AI-Assisted Custom Rule Generation for Signature-Based Network Intrusion Detection Systems

Micheal Opeyemi Durodola

Department of Computing, Dabble Lab, London, United Kingdom

DOI: https://doi.org/10.51584/IJRIAS.2025.100800154

Received: 22 July 2025; Accepted: 30 July 2025; Published: 29 September 2025

ABSTRACT

Signature-based Network Intrusion Detection System plays a critical role in the security infrastructure of a network domain. It function solely by using predefined rules to compare incoming network traffic against a database of attack signatures to determine if the network traffic is safe or malicious. Although, this method has proven to be effective with known attack signatures, network security expert still need to spend quality time to investigate activities in a particular network domain to create custom rules that will complement the predefined signatures. This dissertation investigates an AI-assisted approach to generate custom rules for NIDS. Leveraging machine learning to enhance adaptability, accuracy, and speed.

In this study, Random Forest Classifier which is a subset of a supervised machine learning model was trained using the NSL-KDD dataset for binary classification of network traffic and provide additional insight on the network traffic if it is malicious. The Random Forest Classifier was selected due to its robustness and efficiency in handling imbalance labelled dataset that are common in network traffic data.

The proposed system achieves an accuracy of 99.92% and precision of 99.90% showing its efficiency is classifying network traffic and reducing false positive. By focusing on custom rules generation, this research reveals the transformative effort go AI in developing proactive solution to organisation-specific security risks, offering a significant step forward in reinforcing the security stance of a NIDS.

INTRODUCTION

In recent times, threat to digital assets is fast evolving making network security a crucial area of focus for network experts. With the unprecedent increase in cyber threat and novel threat techniques, the demand for a robust defence mechanism needs to be intensified. If a web-service for example, is compromised, it can lead to a significant drawback in the organization revenue or reputation (Kim et al., 2020). Network Intrusion Detection System (NIDS) is critical in safeguarding network resources to threat to confidentiality, integrity and availability of data. Traditionally, NIDS is used to identify intrusion attack or violation of organisation security policies by inspecting network packets to detect attacks. Allowing network security expert to identify threat to the system before it compromises information availability, integrity and confidentiality (Mahbooba et al., 2021).

NIDS constantly monitors network traffic for any form of suspicious behaviour that violates network security policies and generate alert when an anomaly traffic or network usage behaviour is discovered. It can either be partially deployed by connecting it network switches using port mirroring technology or between network firewalls and switches, allowing network traffic pass through the NIDS (Z. Ahmad et al., 2021)

This study employs the NSL-KDD datasets, a refined version of the KDD'99 dataset – which is a better representation for real-time network traffic. Using the Random Forest Classifier Algorithm – a subset of ensemble machine learning algorithm – seek to bridge the gap between data analysis and practical implementation of machine learning model in an existing Network Intrusion Detection system. The integration leads to real-time analysis and automate the process of rules generation for the NIDS system. Be exploring this synergy between AI and NIDS technologies, this research seeks to contribute on how AI can proactively detect and response to network threat and as a result, create a more resilient network security system.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Aims

To develop and implement AI system that can analyse real time traffic from Intrusion Detection System and generate custom rules based on real-time network data. The system will dynamically create adaptive rules in response to observer pattern. As a result, reduce the need to manually update custom rules and enhancing the ability of the NIDS to detect and respond to revolving threats.

Background

Traditional signature-based NIDS relies on predefined rules to identify known attacks. This works by comparing network packets behaviour with known network data patterns that is present in the NIDS signature database to identify anomalies (Young et al., 2019) . This, however, rely on human input to constantly update the signature database to combat known threats and can be handicapped in case of unknown attacks (Khraisat et al., 2019). As a result, leaving the network system vulnerable to zero-day threats and novel malicious behaviours.

The integration of Artificial Intelligence (AI) in NIDS has emerged as a promising solution. Machine learning for instance, enable the analysis of complex network data patterns to detect anomalies in real-time as well as allowing the system to learn from past attack to adapt to new threats. Although integrate AI into NIDS has been found to be effective and is gradually gaining popularity. However, the prediction from this is still very hard to understand for network security expert and take significant time to interpret this information from this detection to rules that can help reinforced the security stance of the network (Mahbooba et al., 2021).

By automating the process of writing custom rules can significantly reduce the reliant of manual input, potentially improving response speed and detection accuracy. Machine Learning Ensemble learning technique such as Random Forest has shown promise for this application due to its robustness, interpretability and effectiveness in handling large and complex network traffic data. Through the implementation of machine learning, traditional NIDS can evolve from depending on static rule-based system into dynamic intelligence security solution that is able to generate rules that is able to identify malicious activities.

The finding from this study could have significant implications for organization to improve their security posture by enhancing the ability to detect cyber threat in real-time.

Challenges in modern NIDS

Modern NIDS has proven to be critical component of a robust network security infrastructure. Due to the evasive nature of the network cyber landscape, the NIDS system has been encumbered with numerous challenges such as volume of traffic, sophisticated attacks, encrypted networks, and decentralized network environment etc.

As many organizations begins to embrace big data architecture for their data pipeline, adoption for high-speed network traffic becomes imminent making it difficult for traditional to process this network data in real-time. Also, the necessity for encrypted network to establish safe communication between two entities further compound the challenges that traditional NIDS, making it difficult to inspect network packets (Papadogiannaki & Ioannidis, 2021).

Attackers are increasingly employing tactics such as obfuscation, encryption and machine learning model to adapt to NIDS defence, as a result, making it difficult to differentiate malicious activities from normal traffic. In addition, the rise in zero-day attack renders further strain of NIDS, as this attack target unknown weakness and renders traditional signature-based NIDS inadequate (Mbona & Eloff, 2022).

In addition, a decentralized network environment that includes different data point such as IoT ecosystems, hybrid clouds, edge computing, and such, introduces a unique challenge to the NIDS. IoT devices for example, operate with minimal security configuration, expand attack surface for threat actors. Similarly, network architecture such as hybrid cloud and edge computing setup can complicate a centralise monitoring and control making the NIDS generate high rate of false positive alert that can overwhelm network security or even leads to security team missing genuine threats (Bhavsar et al., 2023).

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

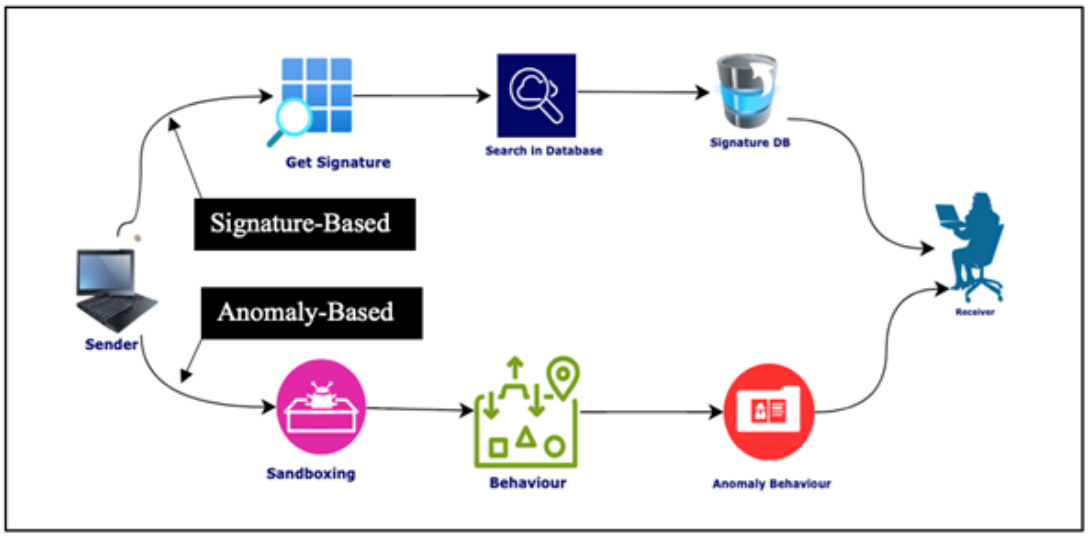

Figure 1 Traditional Network Intrusion Dtetection Systems

Traditional NIDS can be broadly classified as either anomaly based or signature-based NIDS. The signature-based based NIDS relies on its signature database that contains patterns identified with known threat. When there is a network traffic, the NIDS search its database to see if there is an identified pattern, if there is a match an alert is created for the network security administrator. On the other hand, anomaly-based NIDS uses information from known network traffic behaviour from an organization to set a baseline, and any traffic pattern that deviates from this s considered as a threat.

Research Questions

- How effective is an AI-based system in generating reliable custom rules for NIDS?
- How adaptable is AI-generated rules in addressing new and evolving network threats?
- How can AI minimise false positive without human intervention?

Objectives

- To develop and deploy an AI model that can analyse network traffic and generate custom rules with accuracy
- To identify suitable machine learning algorithm for real-time network analysis
- To test AI-generated NIDS rules with simulated datasets and assessing their performance.
- To measure the reduction in manual workload and efficiency gain by network security expert using AI-generated custom rules
- To document and analyse the impact of automated rule generation for NIDS on the overall response time to network threats

Structure of report

This dissertation is structured in a chronological manner to address the research problem and answer the research question in a logical and cohesive manner.

- **Chapter 1**: lays a foundation to the research topic, its context, aims and significance. Starts by introducing the reader to the topic, the central research problem, and clear objectives of the study. Finally, provided a summary of structure of dissertation that detail's the content and purpose of each chapter.
- **Chapter 2:** present a compressive analysis of available literature that is relevant to the topic, critical analysing the key theories of previous studies in other to establish the academic foundation of the study. Also, identify research gaps in the existing literature, emphasizing on the insufficiently explored area that this current study seeks to address.
- **Chapter 3:** details the research methodology, describing the design and method employed for data collection and processing. It further explains how the proposed methodology is justified for addressing

the search questions. Sampling techniques and theoretical concept of the design and evaluation method are further discussed.

- **Chapter 4:** present and analyse the collected data, offering result from analysis in form of tables and charts for clarity and ease of interpretation. The analysis in this chapter follows the methodology in previous chapter, focusing on addressing the research questions. The findings are further discussed in relation to the proposed research questions providing an insight on how the result align or deviate from the expectations set during the research design.
- **Chapter 5:** provides a comprehensive discussion of the result from the analysis in the previous chapter. Linking the result from the study to existing theories and studies. This chapter also acknowledge the limitation of the study, highlighting areas that warrant further research based on the findings.
- **Chapter 6:** concludes the dissertation by summarising the key finding and providing direct answers to the research questions. It reflects on the study's contributions to the field, highlighting its broader significance and impact. Finally, it present recommendation for future research, identifying area of further exploration.

LITERATURE REVIEW

In a robust network security infrastructure, NIDS is a vital component that is designed to monitor network traffic and triggers an alert in case of a malicious activity. Giving insights to the network analyst on how to response to a network incident correctly. With primary purpose of protecting the network from cyber threats such as data breaches, malware infections, denial-of-service (DoS), and unauthorised access; NIDS can achieve this by monitoring network traffic to observer patterns and behaviours over time or using widely recognise network attack behaviour to identify malicious activity or behaviours.

According to Ahmad et al. (2021), IDS is a software that serves as a security tool to monitor network traffic to ensure it does not violate the security policy of the organization. The figure below shows how NIDS is deployed passively, where it is connected to a network switch and configured with port mirroring technology, with the task of mirror network traffic to NIDS to detect intrusions. Unlike Network Intrusion Prevention System. (NIPS), which focus on blocking threats in real time, NIDS focuses on detection and alerting, giving security analyst insights on how to respond to incidents.

