Performance Comparison of Traditional Machine Learning and Ensemble Models in Network Intrusion Detection

MINOR PROJECT - 2, EVEN SEM – 2025

Enrolment Numbers - 22103183, 22803030, 22803029

Name of Students - Ravi Raushan Vishwakarma, Viyom Shukla, Abhijeet Kumar

Name of Supervisor - Dr. Meenal Jain

May - 2025

Submitted in partial fulfillment of the Degree of

Bachelor of Technology And 5 Year Dual Degree Programme

Bachelor of Technology

In

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

(I)

TABLE OF CONTENTS

(II)

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place:
Date:

Student Id	Student Name	Student Signature
22103183	Ravi Raushan Vishwakarma	
22803030	Viyom Shukla	
22803029	Abhijeet Kumar	

(III)

CERTIFICATE

This is to certify that the work titled "**Performance Comparison of Traditional Machine Learning and Ensemble Models in Network Intrusion Detection**" submitted by **Ravi Raushan Vishwakarma, Viyom Shukla, and Abhijeet Kuma**r in partial fulfillment for the award of the degree of **B. Tech** of Jaypee Institute of Information Technology, Noida has been carried out under my supervision.

This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor …….

Name of Supervisor - Dr. Meenal Jain

Designation - ASSISTANT PROFESSOR (SENIOR GRADE)

Date - May 06, 2025

(IV)

ACKNOWLEDGEMENT

We would like to place on record our deep sense of gratitude to **Dr. Meenal Jain**, ASSISTANT PROFESSOR (SENIOR GRADE) Jaypee Institute of Information Technology, Noida for his generous guidance.

We also wish to extend our thanks to our group members and other classmates for their insightful comments and constructive suggestions to improve the quality of this project work.

Signatures of the Students:

Student Id	Student Name	Student Signature
22103183	Ravi Raushan Vishwakarma	
22803030	Viyom Shukla	
22803029	Abhijeet Kumar	

(V)

SUMMARY

This study compares the performance of traditional machine learning models and ensemble learning models for network intrusion detection using the NSL-KDD dataset. The dataset contains over 113,000 network traffic records labeled as normal or one of four attack types: Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R). A significant challenge in this dataset is class imbalance, with normal and DoS attacks dominating, while R2L and U2R attacks are rare, making detection difficult.

To prepare the data, missing values were handled through imputation, and categorical features such as protocol type, service, and flag were encoded numerically using label and one-hot encoding. The dataset was divided into four binary subsets, each focusing on distinguishing normal traffic from a specific attack type. Recursive Feature Elimination (RFE) was applied to select the most relevant features for each subset, reducing complexity and improving model accuracy. Due to the scarcity of samples in the Probe, R2L, and U2R subsets, synthetic data was generated using Conditional Generative Adversarial Networks (cGANs) to balance the classes and enhance detection.

The study evaluated four traditional models-Support Vector Machine, Decision Tree, Logistic Regression, and Random Forest-against three ensemble models-Random Forest, Gradient Boosting Machine, and XGBoost. Models were trained on the prepared datasets, with synthetic data augmenting the minority classes. Performance was assessed using accuracy, precision, recall, and F1-score.

Results showed that ensemble models generally outperformed traditional ones, especially in detecting rare attacks like R2L and U2R. Ensemble methods such as XGBoost achieved higher recall and F1-scores, indicating better sensitivity and precision balance. The use of synthetic data significantly improved minority class detection, though challenges remain due to extreme scarcity in some classes. Feature selection via RFE effectively reduced dimensionality without sacrificing performance.

