The Decision Tree also exhibited the highest F1 score, indicating a balanced trade-off between precision and recall. This reinforces the model's robustness in accurately detecting intrusions while minimizing false alarms. The F1 score is a vital metric in cybersecurity applications, as it highlights the model's effectiveness in real-world scenarios where both false positives and false negatives can have significant consequences.

Figure 11. The graph of the F1 scores represent that the superior performance of the Decision Tree in comparison to the KNN and Logistic Regression.



V. Conclusion

The Decision Tree classifier has proven to be a robust and efficient tool for network intrusion detection, achieving a remarkable accuracy rate of 99.4%. This high level of accuracy surpasses that of both Logistic Regression and K-Nearest Neighbors, reinforcing the Decision Tree's suitability for deployment in real-world intrusion detection systems.The performance metrics—precision, recall, and F1 score—further underscore the Decision Tree's capability to effectively identify intrusions while minimizing false positives and false negatives. The successful application of the Decision Tree on the simulated dataset demonstrates its potential for real-time intrusion detection in diverse network environments.

In conclusion, this research highlights the efficacy of the Decision Tree algorithm in enhancing cybersecurity measures through accurate and timely detection of network intrusions. Future research could explore hybrid models and the integration of additional features to further improve detection capabilities in complex and evolving threat landscapes.

This study lays the groundwork for continued advancements in intrusion detection systems, emphasizing the importance of choosing appropriate algorithms that balance accuracy and reliability in safeguarding network security.

References

[1] Hodo, E., Bellekens, X., Hamilton, A., Dubouilh, P., Iorkyase, E., Tachtatzis, C., & Atkinson, R. (2016). Threat Analysis Of Iot Networks Using Artificial Neural Network Intrusion Detection System. In Proceedings Of The IEEE International Symposium On Networks, Computers And Communications (ISNCC), Yasmine Hammamet, Tunisia.

[2] Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., & Atkinson, R. (2017). Shallow And Deep Networks Intrusion Detection System: A Taxonomy And Survey. Cornell University Library. Arxiv Preprint.

[3] Brifcani, A., & Issa, A. (2011). Intrusion Detection And Attack Classifier Based On Three Techniques: A Comparative Study. Engineering Technology Journal, 29(4), 368-412.

[4] Roopadevi, E., Bhuvaneswari, B., & Sahaana, B. (2016). Intrusion Detection Using Support Vector Machine With Feature Reduction Techniques. Indian Journal Of Science, 23(2), 148-156.

[5] Zhang, J., & Zulkernine, M. (2006). A Hybrid Network Intrusion Detection Technique Using Random Forests. In Proceedings Of The IEEE First International Conference On Availability Reliability And Security (ARES'06), Vienna, Austria.

[6] Farid, D. M., Zhang, L., Hossain, M. A., & Strachan, R. (2014). Hybrid Decision Tree And Naïve Bayes Classifiers For Multi-Class Classification Tasks. Expert Systems With Applications, 41(1), 1937-1946.

[7] Koc, L., Mazzuchi, T. A., & Sarkani, S. (2012). A Network Intrusion Detection System Based On A Hidden Naïve Bayes Multiclass Classifier. Expert Systems With Applications, 39(11), 13492-13500

[8] Jun GUO , Norikazu Takahashi, Wenxin Hu . An Efficient Algorithm For Multi-Class Support Vector Machines. IEEE- 2008.

[9] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G.Tedesco, And J. Twycross, "Immune System Approaches To Intrusion Detection - A Review," Natural Computing, Vol. 6, No. 4. Pp. 413–466, 2007.

[10] W. C. Lin, S. W. Ke, And C. F. Tsai, "CANN: An Intrusion Detection System Based On Combining Cluster Centers And Nearest Neighbors," Knowledge-Based Syst., Vol. 78, No. 1, Pp. 13–21, 2015.

[11] M. K. Siddiqui And S. Naahid, "Analysis Of KDD CUP 99 Dataset Using Clustering Based Data Mining," Int. J. Database Theory Appl., Vol. 6, No. 5, Pp. 23–34, 2013