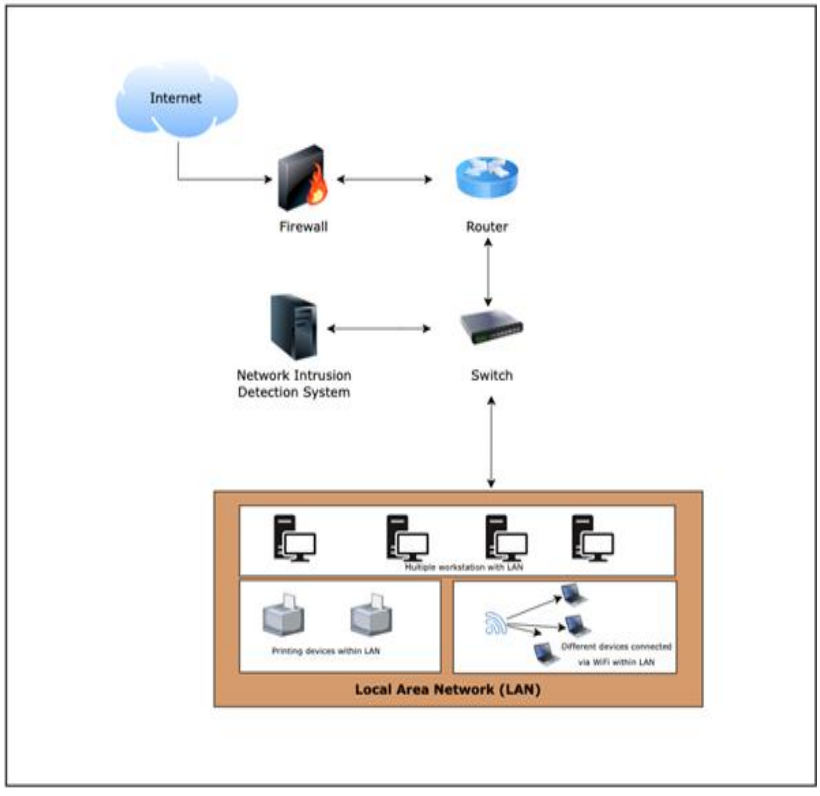

Figure 2 Network Intrusion Detection System deployment

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

NIDS has evolved over the years, it can be categorised into three primary generations, where each generation is characterised by different detection methodologies.

Signature-Based Detection (First Generation)

The initial generation of NIDS was introduced in the 1980s and early 1990s, it employed a signature-based approach. Where the system compares network traffic against predefined set signature and pattern of known attacks. It compares network traffic or system activities against database of predefined attack signature for a match to know malicious traffic. The strength of signature-based NIDS lies on its simplicity in detecting known strength. By maintaining a comprehensive database, it swiftly identifies attack pattern making a critical tool in an organization network security infrastructure. However, this approach is limited in identifying known threats and requires constant update of signature database to stay relevant against emerging threats.

In signature-based NIDS, the process usually begins with real-time monitoring, where the NIDS do a continuous scan of network packets, system call logs, or application events. With the help of matching pattern algorithm, it can compare observed patterns with stored attack signatures. When a match is found, the NIDS triggers an alert to identify to potential attack (Thankappan et al., 2024). This mechanism is highly effective for known attacks, if the signature database is updated, the NIDS just need to match network pattern with predefined signature. However, the method is not infallible. Its operational dependence on predefined rule makes it susceptible to novel or zero-day attacks, for which signature has not yet been created.

The primary challenges with signature-based NIDS are its reliance on signature of known attack to detect intrusion. This limitation necessitates the need for regular updates of the signature database to keep dating with evolving threats in the cyber space. This implies that attacks can remain unnoticed until the attack pattern is added to the NIDS signature database. In other to address limitations, signature-based NIDS are often used in conjunction with anomaly detection system, which provides a complementary task to cover its weakness.

The study by Alonso et al., (2022) explore the performance of signature-based IDS in the context web attacks. The study focuses on detection rate obtained from a predefined subset of roles for common signature-based IDS such as Snot, Nemesida and ModSecurity. In addition, the precision rate of the alert generated by each detection was monitor in real-life cases. The result from this study reveals maximum detection rate from the test is insufficient to protect the system effectively. The result further reveals that false rate alarm is highly dependent on the choice of the predefined rules select in the signature database. As a result, it is necessary to reflect on the default configuration of open-source signature-based IDS in the context of web attacks.

Jin et al., (J2021) studied how in-vehicle CAN (Control Area Network) bus network is prone to network attack because of the absent of security protection measure. However, because of the computational resource needed contemporary IDS cannot be implemented. In other to combat this, the study proposed a lightweight signature-based IDS. Anomalies caused by several attack modes own CAN buses was detected from real-world scenarios and provided the basis of selecting signatures. The result from the study reveals that the proposed IDS model can effectively detect and replay attacks.

Anomaly-Based Detection (Second Generation)

The anomaly-based detection is a type of NIDS that detect threats by identifying deviation from established patterns of normal behaviour within a network or system. Unlike signature-based NIDS, which relies of predefined attack signatures, anomaly-based detection operates on the principle of detecting anything that signifies deviation from the baseline of normal traffic activities. This method proves to be very valuable in identifying zero-day attack as well as novel attacks. The essence of anomaly-based detection NIDS is to adapt and learn from baseline behaviour of the network system, making it a powerful tool in identifying a novel threat. However, this adaptability means that anomaly detection system requires sufficient time and data to establish accurate baseline to identify anomalous behaviour.

The process by creating a baseline that representing normal network traffic behaviour. The baseline is constructed with historical data that includes traffic patterns, user activities and system resource utilisation.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Learning from this data, the system can set a baseline and detect anomalous behaviour that deviates from the baseline pattern. There are various establish techniques that can be used detect anomalies such as statistical analysis, machine algorithm, or an advance data mining method. For example, a statistical model can be developed that recognised behaviour between expected ranges for traffic volume, login attempts, or sudden spike in docile data transfer, that deviates from baseline traffic and point to a potential network threat.

Jabez & Muthukumar (2015), proposed a novel approach called Oilier Detection to detect malicious activities in computer network. This approach involves training a machine model with dataset with distributed storage - to increase performance and tested with data from received from real world. In the proposed approach, anomaly was measured using the Neighbourhood Outlier Factor (NOF). The result show exceptional performance in detecting anomaly activities in computer network compared to traditional intrusion detection systems.

Anomaly-based detection shines bright in discovering novel threats compared to signature-based mode, and its particularly useful in environment where attack is constantly evolving, and the detection of insider threats that is not captured in signature-based NIDS. Nevertheless, this approach is not void with its own challenges, one of the primary challenges is the high rate of false positive. Anomalies can be triggered by legitimate but common in activities which can leads high volume of alert that requires investigation. For example, when a new application is installed that cause a deviation in the baseline network pattern.

Furthermore, building accurate baseline can be time-consuming, especially in a dynamic environment with fluctuating network patterns. Also, the needed for a well-defined baseline can leads to trade-offs that can miss sophisticated attack patterns. To address these challenges most anomaly-based NIDS are used in conjunction with signature-based NIDS to enhance accuracy and reliability.

Machine Learning and Network Intrusion Detection Systems (Third Generation)

Integrating Artificial Intelligence (AI) and Machine Learning (ML) into Network Intrusion Detection System (NIDS) represent a paradigm shift in how threats is monitored in cybersecurity. The traditional NIDS relies on signatures-based methods to identify known threats. It however, struggle to identify emerging or novel threats. Incorporating ML and AI has enabled NIDS to evolve into dynamic and adaptive system capable of identifying complex threats, reduce the dependence on manual intervention and improving the overall detection accuracy.

Machine Learning algorithm can be broadly categorised to supervises, unsupervised and semi-supervised learning. In our research we favour supervised learning which involves training a model on labelled datasets, where the outcomes are already known. The supervised learning is very effective on labelled datasets to build a model that can detect normal traffic from abnormal traffic.

The common task in supervised learning is classification that separates the data. For example, predicting class label or sentiment analysis for a piece of text (Sarker, 2021).

In this complex system, machine learning empowers NIDS to learn from historical data to improve its performance over time. By analysing this historical network traffic and user behaviour to develop model can accurately predict normal behaviour and flag deviation that might indicate malicious activity, the key aspect includes supervised learning, unsupervised and semi-supervised learning approach.

Supervised Learning

Supervised learning algorithm relies on large, labelled datasets that classify network activity as either normal or malicious. Popular supervised learning algorithm such as Random Forest Classifier, Neural Networks, and Support Vector Machines (SVM); are trained on these large datasets to be able to classify different kind of intrusions. This approach fare better in an environment where network patterns are logged correctly to accurately identify down attack types or patterns that deviates from normal network traffic.

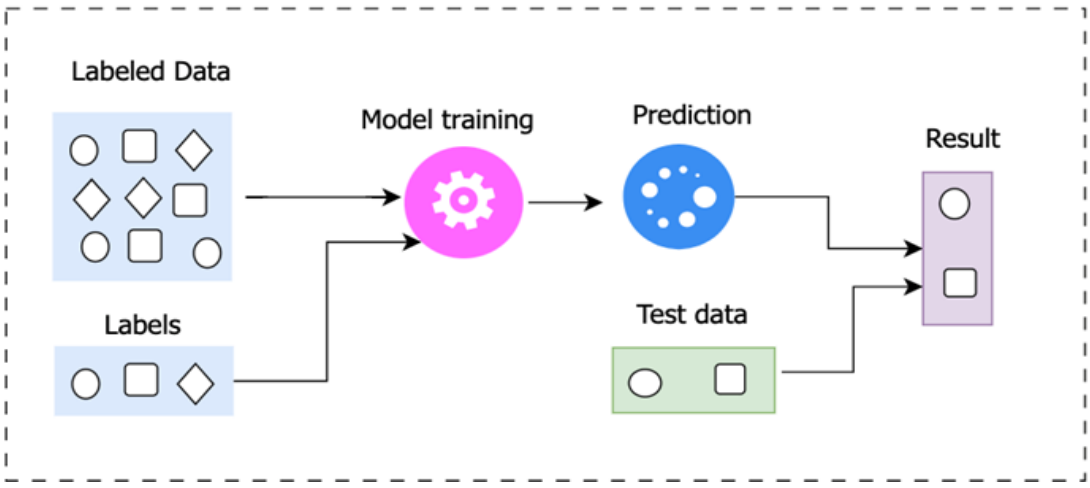

Figure 3 Supervised machine learning model

Taher et al., (2019) proposed a supervised learning approach using train dataset with Weka software. Where it used 20% NSL-KDD dataset as training. To train the model, the SVM and ANN learning algorithm for each type of feature selection with a 94.02% detection rate.

M. Ahmad et al., (2021) work proposed a model framework for defence of IoT-based network from cyber-attacks using NIDS. The proposed model uses popular supervised machine learning technique such as Random Forest, Supervised Vector Learning and Random Neural Networks uses a set of large datasets. In their study, traffic network flow from MQTT and TCP was also explored, and cluster were selected after adequate measure was used to remove overfitting. The result from this research achieves 96.96% and 97.37% in binary and multi-class classification respectively.

Unsupervised Learning

Unsupervised learning algorithm does not depend on labelled datasets, rather than requiring labelled examples. Algorithm such as k-means, Principal Component Analysis (PCA), and Autoencoders can be used cluster similar activities to highlight outliers. This algorithm is quite effective for large datasets; however, it falters for zero-day attack detection where the is no predefined signature in the database to be able to classify novel attacks.

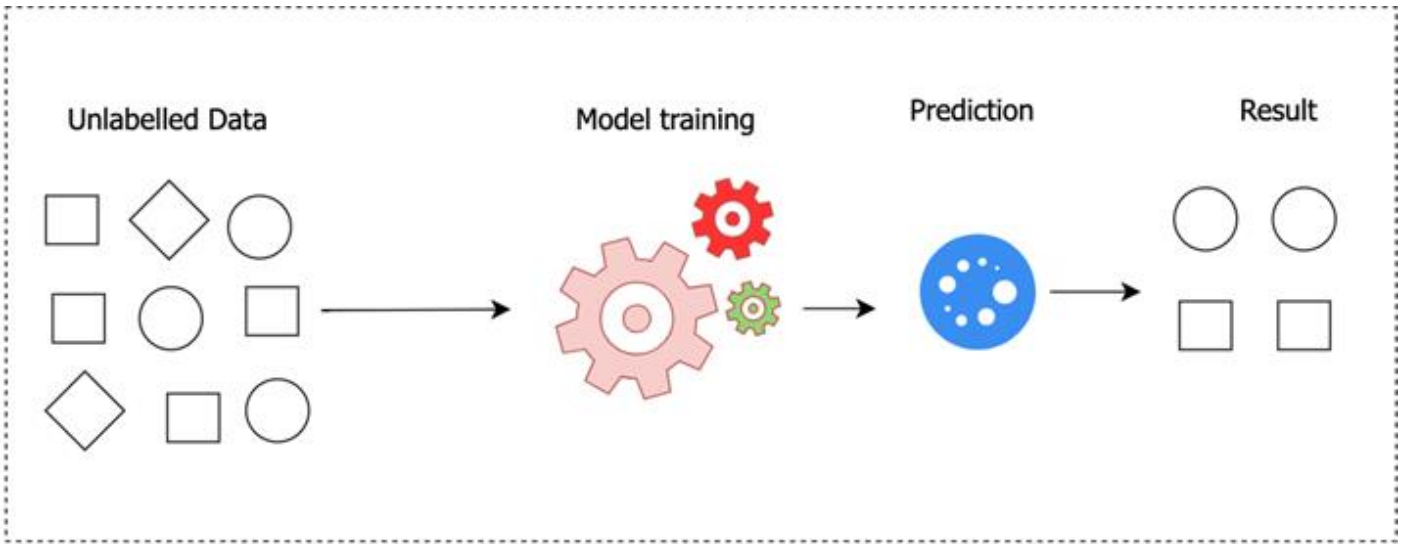

Figure 4 Unsupervised machine learning model

(Choi et al., 2019) proposed NIDS based on unsupervised learning algorithm using autoencoder and verify its performance. It also shows a practical guideline on for developing Network Intrusion Detection System based on auto encoders which contributes to exploration in the unsupervised learning subject in NIDS. In the study,

training data that reflects real network and varies in degree of abnormality was used. The result shows a 91.7% accuracy, which outperforms benchmark performance from using cluster analysis algorithms.