In conclusion, combining ensemble learning with targeted feature selection and synthetic data generation offers a robust framework for network intrusion detection. While traditional models perform adequately for common attacks, ensemble methods better handle complex, infrequent attacks. Future research should explore advanced deep learning techniques and improved data augmentation to further enhance detection accuracy for rare attack types. This approach provides a strong foundation for developing adaptive and effective intrusion detection systems capable of addressing evolving cybersecurity threats

CHAPTER 1

INTRODUCTION

1.1 General Introduction

An Intrusion Detection System (IDS) is a critical network security tool designed to monitor and analyze network traffic to identify suspicious or malicious activities that may indicate security breaches or attacks. Specifically, a Network Intrusion Detection System (NIDS) focuses on monitoring network traffic at strategic points within the network infrastructure, such as behind firewalls or in demilitarized zones (DMZ), to detect unauthorized access, denial-of-service (DoS) attacks, port scanning, malware infections, and other threats. NIDS operates by examining data packets for known attack signatures or unusual behavior patterns, alerting administrators when potential threats are detected.

NIDS employs multiple detection methods, primarily signature-based detection, which compares network traffic against a database of known attack patterns, and anomaly-based detection, which identifies deviations from established normal network behavior. While signature-based detection is effective against known threats, it cannot detect new or unknown attacks. Anomaly-based detection, on the other hand, can identify novel threats by learning normal traffic patterns but may suffer from higher false positives. Many modern NIDS combine both methods to improve detection accuracy.

These systems monitor various network protocols such as TCP/IP, HTTP, FTP, DNS, and SMTP, as well as network devices including routers, switches, and firewalls, to detect suspicious activities or configuration changes. Additionally, NIDS can analyze application traffic, operating system communications, and wireless network traffic to provide comprehensive security coverage. Upon detecting suspicious activity, a NIDS generates alerts containing details like attack type, source and destination IP addresses, and timestamps, enabling timely response to potential breaches.

This project leverages the NSL-KDD dataset, a benchmark dataset widely used for evaluating intrusion detection systems. The dataset contains detailed records of network connections, labeled as either normal or one of several attack types, including Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R). A significant challenge presented by this dataset is the imbalance between common and rare attack types, which can hinder the performance of traditional machine learning models.

The NSL-KDD dataset is widely used for developing and evaluating IDS models. It addresses limitations of the older KDD Cup 99 dataset by removing redundant records and providing a balanced set of labeled network traffic data, including normal and multiple attack types. This dataset facilitates consistent benchmarking of

intrusion detection techniques, including machine learning approaches that leverage feature selection and data augmentation to improve detection, especially for rare attack classes.

1.2 Problem Statement

In today's increasingly connected digital environment, network security faces constant and evolving threats from a wide range of cyberattacks. Intrusion Detection Systems (IDS) play a vital role in identifying and mitigating these threats by monitoring network traffic for malicious activities. However, traditional IDS approaches often struggle to accurately detect sophisticated and emerging attacks, especially when dealing with highly imbalanced datasets where certain attack types occur very rarely. This imbalance leads to poor detection rates for critical but infrequent attacks such as Remote-to-Local (R2L) and User-to-Root (U2R), which can cause severe damage if left undetected.

Moreover, the high dimensionality and heterogeneity of network traffic data, including categorical and continuous features, pose significant challenges for machine learning models in terms of computational complexity and detection accuracy. While traditional machine learning models have shown promise in intrusion detection, they often fall short in handling complex patterns and rare attack classes effectively.

Ensemble learning models, which combine multiple base learners to improve predictive performance, have demonstrated superior accuracy and robustness in various classification tasks, including network intrusion detection. However, there is a need for systematic evaluation and comparison of traditional and ensemble models on realistic datasets like NSL-KDD, particularly when augmented with synthetic data to address class imbalance.

Therefore, the problem addressed in this study is to develop and evaluate an effective intrusion detection framework that leverages feature selection and synthetic data generation techniques to enhance the detection of both common and rare network attacks. The goal is to compare the performance of traditional machine learning models and ensemble learning models on balanced and imbalanced subsets of the NSL-KDD dataset, identifying the best approach for accurate and reliable network intrusion detection.