The study by (Verkerken et al., 2022) started by evaluating four unsupervised algorithms on two realistic and recent datasets. Afterwards, the results served as a baseline that was used to generalise the strength of the models. The result of the study reveals high classification score on the individual datasets. Averagely, the SVM yielded the highest AUROC scores of 0.9705 on the CIC-IDS-2017. This proves the capability of the model to detect novel attacks such as zero-day attack. The study also reveals there is a significant drop in classification performance, all four algorithms are seen to perform better than a completely supervised learning such as a random classifier.

Semi-Supervised Learning

Semi-supervised learning combines the strength of unsupervised learning and supervised learning method. This method leverages a small amount labelled data alongside a pool of large unlabelled datasets.

Due to the effectiveness of deep learning algorithm in processing large amount of data and deriving meaning from such complex data many studies have investigated the application of this type of algorithm in threat discovery framework such as intrusion detection systems. In this section we would review some of the studies.

Paul et al. (P2021), studies how to detect zero-day intrusion attack and proposed a hybrid unsupervised anomaly-based NIDS, which is a combination of one class support vector machine model (OCSVM) and subspace clustering (SCC) to evaluate the NSL-KDD network intrusion dataset. The OCSVM was employed to train the model using only one class, while the SCC models used to train the model on group unlabelled datasets. The hybrid model was seen to achieve a very exceptional result of 99% detection rate. Although, there was no sufficient information significant features that was used to achieve these results.

Hara & Shinomoto (2020) proposed an IDS that employs the use of a semi-supervised learning model. Compared to supervised training model that requires large datasets and quality time to label the datasets, semi-supervised model uses a small number of unlabelled datasets to reduce human labour tasks and improves its performance with unlabelled datasets. The proposed method incorporates Generative Adversarial Networks (GAN) into the Auto-encoder (AE). Evaluating the model with 0.1 percent NSL-KDD dataset reveals it had comparable result with IDS that uses machine learning method.

Zavrak et al. (2020) implemented autoencoder that was based on semi-supervised machine learning method on detecting zero-day attacks. The CICIDS2017 datasets which contains 11 benign and 11 malicious attack was employed. The variation auto encoder (VAE), auto encoder (AE), as well as the OCSVM proves to be very effectively, achieving area under the curve (AUC) of 76%,74% and 66% respectively. In detecting specific attack types, the model even achieves a better result. For example, OCSVM achieved AUC of 98.5% for heartbleed attacks.

Qureshi et al. (Qureshi et al., 2019) demonstrate how swarm intelligence and Random Neutral Network can be used to effectively classify cyber threats with an accuracy of 91.65%. A novel Artificial Bee Colony (ABC) algorithm was proposed based on Random Neutral Network (RNN) intrusion detection system. The result shows that RNN-ABC trained infusion detection system outperformed traditional neural network is detecting anomalies in network traffic.

Kumar et al. (Kumar et al., 2019) proposed the use of AI algorithm in protecting networks from known and future cyber threats. This is done by monitoring network traffic to detect anomalies detect malicious traffic. Port scanning attack was investigated and present an algorithm was presented that can learn from previous data and update the conventional rules of intrusion detection systems. In this paper, investigation of port scanning attack is carried out to infer the threat intelligence for the port scan and present an Artificial Intelligence managed NIDS system can learn from previous data

Faisal & Islam, (2023) study combines machine learning and deep learning algorithm to optimize Intrusion Detection Systems. In their study, NSL-KDDD datasets that are widely recognized as a true interpretation of

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

intrusion detection were analysed using multiple machine learning and deep learning model. Also, ensemble learning was studied to evaluate accuracy in threat detection. Their selected model proved to be adaptable to cyber threats. Consequently, opening a validating the efficiency of machine learning algorithm in monitoring cyber threats.

Verma et al.(2023) argues that modern intrusion detection systems are not sufficient in combating modern cyber threats in the cloud environment because they are trained with a single dataset. Further suggest that to combat cloud danger that exists in recent time proposed a Network Intrusion Detection System with intelligence that is orchestrated in a Kubernetes cluster that is hosted on server instance on cloud service provider to combat cloud-based network threats.

Latha & Thangaraj (2023), research introduced an all-inclusive intrusion detection system that can swiftly detect misuse and anomaly in computer networks. Using the Self-Organizing Map (SOM) for anomaly detection, and the Gradient Boosting Algorithm coupled with Ada Boosting Algorithm to detect abuse. The system shows great accuracy in detecting known and unknown threats. After testing the ensemble learning algorithm, the result shows the system ability to swiftly warn cybersecurity of imminent threats. Further, the focus on clear explanation and interpretation could be on future intrusion detection systems according to the authors.

Rule-Based NIDS and Manual Challenges

A rule-based Network Intrusion Detection System analyse network traffic by comparing it with a set of predefined rules to identify abnormal traffic that can pose security threat to the system. NIDS learns from predefined rules that can be periodically updated with custom rules to accommodate dynamic system behaviour (Liu et al., 2021). These rules are based on known traffic signature or patterns that can be highly effective in detecting familiar malicious behaviour and become obsolete with times as attack to network system becomes more sophisticated (Alcantara et al., 2022).

The main concern of rule- based network intrusion detection systems has always been false alarm, accuracy and efficiency. Because signature intrusion detection system is rule-based and depend on predefined rule to identify malicious activities; the default configuration usually comes with huge detection as well as high false alarm that can be detrimental in a production environment. As a result, expert needs to manually tuned the rule sets to reduce false alarm that is particular to a production environment (Alonso et al., 2022).

Although, NIDS can be effective in identifying known threat. However, they are encumbered with significant challenges particularly in terms of manual maintenance. Crafting and updating predefined rules can be time-consuming require expert knowledge in the domain. Security analysts require a high learning curve to be able to effectively create rules that can detect network threat and reduce false positive and false negatives.

Also, the need for precision in manual rule creation can be a challenge as rule that appears to be too strict can significantly increase the number of false positive, flagging legitimate network traffic as threats. In contrast, when rules appear too broad, it can miss genuine threats to the network infrastructure. The overhead here requires to constantly fine-tine and balance these rules which can be taxing to the security teams.

There has been much research conducted on the application of artificial intelligence to train Network Intrusion Detection Systems. Taking the baton from cybersecurity analyst, giving them little or no room to be part of the process. This research is focused on the ai-assisted approach in writing custom rules for signature-based NIDS.

Importance custom rules generation in modern NIDS

In a dynamic network environment, custom rules have emerged has a pivotal tool to complement predefined rule in a signature-based NIDS, thereby enhancing its threat detection capabilities. Traditional signature-based NIDS depends on static and predefined rules. As a result, struggle to keep pace with proliferation of modern cyber threats. This limitation underscores the needs for an adaptive and intelligent mechanism that can aid custom rule generation that is tailored specific for a unique network environment.

Tailored threat detection

Custom rules in signature intrusion detection system offers opportunity to define security rule that align to a unique network environments, operational efficiencies and specific threat profiles. Compared to generic rules, that can be too broad in certain context, custom rules focus on specific attack patterns and network behaviours that is peculiar to an organisational infrastructure. Having custom rules that gives a security team a granular control help security team to enhance accuracy in NIDS, reduce false positive and help to address security issues more efficiently.

Adaptive to emerging threats

As cyber threat landscape becomes more precarious with attacker using advance tactics such as polymorphic malware, zero-day exploit, and such, which is very difficult for modern NIDS to discover, custom rule generation approach, especially when integrated with machine learning, allows NIDS to generate custom rules on the fly in response to threats. AI-powered NIDS can gather knowledge from vast amount of network traffic data to identity previous unknown pattern. Afterwards, create actionable custom rules to combat these threats. Such adaptability ensure that NIDS can remain effective when expose to novel threats that are susceptible to traditional NIDS.

Automation and expert insights gap

One of the profound advantages of ai-assisted custom rule generation is its ability to bridge the gap between human expertise and automation. When a machine learning model can suggest rules based on data-driven insights, security expert c review these rules and ensure their adaptability and precision. These collaboration between security expert and automation process minimises the risk or underfitting or overfitting in custom rules generation and provides the opportunity to integrate domain knowledge to automated system. For example, a rule can flag a network activity as suspicious, but an expert can adjust the threshold to make these activities normal - based on the requirement of a unique network environment.

Reduce false positive and improve efficiency

False positives are a significant challenge in NIDS, because of the alert system that is peculiar to NIDS, false positive can overwhelm security team, creating nuisance by unnecessary alerts that diverts attention of security team, and eventually lead to missing of potential threats. Custom rules when carefully crafted, can help reduce noise by focusing only potential threats and excluding activities that are deem benign in the network environment. This improvement can help save time and resources, as well as strengthen the overall response strategy of the NIDS.

Supporting compliance and operational goals

Modern organizations are often challenged with regulatory and compliance requirement that mandate specific security measures. Most times these measures may not be covered in default signature database of network intrusion detection systems. The ability to generate custom rules that is specific to an organization, gives the security team to control to tailor rules that meets these obligations. Ensuring that monitoring and detection is compliance with industry standards and legal obligations. In addition, custom rules can strengthen operation performance by protecting critical assets, optimise the allocation of resources, and aligning secret practices to business operations.

RESEARCH METHODOLOGIES

This chapters describe th systemtaitc approach taken to develop, train and deploy an AI-assisted custom rules geenration for Network Intusion Detection System. Teh AI model selected for this purpose of the Randfom Forest, a subset of ensemble machine leaerning algorithm. The system is designed to analyze network traffic using pre-trained model that collects data in JSON format from an API endpoint. The response from this endpoint is a tailored security rules that can be directly applied to strength the stance of an IDS in real-time.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Research Design

The research methodology adopted for this dissertation is a quantitive experiemntal apporach, which is suitable for analysis of machinelearning models in a controlled environment. This method ensure that result is measurable and reproducible. In addition, it provides insights into how effecitve AI- assisted custom rules can enhance the capabilities of Network Intrusion Detection System (NIDS).

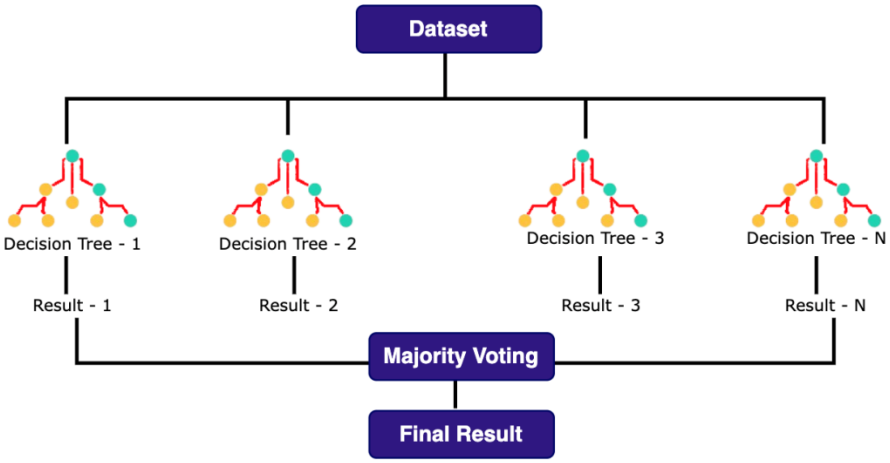

Figure 5 Random forest classifier model flow

The research design concept entails the development, training and evaluation of Random Forest classifier – a subset of ensemble machine learning model that is fame for its accuracy and capability in handling complex data structure with high dimensaionality. The choice of Random Forest model is justified by its inherent ability to combine with multiple decision tress which results to improved predcition of anomalies in a network traffic.

Data Source

For this project, NSL-KDD dataset was chosen. An enhancement of the original KDD'99 dataset which has been used wodely for research on intrusion detection system. The KDD'99 while effective historically was found to have some inherent limitation which includes: excessive redundancy and deuplicate records which can lead to overfitiing and biased perfromancen evaluation. The NSL-KDD dataset, however, was developed to address this limitations by offering cleaner and more balanced dataset that suited to for training a reliable intrusion detection model.

The NSL-KDD dataset is made up of diverse network traffic records, each been represented by 41 features that describe various attributes of the network connection such as protocol type, service flag, duration, etc. The features outlined in the dataset is designed to capture behaviour and charactertisitc of bothe benign and malicious network activities. In addition, the dataset provides label for each record, specifying whether the connection is a normal network activity or belong to several other attack types such as Dos (Denial of Service), R2L (Remote to Local), U2R (User to Root) and probing attacks.

For the purpose of this research the NSL-KDD dataset was adapted to facilittae binary classification, which simplkifies the task of differentiating a normal network traffioc from an abnormal traffic. This binary approach is critical in creating an Intrusion Detection model that can flag potential threat in rela-time.

Some of the key benefits the NSL-KDD datasets includes:

Reduction in Duplicate Records

Compared with the original KDD'99 datasets. The NSL-KDD datasets significantly reduce duplacte records in both training and testing. As a result preventing skewed evaluation and ensuring that the model learns from a diverse set of example.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Balanced Data Distribution

According to Ahmad et al., (2021), most proposed IDS methodology exhibits lower detection accuracy for certain attack compared to the overall threat detection of the model due to imbalance nature of the dataset. The NSL-KDD datasets promoted balance learning across various network traffic patterns. As a result, help with model become more balanced when analysing the various network traffic patterns.

Comprehenseive Feature Set

The NKSL-KDD dataset has 41 features that covers various network traffic attributes that covers connection details, timestamps, and contentn based details. This comprehensive features creates a solid foundation for training machine learning models. The label data in NSL-KDD dataset makes it more suitable for supervised learning approach that Random Forest Classifier proviedes. For instances, the presence of labelled datasets allows for a more srighta-forward approach in testing and enable model to learn distinguishly between normal traffic and abnormal traffic. In addition, labelled datasets allows for effewtive tracking of performance metrics such as F1-score, accuracy and precision recall.

In conclusion, the NSL-KDD datasets was selcted to due to its improvement and reliability when compared to its predecessor – KDD'99 dataset. Making it a preferrred chopice in developing and testing Ai- assisted rules generation for Network Intrusion Detection System. Leveraging this datasets helps to simulate a real-world network traffic scenarios.

Data Preprocessing

The preprocessing stage is significant stage of machine learning model pipeline espcecillay when dealing with network traffic data. Proper preprocessing ensure that the data is cleaned and uniform abd missing data is handled properly.

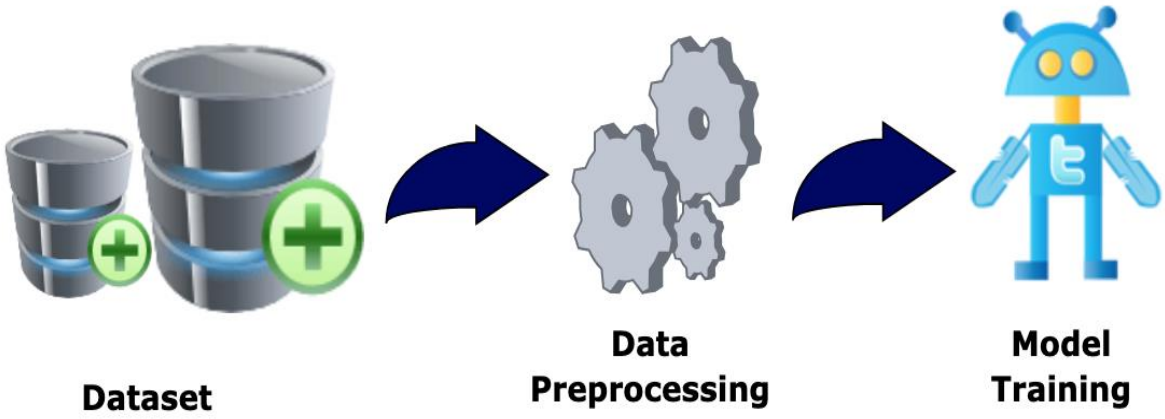

Figure 6 Data preporcessign flow

This ensures an effficient training and more accurate predctions. The followinf prepreocesing step was taken in this study:

Encoding Categorical Features

The NSL-KDD datasets contains various categorical features such as protocol-type, services and flags. This features represents different attrribute of the network traffics. This non-mumeric features need to be tranformed into numerical representation to make it suitable for machine learning algorithm. At this stage label encoding to this feature converting each category to a unique integer. For instance, protocol types such as TCP, UDP, ICMP are encoded into unique integers to ensure model can effectively intepret these features. This steps would help to preserve these features as well as allowing effective processing of the data by the Random Forest Classifier.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Scaling Numerical Features

To ensure that the numerical features is well accounted for in the model training the the numerical features was sclaed. In order to scale the numerical features, the MinMaxScaler method was used to scale the attributes to a standardized range typically 0's and 1's. The scaling measure is critical because it prevent features with larger attribute from dominating smaller magnitudes. Scaling this features can enhance model convergence speed and help to improve overall model performace by standardizing the input data.

Handling Missing Values

Ensuring that missing data or null values is handle correctly help to maintain the data quality.Missing value can lead to bottlenecks and inaccurate result when traning a model . In this study missing values was handled my inserting a default value in place to ensure smooth training of the model without any bias.

Feature Selection

One of the critical aspect of supervised learning is feature selection. Because it can significant reduce the time required for computation (Mahbooba et al., 2021).

Although the NSL-KDD datasets is comprehensive. However, it is not all may contribute equally to the model predictive power. To optimize the model and reduce training time, feature selection was performed. To effective perform this, feature importance anaylsis was performed using the inherent capabilities of the Random Forest Classifier model to retain important features. This steps is important because it enhances the model interpetability and chanle computational resouces to the most relevant data.

Balancing the Data

Inbalances ae common in datasets, where one class – in our case normal traffic, outnumbers others – specific attacks. These can skew model training to favour the majority class. To control this, techniques such as oversampling (using methods like SMOTE – Synthetic Minority Oversampling Technique) or undersampling the majority class was employed to ensure a more balanced dataset. These act ensure that model can detect normal and abnormal network traffic effreectively, reducing biases and improving classifying performance.

The SMOTE oversampling technique works by generating synthetic data points between existing samples of the minority class. For each minority class $\varkappa_i$ it finds its κ_{th} nearest nieghbour in the minority class, then randomnly select one of these neighbours $\varkappa_{neighbour}$.

The synthetic samples is computed as :

$$x_{synthetic} = x_i + \delta . (x_{neighbour} - x_i)$$

Data Splitting

The NSL-KDD dataset was split into training and testing sets. A typical splitting ratio been 80/20 – where 80% is for training and 20% is for testing. This ensure that the model has ample amount of data to learn from whule retaining a sufficient portion for evaluation purpose. In addition, a validation split was mainained with the training set for hyperparameter tuning. This help to optiize the model by adjusting the parameters such as the number of tress in the Random Forest Classifier.

All the process in the processing stage ensure that we have a high quality dataset to train our model effectively. This processing pipeline helps to facilitate the training and development of an Ai-assisted Intrusion Detection System that is capable of genrating custom rules in response to detetcted anomalies.

Model Development

For this dissertation, Random Forest Classifier – a subset of ensemble learning algorithm was selected to train a model using the NSL-KDD dataset. The selection was due to its versatitlity and effectiveness I handling categorical and numerical data.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

The Random Forest build multiple decision trees and merger them to obtain a more accurate prediction. The underlying theory behind this algorithm is based on bagging – a concept of training multiple instances of the same model with subset of the training datasets and avergaing the outputs.

Mathematical Representation

Let D be the original datasets with N data points. Random Forests creates M different training subsets D_1, D_2,D_3,….., D_M. Each sample randomly with replacement with D. The subset D_i is used to train decision tree T_i

The final output for a classification problem is voting among all the tress.

$\hat{y} = mode\{T_1(\mathcal{X}), T_2(\mathcal{X}),..., T_M(\mathcal{X})\}$

Where $\mathcal{X}$ is the input value feature vector and $T_M(\mathcal{X})$ is the prediction from the decision tree.

Feature Extraction

The relevant feature which may inlcude numerical and categorical fileds are extracted from the NSL-KDD dataset. This help to isolate attributes that most contribute to distinguishing between benign and malicious activities.

Mathematically, the feature importance can be representd as

$$I(f_j) = \sum_{t=1}^{M} \sum_{n \in nodes(t)} \frac{\omega_{\cap}\ \Delta_i\ (n, f_j)}{T}$$

Where:

- M is the number of trees,
- $\omega_{\cap}$ is the weighted number of samples reaching node nnn,
- $\Delta_i\ (n, f_j)$ is the decrease in impurity at node nnn for feature fjf_jfj,
- T is the total number of trees.

Hyperparameter Tunning

The process hyperparameter tuning involves the optimization of setting of machine learning algorithm that are not learned from the data byt set before the training process. Aside from paramter leanred from training data, hyperparameters control the behaviour of the training process. This process inRandom Forest Classifier can significantly control the behaviour and performance of the traning process.

For the pruporse of this reserch using NSL-KDD datasets baseline hyperparamters such as n_estimators = 100, max_depth = None and min_samples_split = 2 are tested to establish a starting point.

Training and Validation

In a reliable machne learning model, the training and validation phase is critical to building a reliable model. The phase involves using prepreocessed datasets to teach the random forest classifier how to identify pattern that constitutes anomalies in network traffic. The training data would usually includes input features and corresponding labels that guide the models that guides the model to understand the relationship between the inputs and their respective classification – in our case normal or abnormal. Random Forest Classifier creates multiple deicison trees in this phase, each trained with a subset of the training data. The prediction derived from this phase is then aggregated to from the final output.

At the validation phase we test model with unseen portion of the dataset to gauage generalization abaility. This steps is important because it ascertain that the model did not just memorize the data but can also perfrom well on new unseen data. Cross validation techniques such as k-cross validation are used to enhance reliability. In k-cross validation, data are splitted into k-equal parts and the model is trained and validated in k times, each

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

time suing different fold for validation and training. This method can help to mitigate the problem of overfitting and provides a comrephensive view on different data segments.

The outcome from validation is analyze using perfoamce metrics such as accuracy, precision, recall and F1-score. Ultimately, a model that performs effectively during validation is more likely to maintain its effectiveness when deployed for real-time analysis in NIDS environment.

Evaluation Metrics

To comprehensice evaluate the performance of our model, several metrics will be employed. These metrics would help to provide a balanced view on the model performance, enabling both qualititative and quantitative analyses to validate the results.

Accuracy

Accruacy is one of the straight-forward metrics, it represents propportion of coreectly classified instances out of the total instances. It is calculated as:

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

Where:

- TP (True Positives): Instances correctly predicted as positive (e.g., actual malicious traffic identified correctly).
- TN (True Negatives): Instances correctly predicted as negative (e.g., actual benign traffic identified correctly).
- FP (False Positives): Instances incorrectly predicted as positive (e.g., benign traffic misclassified as malicious).
- FN (False Negatives): Instances incorrectly predicted as negative (e.g., malicious traffic misclassified as benign).

While accuracy has been found effective in initial datasets, it is not always reliable with datasets with imbalances. For instances a model biased towards normal prediction could appear highly accurate but perfrom poorly in detecting actual threats.

Precision meausre the model's ability to correctly identify positive instances reducely mistake:

$$Precision = \frac{TP}{TP + FP}$$

High precision score signals low false positive in the model which is critical in reduing false alarm in a Network Intrusion Detection System.

Recall – as an evaluaion metric, evaluates how well the model identifies all relevant positive instance:

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

A high recall signals that the model captyres the true malicious instances. Minimising missed detections, which is important for security application where missed instances can have a profound consequence.

Lastly, F1-Score is the harmonice mean of precision and recall, offering a balanced metrics that accommodate both fals positive and false negative:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The F1-socre is the harmonic mean of precion and recall, offering a trade-off between precision and recall.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Confusion Matrix Analysis

Confusion matric (CM) also helps to create a better clarification on accuracy of model training by creating a visual representation of outcomes. The classification of CM includes true positives, true, negatives, false positive and false negatives. This provides insights on how the model can clearly distinguish between normal and malicious network traffic. Analysing the result in the CM table would be to observe whether the model correctly distiniguish traffic pattern or how accurate it performs in classification of network traffic.