1.3 Significance of the Problem

Intrusion Detection Systems (IDS) play a vital role in modern cybersecurity by continuously monitoring network traffic to identify suspicious or malicious activities that may indicate security breaches or cyberattacks. They serve as an essential layer of defense, especially as cyber threats become increasingly sophisticated and diverse. IDS tools analyze network packets for known attack signatures or detect anomalies that deviate from normal behavior, alerting security teams to potential threats before significant damage occurs. This early

detection capability helps organizations respond promptly to attacks, reducing the risk of data breaches and system compromise

Network-based IDS (NIDS) are strategically placed within the network infrastructure to monitor traffic flowing through critical points, such as behind firewalls or between subnets. By analyzing copies of network packets without interrupting legitimate traffic, NIDS provide broad visibility into network activity, enabling detection of various attacks including denial-of-service (DoS), port scanning, malware propagation, and insider threats. They complement other security measures by catching threats that bypass primary defenses, making them indispensable for comprehensive network protection.

Beyond threat detection, IDS support compliance with regulatory standards by maintaining detailed logs of security incidents, which assist in audits and forensic investigations. While IDS primarily alert administrators, some advanced systems integrate intrusion prevention capabilities to automatically block or mitigate detected threats. Despite their effectiveness, IDS face challenges such as evasion techniques by attackers who use fragmentation, spoofing, or coordinated low-bandwidth attacks to avoid detection

Given the growing sophistication of cyber threats, the ability to detect both known and unknown attacks-including rare and evolving threats-is more important than ever. Effective intrusion detection not only helps prevent immediate harm but also contributes to long-term resilience and trust in digital infrastructure. Without advanced IDS, organizations face increased risks of undetected intrusions, costly breaches, and regulatory penalties.

1.4 Empirical Study

An empirical study of intrusion detection systems (IDS) using the NSL-KDD dataset reveals that ensemble learning models generally outperform traditional single classifiers in detecting network attacks. Several recent works have evaluated various machine learning and deep learning approaches on NSL-KDD and related datasets, focusing on metrics such as accuracy, precision, recall, and F1-score.

One key finding across multiple studies is that ensemble methods-such as Random Forest, Gradient Boosting Machines, and XGBoost-achieve higher overall accuracy and better detection rates for both common and rare attack types compared to traditional models like Support Vector Machines or Decision Trees. These ensemble models benefit from combining multiple learners to improve robustness and reduce false alarms.

Feature selection and data preprocessing are crucial for improving IDS performance. Techniques like Recursive Feature Elimination (RFE), particle swarm optimization, and Chi-square selectors help reduce dimensionality and focus on the most relevant features, enhancing model accuracy and efficiency. Handling class imbalance, a

significant challenge in NSL-KDD due to rare attacks like R2L and U2R, is effectively addressed by synthetic data generation methods such as Conditional Generative Adversarial Networks (cGANs) or by using loss functions like focal loss during training

Comparative analyses using the NSL-KDD dataset have also assessed traditional machine learning models, such as Decision Trees, Random Forest, Logistic Regression, and K-Nearest Neighbors. While Decision Trees showed high accuracy on training data, their performance dropped on testing data, indicating potential overfitting and the need for more robust solutions. These findings reinforce the advantage of ensemble and hybrid models, which not only achieve higher accuracy but also generalize better to unseen data.

Empirical evidence strongly supports the superiority of ensemble and deep learning approaches over traditional methods for network intrusion detection on the NSL-KDD dataset. Effective feature selection, imbalance handling, and hyperparameter tuning are critical components of high-performing IDS models. Future research is encouraged to balance detection accuracy with computational efficiency and to extend evaluation to multi-class attack detection scenarios for more comprehensive security solutions.

1.5 Motivation behind the Project

With the rapid growth of digital networks, cyberattacks have become more frequent and sophisticated, threatening the security of sensitive data and critical systems. Intrusion Detection Systems (IDS) are essential tools that monitor network traffic to identify and alert administrators about suspicious activities. However, traditional IDS methods often struggle to detect rare or new types of attacks because of imbalanced data and irrelevant features, which can lead to missed threats or false alarms.