Table 1 Confusion matrix table

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Feature Importance Analysis

The faeature importance in Random Forest Classifier model helps to interpret which features contribute significance to model predictions. The Random Forest algorithm calculates feature importance by measuring the impact of each feature on the model's predictive power, typically based on:

- Decrease in impurtiy when the feature is used to split nodes
- The average decrease node impurity weighted by nth number of sample in each split

System Architecture

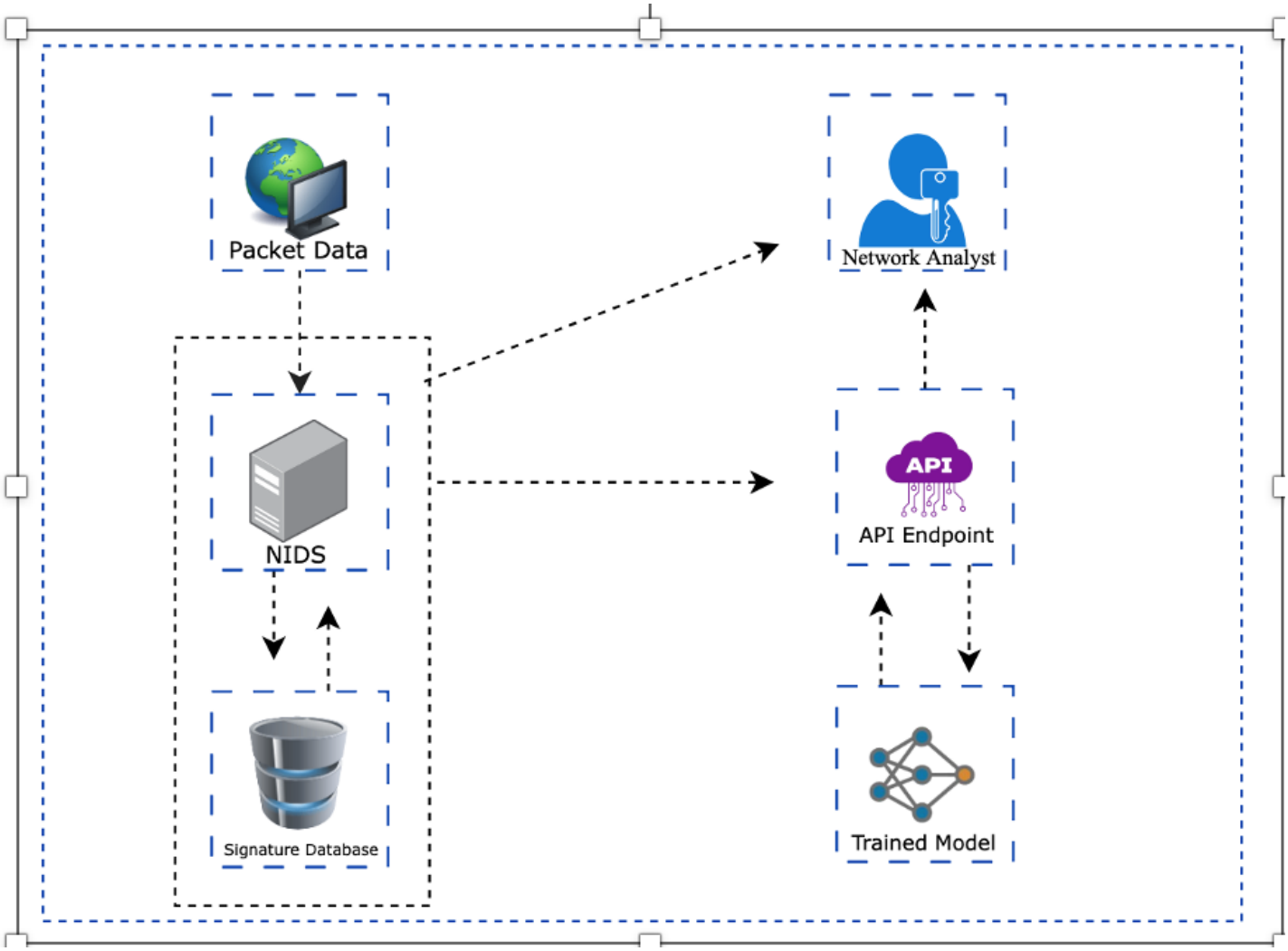

Figure 7 Proposed system architecture for custom rules generation

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

In the proposed system architecture, incoming network packets are processed by a NIDS connected to the network. The NIDS performed a signature-based analysis on the incoming traffic by checking the traffic against its database to classify the traffic. Simultaneously, it sends the traffic to a REST API endpoint that logs the data for future training and further analyse the traffic with a pre-trained model to identify and potential matches that might be missed from a predefined rule in a signature database. This dual layer approaches enhance detection accuracy and potentially help security analyst to create custom rules.

RESULT AND DATA ANALYSIS

The aim of this chapter is to discuss and analyse the result obtain from AI-assisted rules generation for Network Intrusion Detection System. In other to evaluate the model strength and limitation, metric such as accuracy, precision, F1 score and confusion metric matrix, are employed to provide understanding into the model performance.

Dataset Preprocessing

Training a large dataset such as the NSL-KDD dataset requires a strategic approach in the pre-processing state to ensure that the machine learning can learn effectively from the dataset. Here we analyse the preprocessing steps applied to the NSL-KDD dataset which include data splitting, feature selection, hyperparameter tuning, and balancing the dataset. The goal is to improve the quality of data that is fed into the model to ensure it produce accurate result.

Data Splitting

At this stage rather than rely on the conventional approach of using predefined test and train dataset available. The NSL-KDD train and test datasets was combined into a single large dataset and then a custom split was performed.

After combining the datasets, into was later divided into two parts: 65% for training and 35% for testing. The rationale behind the 65/35 split is to ensure that the model had a sufficient dataset to learn from. As well as leaving an adequate amount of data for evaluating and evaluating the model's ability. The splitting process was performed randomly while maintaining stratification of the data ensuring that each type of traffic is maintained in both the training and testing datasets. This is important in other to ensure that the rare attack types such as R2L and U2R is represented in the two datasets.

Feature Selection

After the data splitting stage, the next process in the preporcessing stage was feature selection. Given that the NSL-KDD dataset has 41 features, and not all features are necessary for model training. Some features are considered to be redundant that might introduce noise and reduce model perfromance, To cater for this, we perform Recursive Feature Elimination, and uses Gain to identify and retain the most import feature.

The feature selection process result ina refined features that were ore impactful in differentiating between normal and malicios network traffic. Key feature that was critical to model perfromance such as protocol type, source bytes were retained. Redundant or less relecant feautes was discadrded. The feaure importance chart shown below reveals how the various features contribute to how the different features contribute to the model development.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

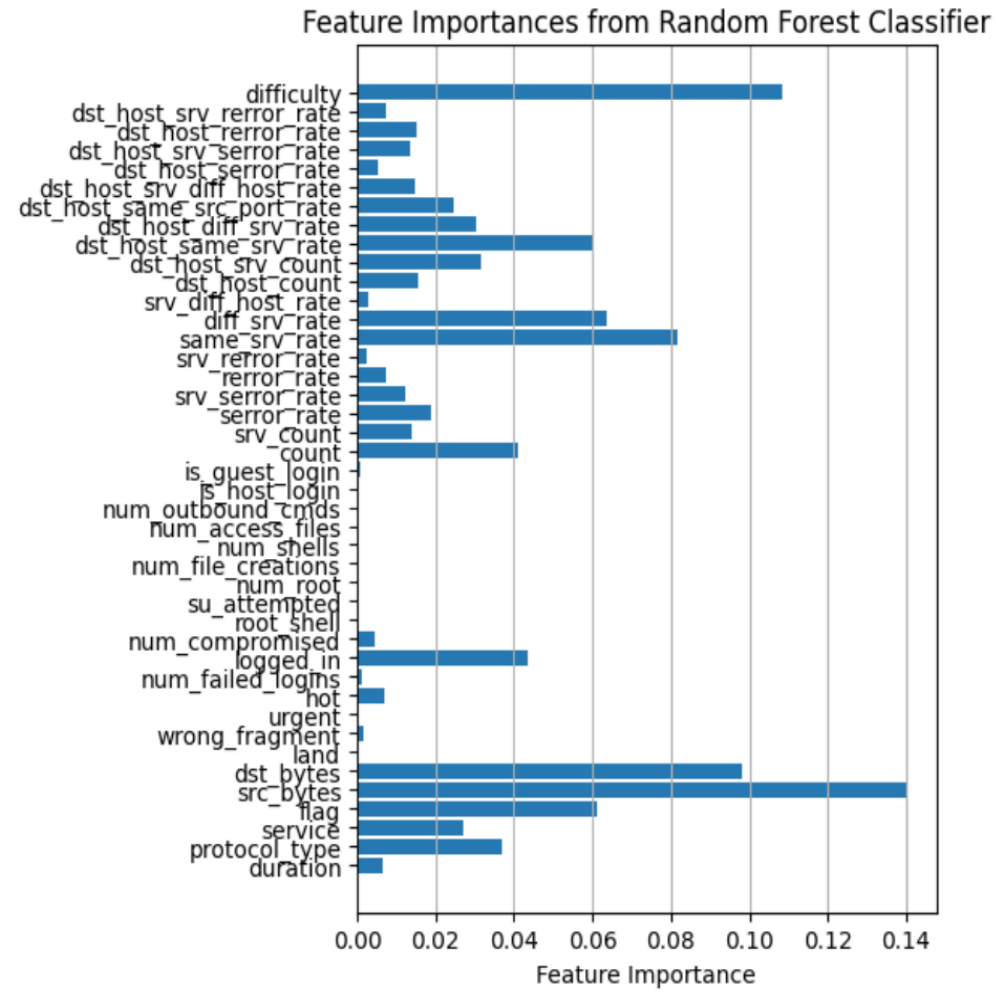

Figure 8 Model feature response result

Table 2 Feature selection result data

Feature	**Importance Score**
duration	0.0063
Protocol_type	0.0369
service	0.0269
Flag	0.0613
Src_bytes	0.1406
Dst_bytes	0.0981
Land	0.0000
Wrong_fragment	0.0016
Urgent	0.000
Hot	0.0068
Num_failed_login	0.0013
Logged_in	0.0437
Num_compromised	0.0044
Root_shell	0.0001
Su_attempted	0.0000
Num_roots	0.0003
Num_file_creations	0.0001
Num_shells	0.0000
Num_acess_files	0.0001
Num_outbound_cmds	0.0000
Id_host_login	0.0000

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Feature	Important Score
Is_guest_login	0.0008
Count	0.0411
Srv_count	0.0141
Serror_rate	0.0189
Srv_serror_rate	0.0122
Rerror_rate	0.0075
Srv_rerror_rate	0.0025
Same_srv_rate	0.0818
Diff_srv_rate	0.0635
Srv_diff_host_rate	0.0030
Dst_host_count	0.0157
Dst_host_srv-count	0.0318
Dst_host_same_srv_rate	0.0598
Dst_host_diff_srv_rate	0.0305
Dst_host_same_src_port_rate	0.0245
Dst_host_srv_dff_host_rate	0.0147
Dst_host_serror_rate	0.0051
Dst_host_srv_serror_rate	0.0135
Dst_host_rerror_rate	0.0150

Discuss result form table

The features important chart shows how different features of the NSL-KDD dataset contribute to the model training. The result from the chart can be classified to most important, moderately important and least important features.

Most Important Features

Features such as src_bytes (0.1406), and dst_bytes (0.0981) have the highest importance socres. This result suggest that the network traffic transfer in both direction in the network is a singnificant factor in classifying a traffic as normal or abnormal.

Also count (0.0411), and logged_in (0.0437) aslo show considerable importance. Suggesting that number of connection and login status are alo very relevant for classification

Moderately Important Feature

Features such as flag (0.0613), same_srv_sate (0.0818) and diff_srv_rate (0.0635) have morderate importance score. This suggest that type of connection as well as rate of connection to the differetn service in the network is a moderate important features in distinguishing between a normal or malicious traffic. Other moderately important feature to consider are service(0.0269) and protocol_type (0.0369) are also a strong contributor in this classification.

Least Important Features

Least important features show features that contribute the least to the model training. Several feature land (0.0000), num_root (0.0003), root_shell(0.0001), su_attemoted(0.0000) has the lowest feature importance score. Showing that this attacks did not provide enough data for the model to learn from.

Balancing the Dataset

The NSL-KDD dataset - just like many large datasets, suffers from class imbalance. Where classes such as normal traffic is overrepresented compared to classes such as U2R (User-to-root) that are underrepresented. Data imbalance such as this can lead to model performing poorly in detecting minority class attack. Making

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

the model biased toward predicting majority class. SMOTE oversampling techniques was applied in other to balance the dataset before subsequently training the model. After applying this technique, the dataset shows a more balanced dataset. As a result, it will contribute to the model ability to detect minority class attack as well as properly do a binary classification of the network traffic.

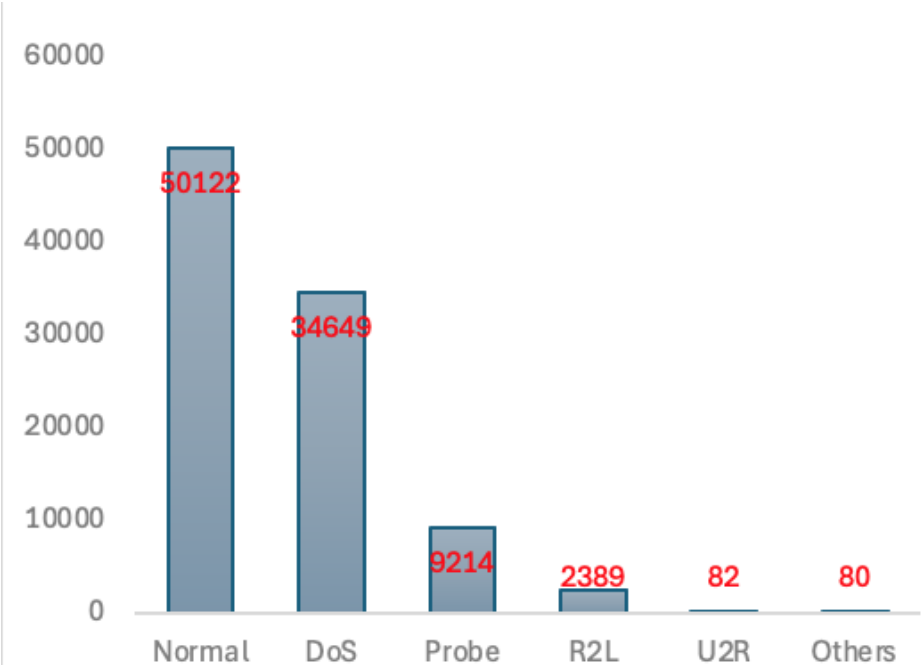

Figure 9 Original dataset distribution

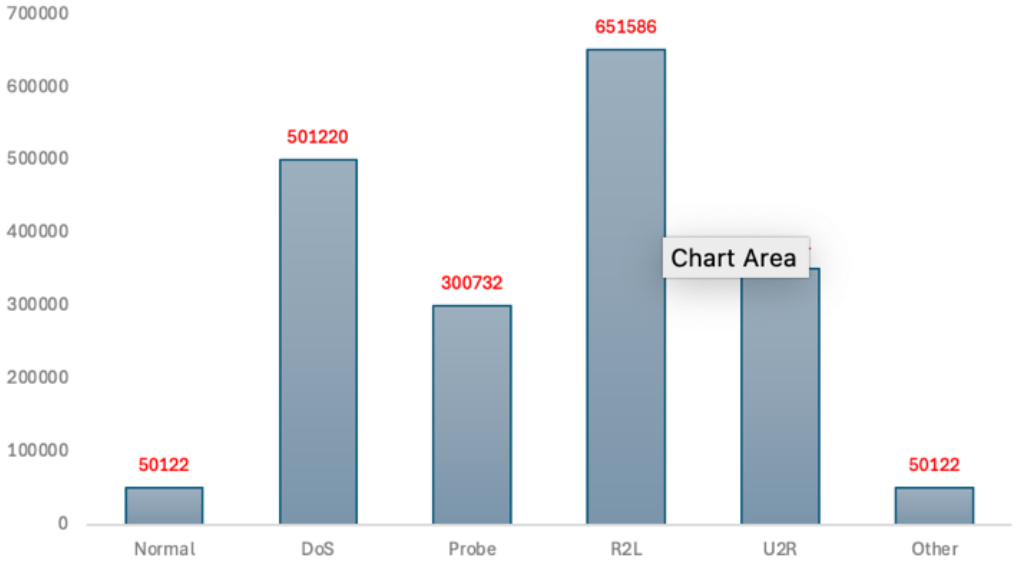

Figure 10 Dataset distribution after balancing

Model Development

For model development, Random Classifier librbary of the scikit pyton librbary was used to train the NSL-KDD dataset, and using the label datasets categories, an approach to create cusom-rules was established.

To predict a network traffic and assist with custom rules, the system process network packet logs from a NIDS system, using the Random Forest Classifier, it first does a binary classification of the network traffic with probability of the accuracy. Afterwards, using the attack category the Random Classifier was trained with, it assists network analyst to suggest custom to mitigate this. Figure 11 above shows a malicious sample malicious network packet coming from a NIDS log. The network packet is sent as a payload to an API endpoint that accepts this payload and process the data to an acceptable format in the model training.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

```
[**] [1:4000001:1] Smurf Attack (ICMP Echo Request with spoofed source) [**]
[Classification: Denial of Service] [Priority: 1]
04/13-18:05:00.123456 192.168.1.10 -> 192.168.1.255
ICMP TTL:255 TOS:0x0 ID:12345 IpLen:20 DgmLen:80
Type:8  Code:0  ID:0  Seq:2
X.X.X.X: 0.0.0.0 -> X.X.X.X: 0 len=80
```

Figure 11 Malicious Network Traffic


```
{
    "prediction": 1,
    "probability": 0.51,
    "suggested_rules": [
        [
            "alert tcp any any -> any any (msg:\"Possible DoS attack detected: src_bytes exceeds threshold\"; src_bytes > 80.00;)",
            "alert tcp any any -> any any (msg:\"Possible DoS attack detected: dst_bytes exceeds threshold\"; dst_bytes > 0.00;)"
        ],
        "alert tcp any any -> any any (msg:\"Possible DoS attack detected: src_bytes exceeds threshold\"; src_bytes > 80.00;)",
        "alert tcp any any -> any any (msg:\"Possible DoS attack detected: dst_bytes exceeds threshold\"; dst_bytes > 0.00;)"
    ]
}
```


Figure 12 API response for custom rule generation

Figure 12 above shows a typical reponse response for a malicious attack. Firstly, the prediction =1 shows that the model recognize the packet as malicious and the proabaility = 0.51 reveals the level of certainity of this in the classification of the network packet. Followed by the suggest rules that contains message to assist network security analyst quickly get insight into an attack and quickly responds with custom rules to mitigate this.

```
[**] [1:1000001:1] ICMP PING [**]
[Classification: Misc activity] [Priority: 3]
04/13-16:52:43.268087 192.168.1.10 -> 192.168.1.20
ICMP TTL:64 TOS:0x0 ID:25575 IpLen:20 DgmLen:56
Type:8  Code:0  ID:5555  Seq:1
X.X.X.X: X.X.X.X  len=56

```

Figure 13 Normal network traffic packet


```
{
    "prediction": 0,
    "probability": 0.5,
    "suggested_rules": [
        "No attack detected; normal traffic observed."
    ]
}
```


Figure 14 API response - normal network packet

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Figure 14 above shows the API endpoint response when it receive a normal network packet. After doing a binary classification of the network packet, it just logs the traffic details for future model training.

Evaluation Metrics

The result from the confusion metrics matrix shown in Fig.11 below, shows that the model is near-perfect in classifying network trafiics as either malicious or normal. With 26,908 true positive ans 25,034 true negatives, the model was able to predict nearly all instances in both classes accurately.

The exceptional low values for false positive and false negatives indicates that the model has an excellent capability to be able identify malicious network activities without necesaarily flagging benign traffic as malicious.

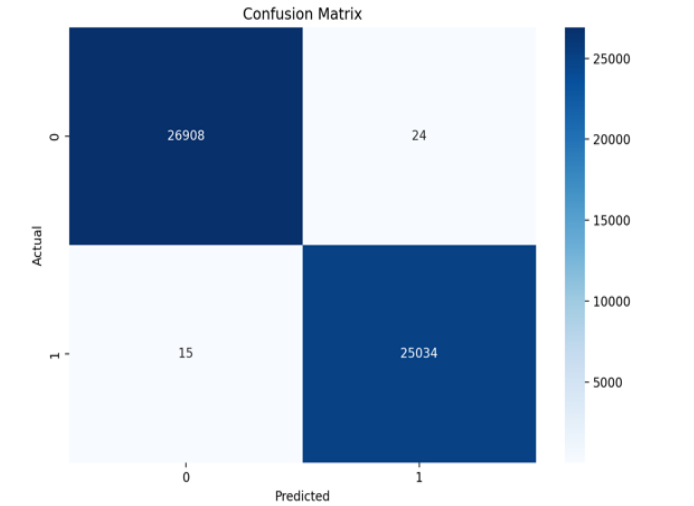

Figure 15 Confusion matrix result

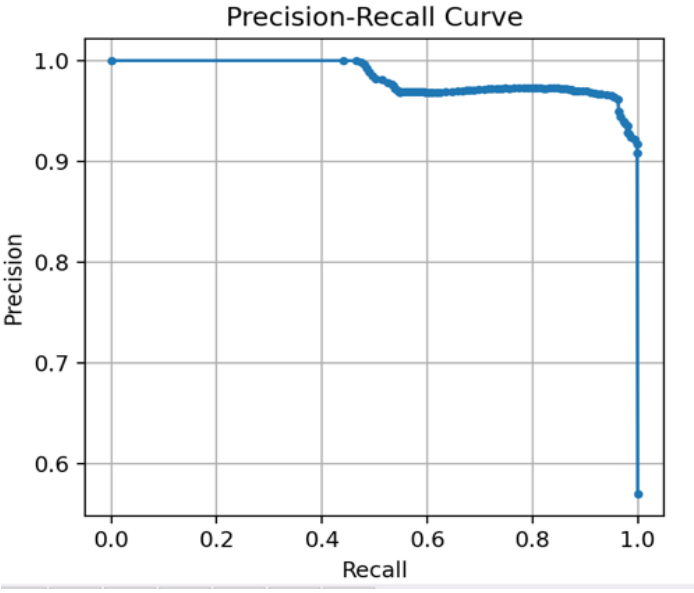

Figure 16 Precision - recall curve chart

The Precision-Recall (PR) curve in the figure 12 show high precision as the graph travels across most recall level, which demonstrate the model ability identify malicious network traffic with minimal false positive and ensuring that alerts are meaningful and actionable. However, as the recall increases toward 1.0 in the PR curve, precision had some slight decline. An indication that the model sacrifices some accuracy in other to detect more malicious traffic instance. The trade-off is typical of PR curves, where achieving maximum recall can lead to an increase in false positive.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Table 3 Evaluation metrics data

Metric	Value
Accuracy	99.92%
Precision	99.90%
Recall	99.94%
F1-Score	99.92%

The evaluation metrics of the model shown in the table above demonstrate that the model is exceptional effective in classifying network traffic as either normal or malicious. The accuracy of 99.92%, is reliable in network binary classification, drastically reducing false positive that can affect operational efficiency. The high accuracy that the model only has few instances that was classified wrongly.

The precision of 99.90% highlight the model ability to identify malicious traffic with minimal false positive positives. In real-time deployment, reducing false positive in very critical to the sanity of a NIDS. Excessive alert from false threat can overwhelm network administrator or the network security engineer and may even obscure legitimate alert. Also, the recall 99.94% reflects the model ability to be able to detect malicious activity. As a result, legitimate threat is rarely missed.

In addition, F1-Score of 99.92%, confirm balance between precision and recall metrics. The metrics underscore model's ability to correctly flag malicious network activity as well as minimize false alarm in the process. Together, these metrics signals that the trained model is not only accurate but also efficient and can be suitable in real-world application.

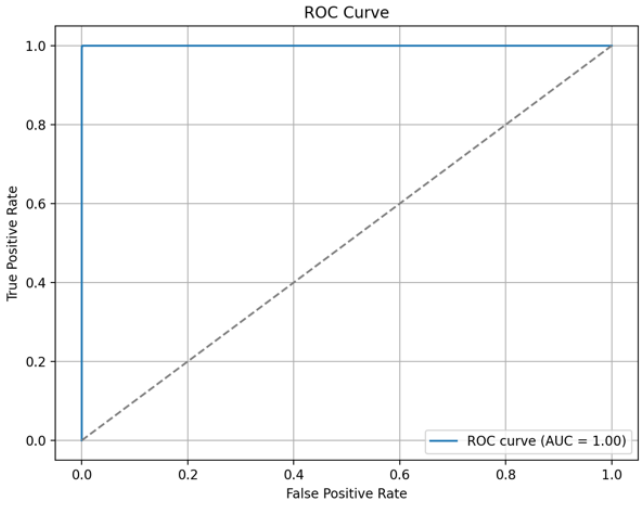

Figure 17 ROC curve chart

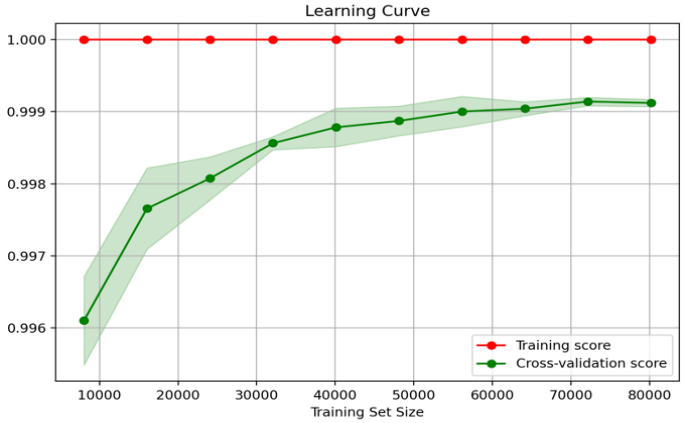

Figure 18 Decision learning curve

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Result from the ROC curve shows an AUC value of 1.00 which indicates a perfect classification. This signifies that the model can correctly distinguish between malicious and normal network traffic with no errors. Signalling the model achieves high sensitivity in detecting malicious activity, which is very critical in the reliability of as Network Intrusion Detection System (NDS).