Machine learning offers powerful tools to improve intrusion detection by learning patterns from data. While traditional models like Support Vector Machines and Decision Trees work well for common attacks, they often fail to detect less frequent or complex ones. Ensemble learning models, which combine multiple algorithms, tend to perform better by increasing accuracy and reducing errors. Still, these models face challenges due to the imbalance of attack samples in datasets.

This project aims to improve intrusion detection by using feature selection to focus on the most important data and synthetic data generation to balance rare attack samples. By comparing traditional and ensemble models on these enhanced datasets, the project seeks to find the best approach for detecting a wide variety of network attacks.

The motivation behind this work is to create more accurate and reliable IDS that can adapt to evolving cyber

threats. Better detection helps protect valuable information and reduces false alarms, making networks safer and easing the burden on security teams. Ultimately, this project contributes to stronger cybersecurity and more trustworthy digital environments.

1.6 Brief Description of Our Solution Approach

Our solution approach for network intrusion detection integrates data preprocessing, feature selection, synthetic data generation, and advanced machine learning models to effectively detect both common and rare cyberattacks using the NSL-KDD dataset.

1. **Data Preprocessing:**
   The NSL-KDD dataset undergoes thorough preprocessing to ensure data quality and suitability for machine learning. This includes handling missing values, removing duplicates, addressing outliers, and managing abnormal data points. Categorical features such as protocol type, service, and flag are converted into numerical form using label encoding and one-hot encoding, making them compatible with ML algorithms. The attack labels are consolidated into five major categories (Normal, DoS, Probe, R2L, U2R) to streamline classification tasks. Feature scaling and normalization may also be applied to ensure all features contribute equally to the model.

2. **Feature Selection:**

Feature selection is crucial for reducing dimensionality, improving computational efficiency, and enhancing model accuracy. Techniques like Recursive Feature Elimination (RFE), Information Gain, Gain Ratio, and Correlation-based Feature Selection (CFS) are commonly used to identify and retain only the most relevant features for intrusion detection. RFE, for example, iteratively removes the least important features based on model performance, resulting in a more compact and effective feature set. This step helps eliminate redundant or irrelevant features that could negatively impact classification accuracy.

3. **Synthetic Data Generation**

To address the significant class imbalance in the NSL-KDD dataset-especially for rare attacks like R2L and U2R-synthetic data generation techniques are employed. Conditional Generative Adversarial Networks (cGANs) or CTGANs are used to create realistic synthetic samples for minority classes, balancing the dataset and improving the model's ability to detect rare attacks. Studies have shown that augmenting datasets with synthetic samples can increase prediction accuracy by up to 8% and improve

other performance metrics compared to models trained only on imbalanced data.

4. Model Training and Comparison:

A variety of traditional machine learning models (such as Support Vector Machine, Decision Tree, Logistic Regression, and Random Forest) and ensemble models (like Random Forest, Gradient Boosting Machine, and XGBoost) are trained on the preprocessed and balanced datasets. Ensemble methods, which combine multiple classifiers, have demonstrated higher accuracy and robustness, often achieving up to 99% accuracy on the NSL-KDD dataset and significantly reducing false alarm rates compared to single classifiers. Cross-validation and parameter tuning are used to optimize model performance.

5. Performance Evaluation:

Model performance is assessed using key metrics: accuracy, precision, recall, and F1-score. Accuracy measures overall correctness, while precision and recall provide insight into the model's ability to correctly identify attacks and minimize false alarms. The F1-score balances precision and recall, making it especially valuable for evaluating models on imbalanced datasets. Comparative analysis of these metrics across different models identifies the most effective approach for network intrusion detection, supporting the deployment of robust and reliable IDS solutions.

This combined approach effectively tackles challenges such as data imbalance and high dimensionality. By integrating preprocessing, feature selection, synthetic data augmentation, and ensemble learning, our solution improves detection accuracy for both frequent and rare attacks, making it well-suited for real-world network intrusion detection systems.