The decision learning curve indicates random Forest achieves perfect training accuracy, as evident by the red line remaining constant at 1.00 across training sizes. This suggest that the model fits perfectly - this is expected for an ensemble language such as Random Forest to handle complex dataset such as the NSL-KDD dataset.

Significance of result

The metrics from data analysis of our model reveals some imoportant information that we can explore deeper in the following section

High Accuracy and Model Effecitiveness

The near-perfect classification metrics such as precision, recall, F1-score and accuracy, signifies the effectiveness of the model in distinguishing between malicious and normal network traffic. The level accuracy signifies that the model can effectively detect network threat. In a production environment this can translate to few missed threats. Also, the low rate of false positive from the confusion metric matrix can reduce unnecessary alert for network administrators improving their ability to detect genuine threat.

Model Overfitting Concern

While the result appears to be flawless there are critical concern about potential overfitting, which occurs when a model performs exceptionally well on the training and test data but falters in real-world scenario. Networks are susceptible to novel attacks. As a result, NSL-KDD dataset may not be robust enough to train the model to detect network threat that is not included in the NSL-KDD dataset. In addition, data imbalance can lead to model to misclassify genuine threat as normal network traffic. In other to mitigate this, balancing technique such as SMOTE technique was used to balance the distribution the attack classes.

Precision-Recall Trade-Off

The precision-recall curve suggests model's ability to maintaining both high precision and recall across. Various threshold. This signifies that in high-stake environment where missing a single threat can be catastrophic consequence. The high recall ensure recall ensure that broad net is cast, and precision signifies that unnecessary disruption is curbed.

Enahancing custom rule creation for NIDS

Implementing machine-driven rule into the process of creating custom rules for signature-based NIDS makes a significant advancement in creating an adaptable NIDS. Traditional NIDS rely heavily on custom rules created by network expert to compliment the predefined rule in the signature database of the NIDS, which are limited and often limited in the dynamic nature of modern network environment. This project addresses this limitation by automating the process of manual rule creation with train machine learning model.

The process of crafting effective rules for NIDS system are usually labour intensive, time consuming as well as deep expertise in the domain. In addition, static rule can be counter-productive because of the dynamic nature of network environment. This project mitigates the process of custom rule creation by based on insight derives from data itself. Allowing network expert gets insights on network data over time, and craft rules that are custom to the environment. Leveraging on NSL-KDD dataset and a trained Random Forest Model at inception, system can dynamically identify most relevant features and generate rules. Subsequently, the model learns from known network pattern and traffic peculiar to a particular network environment to create custom rules that are peculiar to that system.

The radon forest model achieves an accuracy of 99.92% with a recall of 99.94% for malicious traffic, demonstrating its strong capability in identify malicious network activities. The rule generation was able to create a contact-aware Snort rules tailored to the dataset attack pattern.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

Compared to traditional manual rule creation that relies heavily of network administrator domain knowledge of the organization network environment, a data-driven approach was embraced in this studies that ensure that subtle attack vector is uncovered.

Research Limitation

The model binary classification of threat serves as both strength and weakness. While it can effectively classify traffic as either normal or malicious, this method oversimplifies that complexity of network traffic. Real-words attack is diverse in nature, each requiring unique responses. A larger dataset, can provides better representation of legitimate or malicious attack patterns, thereby improve the model ability to be able to classify attack accurately.

In addition, the model at first would be biased towards the traffic pattern in the NSL-KDD dataset at inception. However, gradual integration to live network traffic makes the model learn from real-world scenario that is unique to the organisation and adjust accordingly.

Another challenge is to ensure rules generated are actionable and interpretable, as automated system are known to either develop rules that are too broad or complex, as a result, making it very difficult for security analyst to apply in a real-world environment. Managing this trade-off always a pose a challenge. When the rules are too specific, they are might only apply to a set of attack. In contrast, when they are too broad, they may generate high volume of false positives, as they fail to discriminate well between normal and malicious traffic. To balance this trade-off, it is not only important to statistical consideration but also domain expertise.

Ethical Consideration

Ethical consideration is paramount when designing an AI-assisted rule generation for Network Intrusion Detection System (NIDS). One of the primary concerns is data privacy and security, especially when handling real-world network traffic data in a production environment. Although NLS-KDD dataset is publicly available, yet an application of the system would require user data are anonymized and safeguarded against exposure or misuse. In production environment, where API processes live network traffic, robust encryption and anonymisation practice should be adopted to ensure that sensitive information is well protected.

Also, another significant ethical issue is fairness and accountability in the system decision-making process. Machine learning models are prone to the challenge of class imbalance - in this case of this study can lead to wrongly classification of network traffic. This can lead to the model biased towards a particular network pattern that it is used to. Proper technique should be put in place to ensure fairness between the network traffic patterns.

Evaluation and Implication

The accuracy of 99.89% refelcts the model's ability to classify network as normal or malicious effectively. The Precision-recall curve also higlighted the model's avility to balance false positve and false negeative making the model highly effective in a production environment. In addition, the ROC curve with AUC of 1.00, confirms the model's robustness.

The performance of the learning model correlates the computational resource that is available during training. For this study, the Random Forest model was trained and tested on a hardware quipped with 2.9 GHz Dual-Core Intel Core i5, 16GB RAM and no dedicated GPU. The computational environment also includes macOS Monterey v12.6 operating system. In addition, libraries and framework employed during trainman include Python 3.8, Scikit-learn library, NumPy, pandas, matplotlib, seaborn, joblib and flask.

The resources were sufficient for training and testing with average time of training been 8.88 seconds Despite high efficiency recorded with this dataset, computational constraint needs to be considered when dealing with a large dataset. This because Random Forest algorithm computes in parallel, as a result, large datasets with millions value requires more memory and computational power. In such cases, cloud technologies with

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

scalable and distributed architecture such as AWS EC2 or Google Colab with GPUs to reduce compute time and enhance scalability

Comparison with Benchmarks or Existing Studies

In this study, the NSL-KDD train and test was combined to into a single large datasets and splitter into training and training sets at a 65:35 ratio. This approach differs from conventional method of using predefined training and testing subset provided by the dataset. The rationale was to ensure a randomise split that expose the dataset to diverse data distribution within the dataset. While the method reduces the risk of overfitting for specific attack types that is prevalent in the training sets, it also means that comparing with previous studies that uses standard splits may not be entirely equivalent.

The merging and random splitting methodology has a significance on the performance metrics, yielding a higher value due to the overlapping in data distribution between the training and testing sets. Nevertheless, the result demonstrates a strong trend of improvement. For instance, the proposed model achieved accuracy and recall of 99.89% and 99.94% respectively. Indicating its robustness and efficiency in detecting malicious network traffic. While exact comparison with previous studies maybe challenging, trends in prior research have consistently shown that Random Forest Classifier to perform strongly in intrusion detection tasks. Studies employing SVM typically report accuracies of 97-98%, while Naive Bayes and KNN generally achieve lower ranges of 85-95%, making the proposed model's results notably competitive.

CONCLUSION AND RECOMMENDATION

The research focused on leveraging artificial intelligence to assist with rule custom rule generation for Network Intrusion Detection System (NIDS). Using the NSL-KDD dataset, the study was able prove the potential machine learning model to classify network traffic effectively and provide automated rule generation to enhance network security. This approach addresses the limitation of traditional signature-based NIDS such as ability to adequately evade evolving network threats. The integration of AI into the process of custom rule generation enhances system accuracy's scalability, and responsiveness. However, challenges such as class imbalance, ethical implication, and deployment were evident. Overall, the study underscores the importance of machine learning model in reinforcing the security in NIDS while highlighting the needs for careful consideration of theca, legal and technological factor.

Recommendation

Dataset Diversification

Dataset diversification is essential for enhancing the robustness of model performance in NIDS. While the NSL-KDD dataset is widely used and acceptable dataset from training model on network traffic, it still has its own limitation with regards to limited attack types. Combining with other publicly available dataset such as the UNSW-NB15, CICIDS2017 dataset can help broaden the attack types that the model can learn from. Also, heterogeneity dataset is also crucial because it can better model a real-world scenario that includes data from diverse environment such as enterprise, cloud, IoT networks etc.

Leverage GPT for enhance rules generation

Integrating GPT can serve as an intelligent assistant for cybersecurity analysts to help validate the rules generated as well as providing detailed explanation and applicable alternative. This can lead to enhanced interpretability that can help bridge the gap between automated system and human oversight. Future research work could explore the feasibility, security and scalability of deploying a trained GPT with network-related data to an AI-NIDS system.

Federated Learning Approach

A federated learning approach for an AI - assisted NIDS offers a potential of enhanced model detection for diverse attack patterns, and well suited for modern organization that offers various services and have network data coming from different sources and location. This approach can help the model make more accurate

decision across different range of scenario. Integrating federated learning approach into the trained model can achieve enhanced detection capabilities, as well as fostering collaboration in a globally connected cyberspace.

Ethical and legal compliance

Ethically, future works should focus on transparency and fairness. Model should be more interpretable - allowing users and stakeholder understand underlying factor behind every decision. Also, considerable efforts should be put in place to prevent misuse technology for unethical purposes.

Legally, compliance with prevalent data privacy regulation such as GDPR, CCPA, and other international laws should be adhered to. Future research can incorporate data collection pipeline that ensure that privacy is ensured. Privacy technique techniques such as user anonymisation, secure aggregation, and cryptography should be integrated into network data collection process.

REFERENCES

1. Ahmad, M., Riaz, Q., Zeeshan, M., Tahir, H., Haider, S. A., & Khan, M. S. (2021). Intrusion detection in internet of things using supervised machine learning based on application and transport layer features using UNSW-NB15 data-set. Eurasip Journal on Wireless Communications and Networking, 2021(1), 1–23. https://doi.org/10.1186/S13638-021-01893-8/FIGURES/9
2. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021a). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 32(1), e4150. https://doi.org/10.1002/ETT.4150
3. Alonso, E., Javier Garcia Villalba, L., de Sousa, R. T., de Oliveira Albuquerque, R., Lucila Sandoval Orozco, A., Díaz-Verdejo, J., Muñoz-Calle, J., Estepa Alonso, A., Estepa Alonso, R., & Madinabeitia, G. (2022a). On the Detection Capabilities of Signature-Based Intrusion Detection Systems in the Context of Web Attacks. Applied Sciences 2022, Vol. 12, Page 852, 12(2), 852. https://doi.org/10.3390/APP12020852
4. Alonso, E., Javier Garcia Villalba, L., de Sousa, R. T., de Oliveira Albuquerque, R., Lucila Sandoval Orozco, A., Díaz-Verdejo, J., Muñoz-Calle, J., Estepa Alonso, A., Estepa Alonso, R., & Madinabeitia, G. (2022b). On the Detection Capabilities of Signature-Based Intrusion Detection Systems in the Context of Web Attacks. Applied Sciences 2022, Vol. 12, Page 852, 12(2), 852. https://doi.org/10.3390/APP12020852
5. Bhavsar, M., Roy, K., Kelly, J., & Olusola, O. (2023). Anomaly-based intrusion detection system for IoT application. Discover Internet of Things, 3(1), 1–23. https://doi.org/10.1007/S43926-023-00034-5/FIGURES/7
6. Choi, H., Kim, M., Lee, G., & Kim, W. (2019). Unsupervised learning approach for network intrusion detection system using autoencoders. Journal of Supercomputing, 75(9), 5597–5621. https://doi.org/10.1007/S11227-019-02805-W/TABLES/10
7. Faisal, M., & Islam, M. S. (2023). Improving Network Security with Intrusion Detection Systems Utilizing Machine Learning and Deep Learning Techniques. 2023 26th International Conference on Computer and Information Technology, ICCIT 2023. https://doi.org/10.1109/ICCIT60459.2023.10441582
8. Hara, K., & Shiomoto, K. (2020). Intrusion Detection System using Semi-Supervised Learning with Adversarial Auto-encoder. Proceedings of IEEE/IFIP Network Operations and Management Symposium 2020: Management in the Age of Softwarization and Artificial Intelligence, NOMS 2020. https://doi.org/10.1109/NOMS47738.2020.9110343
9. Jabez, J., & Muthukumar, B. (2015). Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach. Procedia Computer Science, 48(C), 338–346. https://doi.org/10.1016/J.PROCS.2015.04.191
10. Jin, S., Chung, J. G., & Xu, Y. (2021). Signature-based intrusion detection system (IDS) for in-vehicle CAN bus network. Proceedings - IEEE International Symposium on Circuits and Systems, 2021-May. https://doi.org/10.1109/ISCAS51556.2021.9401087

11. Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity, 2(1), 1–22. https://doi.org/10.1186/S42400-019-0038-7/FIGURES/8
12. Kim, A., Park, M., & Lee, D. H. (2020). AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection. IEEE Access, 8, 70245–70261. https://doi.org/10.1109/ACCESS.2020.2986882
13. Kumar, M. S., Ben-Othman, J., Srinivasagan, K. G., & Krishnan, G. U. (2019). Artificial Intelligence Managed Network Defense System against Port Scanning Outbreaks. Proceedings - International Conference on Vision Towards Emerging Trends in Communication and Networking, ViTECoN 2019. https://doi.org/10.1109/VITECON.2019.8899380
14. Latha, R., & Thangaraj, S. J. J. (2023). Securing the Digital Perimeter: A Comprehensive Intrusion Detection System with Ensemble Learning. 2023 International Conference on Data Science, Agents and Artificial Intelligence, ICDSAAI 2023. https://doi.org/10.1109/ICDSAAI59313.2023.10452636
15. Mahbooba, B., Timilsina, M., Sahal, R., & Serrano, M. (2021). Explainable Artificial Intelligence (XAI) to Enhance Trust Management in Intrusion Detection Systems Using Decision Tree Model. Complexity, 2021(1), 6634811. https://doi.org/10.1155/2021/6634811
16. Mbona, I., & Eloff, J. H. P. (2022). Detecting Zero-Day Intrusion Attacks Using Semi-Supervised Machine Learning Approaches. IEEE Access, 10, 69822–69838. https://doi.org/10.1109/ACCESS.2022.3187116
17. Papadogiannaki, E., & Ioannidis, S. (2021). Acceleration of Intrusion Detection in Encrypted Network Traffic Using Heterogeneous Hardware. Sensors 2021, Vol. 21, Page 1140, 21(4), 1140. https://doi.org/10.3390/S21041140
18. Pu, G., Wang, L., Shen, J., & Dong, F. (2021). A hybrid unsupervised clustering-based anomaly detection method. Tsinghua Science and Technology, 26(2), 146–153. https://doi.org/10.26599/TST.2019.9010051
19. Qureshi, A. U. H., Larijani, H., Mtetwa, N., Javed, A., & Ahmad, J. (2019). RNN-ABC: A New Swarm Optimization Based Technique for Anomaly Detection. Computers 2019, Vol. 8, Page 59, 8(3), 59. https://doi.org/10.3390/COMPUTERS8030059
20. Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. SN Computer Science, 2(3), 1–21. https://doi.org/10.1007/S42979-021-00592-X/FIGURES/11
21. Taher, K. A., Mohammed Yasin Jisan, B., & Rahman, M. M. (2019). Network intrusion detection using supervised machine learning technique with feature selection. 1st International Conference on Robotics, Electrical and Signal Processing Techniques, ICREST 2019, 643–646. https://doi.org/10.1109/ICREST.2019.8644161
22. Thankappan, M., Rifa-Pous, H., & Garrigues, C. (2024). A Signature-Based Wireless Intrusion Detection System Framework for Multi-Channel Man-in-the-Middle Attacks Against Protected Wi-Fi Networks. IEEE Access, 12, 23096–23121. https://doi.org/10.1109/ACCESS.2024.3362803
23. Verkerken, M., D'hooge, L., Wauters, T., Volckaert, B., & De Turck, F. (2022). Towards Model Generalization for Intrusion Detection: Unsupervised Machine Learning Techniques. Journal of Network and Systems Management, 30(1), 1–25. https://doi.org/10.1007/S10922-021-09615-7/FIGURES/6
24. Verma, J., Bhandari, A., & Singh, G. (2023). Spark-based Distributed Intelligent Network Intrusion Detection System for Unified Dataset. 2023 International Conference on Artificial Intelligence and Applications, ICAIA 2023 and Alliance Technology Conference, ATCON-1 2023 - Proceeding. https://doi.org/10.1109/ICAIA57370.2023.10169765
25. Young, C., Zambreno, J., Olufowobi, H., & Bloom, G. (2019). Survey of automotive controller area network intrusion detection systems. IEEE Design and Test, 36(6), 48–55. https://doi.org/10.1109/MDAT.2019.2899062
26. Zavrak, S., & Iskefiyeli, M. (2020). Anomaly-Based Intrusion Detection from Network Flow Features Using Variational Autoencoder. IEEE Access, 8, 108346–108358. https://doi.org/10.1109/ACCESS.2020.3001350

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

APPENDIX

Appendix B – Code Snippet API endpoint

```
from flask import Flask, request, jsonify

import joblib

import numpy as np

import pandas as pd

from preprocess import preprocess_nids_log


# Load the saved model, scaler, and label encoders

rf_model = joblib.load('../model/rf_model.pkl')  # Load the trained RandomForest model

scaler = joblib.load('../model/scaler.pkl')  # Load the MinMaxScaler


# Load label encoders for categorical columns

protocol_type_encoder = joblib.load('../model/protocol_type_encoder.pkl')

service_encoder = joblib.load('../model/service_encoder.pkl')

flag_encoder = joblib.load('../model/flag_encoder.pkl')


# Create Flask app

app = Flask(__name__)



# Define the function to generate rules

def generate_rules(feature_names, feature_importances, input_data, prediction, attack_class, threshold_factor=0.75):

    """

    Generate rules dynamically based on feature importances, attack class, and thresholds derived from input data.



    """

    rules = []
```

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

```
# If prediction is 1 (attack), generate rules based on attack class
if prediction == 1:
    attack_label = f"Potential {attack_class} Attack Detected"

    # Generate attack-specific rules
    if attack_class == "DoS":
        # Denial of Service (DoS) specific rules (example: high src_bytes)
        rules.append(generate_dos_rules(input_data, threshold_factor))

    elif attack_class == "Probe":
        # Probe specific rules (e.g., scanning behavior)
        rules.append(generate_probe_rules(input_data, threshold_factor))

    elif attack_class == "R2L":
        # Remote to Local (R2L) specific rules (e.g., failed logins, root access attempts)
        rules.append(generate_r2l_rules(input_data, threshold_factor))

    elif attack_class == "U2R":
        # User to Root (U2R) specific rules (e.g., elevated privileges access)
        rules.append(generate_u2r_rules(input_data, threshold_factor))

    # Dynamically calculate thresholds for other features based on the input data
    for feature, importance in zip(feature_names, feature_importances):
        if importance > 0.1 and feature != 'difficulty':  # Exclude irrelevant features
            # Calculate dynamic thresholds (for numeric features)
            if feature in input_data.columns:
                feature_values = input_data[feature]

                # Calculate the threshold based on the 75th percentile of the feature values
```

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

```
            threshold_value = np.percentile(feature_values, threshold_factor * 100)
            condition = f"{feature} > {threshold_value:.2f}"

            # Create a more descriptive message based on the attack
            msg = f"Possible {attack_class} attack detected: {feature} exceeds threshold"

            # Add the rule in Snort-like format with the dynamic
            rules.append(f"alert tcp any any -> any any (msg:\"{msg}\"; {condition};)")

    else:
        # If the prediction is 0 (normal), no attack detected, return a message instead
        rules.append("No attack detected; normal traffic observed.")

    return rules

# Rule generation for specific attack classes
def generate_dos_rules(input_data, threshold_factor):
    # Denial of Service (DoS) attacks: High `src_bytes` and `dst_bytes`
    dos_rules = []
    if 'src_bytes' in input_data.columns:
        src_bytes_value = np.percentile(input_data['src_bytes'], threshold_factor * 100)
        dos_rules.append(f"alert tcp any any -> any any (msg:\"Possible DoS attack detected: src_bytes exceeds threshold\"; src_bytes > {src_bytes_value:.2f};)")

    if 'dst_bytes' in input_data.columns:
        dst_bytes_value = np.percentile(input_data['dst_bytes'], threshold_factor * 100)
        dos_rules.append(f"alert tcp any any -> any any (msg:\"Possible DoS attack detected: dst_bytes exceeds threshold\"; dst_bytes > {dst_bytes_value:.2f};)")

    return dos_rules
```

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

```
def generate_probe_rules(input_data, threshold_factor):
    # Probe attacks: High number of failed login attempts, scanning activity
    probe_rules = []
    if 'num_failed_logins' in input_data.columns:
        failed_logins_value = np.percentile(input_data['num_failed_logins'], threshold_factor * 100)
        probe_rules.append(f"alert tcp any any -> any any (msg:\"Possible Probe attack detected: num_failed_logins exceeds threshold\"; num_failed_logins > {failed_logins_value:.2f};)")
    return probe_rules


def generate_r2l_rules(input_data, threshold_factor):
    # Remote to Local (R2L): Attempted remote access to a local machine
    r2l_rules = []
    if 'num_access_files' in input_data.columns:
        access_files_value = np.percentile(input_data['num_access_files'], threshold_factor * 100)
        r2l_rules.append(f"alert tcp any any -> any any (msg:\"Possible R2L attack detected: num_access_files exceeds threshold\"; num_access_files > {access_files_value:.2f};)")
    return r2l_rules


def generate_u2r_rules(input_data, threshold_factor):
    # User to Root (U2R): Elevated privileges attempts (e.g., root access)
    u2r_rules = []
    if 'num_root' in input_data.columns:
        root_access_value = np.percentile(input_data['num_root'], threshold_factor * 100)
        u2r_rules.append(f"alert tcp any any -> any any (msg:\"Possible U2R attack detected: num_root exceeds threshold\"; num_root > {root_access_value:.2f};)")
    return u2r_rules


# Define the endpoint for predictions
@app.route('/predict', methods=['POST'])
def predict():
```

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

```
try:
    # Get the data from the POST request
    rawData = request.data.decode('utf-8')  # Decoding the byte data to string
    data = preprocess_nids_log(rawData)

    # Convert the data to a DataFrame (assuming it is a single row of data for prediction)
    input_data = pd.DataFrame([data])

    # Preprocess the input data similarly to your training process
    categorical_cols = ['protocol_type', 'service', 'flag']
    if 'protocol_type' in input_data.columns:
        input_data['protocol_type'] = protocol_type_encoder.transform(input_data['protocol_type'])
    if 'service' in input_data.columns:
        input_data['service'] = service_encoder.transform(input_data['service'])
    if 'flag' in input_data.columns:
        input_data['flag'] = flag_encoder.transform(input_data['flag'])

    # Handle missing values for categorical columns
    input_data = input_data.fillna('unknown')

    # Scale the features using the loaded scaler
    input_data_scaled = scaler.transform(input_data)

    # Make a prediction using the trained model
    probabilities = rf_model.predict_proba(input_data_scaled)
    prediction = 1 if probabilities[0][1] > 0.5 else 0  # Use probabilities to determine prediction

    # Determine the attack class (assuming the model gives the class name or label)
    attack_class = "Unknown"  # Default to 'Unknown'
```

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VIII August 2025

```
        if prediction == 1:
            # If prediction is 1 (attack), you need to map the prediction to the corresponding attack class
            # This could come from your model's output or be manually mapped
            attack_class = "DoS"  # Example, replace with your model's output for class name

        # Generate rules dynamically based on feature importances
        rules = []
        if hasattr(rf_model, "feature_importances_"):
            feature_importances = rf_model.feature_importances_
            feature_names = input_data.columns
            rules = generate_rules(feature_names, feature_importances, input_data, prediction, attack_class)

        # Combine prediction, probabilities, and rules in the response
        response = {
            "prediction": int(prediction),  # Convert prediction to integer (binary classification)
            "probability": probabilities[0][1],
            "suggested_rules": rules
        }

        return jsonify(response)

    except KeyError as e:
        return jsonify({"error": f"Missing or invalid field: {str(e)}"}), 400
    except Exception as e:
        # If an error occurs, return a message with the error details
        return jsonify({"error": str(e)}), 500

if __name__ == '__main__':
    # Run the Flask app
    app.run(debug=True)
```