1.7 Comparison of the existing approaches to the problem framed

1. Traditional Methods (Rule-Based and Logistic Regression)

Strengths:

These methods are simple, easy to implement, and provide clear, interpretable results. Rule-based systems are straightforward and work well for known attack signatures. Logistic Regression offers transparency in how features contribute to predictions.

Weaknesses:

They are limited to detecting only known threats or simple data relationships. These methods struggle with

complex or evolving attack patterns and often have higher false positive rates.

How Our Approach is Different:

Instead of relying solely on simple or rule-based methods, our project applies advanced machine learning techniques such as Random Forest and Gradient Boosting. These models can capture complex, non-linear patterns in network data, leading to better detection of sophisticated and previously unseen attacks.

2. Classical Machine Learning Models (SVM, Decision Tree, Random Forest, Gradient Boosting)

Strengths:

These models can handle more complex patterns in data, adapt to various types of attacks, and generally achieve better accuracy than traditional methods. They are flexible and can be tuned for improved performance.

Weaknesses:

They may require careful preprocessing, feature selection, and parameter tuning. Some models, like SVM, can be sensitive to data scaling or class imbalance, while ensemble models can be computationally intensive.

How Our Approach is Different:

Our approach does not rely on a single model. We train and compare multiple machine learning models, including SVM, Random Forest, and Gradient Boosting, on carefully preprocessed and balanced data. This comprehensive comparison allows us to select the best-performing model for each attack type, ensuring robust and reliable intrusion detection.

3. Deep Learning Models (Neural Networks, CNN, LSTM)

Strengths:

Deep learning models can automatically learn highly complex and abstract patterns from large datasets. They have achieved state-of-the-art results in many domains, including intrusion detection.

Weaknesses:

They require large amounts of labeled data and significant computational resources for training and deployment. Deep learning models are often seen as “black boxes,” making their decisions harder to interpret.

How Our Approach is Different:

Our project focuses on advanced yet efficient machine learning models rather than deep learning. By using feature selection and synthetic data generation, we achieve high accuracy without the need for massive datasets

or expensive hardware, making our solution practical for real-world network environments.

4. Data Augmentation and Synthetic Data Generation (SMOTE, GANs)

Strengths:

Techniques like SMOTE and GANs help address class imbalance by generating synthetic samples for minority classes, improving the detection of rare attacks.

Weaknesses:

Synthetic data must be carefully validated to ensure it does not introduce noise or unrealistic patterns, which could harm model performance.

How Our Approach is Different:

We use Conditional GANs (cGANs) to generate high-quality synthetic data specifically for rare attack types in the NSL-KDD dataset. This targeted augmentation balances the dataset and significantly enhances the model's ability to detect rare and complex attacks.

5. Feature Selection and Preprocessing

Strengths:

Feature selection techniques like Recursive Feature Elimination (RFE) reduce dimensionality, improve model interpretability, and speed up training without sacrificing accuracy.

Weaknesses:

Improper feature selection can lead to loss of important information or retain irrelevant features, affecting model performance.

How Our Approach is Different:

We apply RFE separately to each attack subset, ensuring that only the most relevant features are used for each classification task. This tailored approach maximizes detection accuracy and computational efficiency.

Summary:

Our approach combines advanced machine learning models (Random Forest, Gradient Boosting, SVM) with targeted feature selection and synthetic data generation. This enables accurate and efficient detection of both common and rare attacks, avoids the complexity and resource demands of deep learning, and provides a practical, scalable solution for real-world network intrusion detection

CHAPTER- 2

LITERATURE SURVEY

2.1 Summary of Paper Studied

In recent years, advanced machine learning techniques, particularly ensemble models, have demonstrated significant promise in enhancing network intrusion detection systems (IDS). By leveraging sophisticated algorithms, these approaches can analyze vast and complex network traffic data to identify both common and rare cyberattacks with greater accuracy. Through intelligent data preprocessing, feature selection, and the integration of synthetic data, modern IDS can adapt to evolving threats and uncover subtle patterns that traditional methods might overlook. Ensemble learning models, such as Random Forest and Gradient Boosting, combine the strengths of multiple classifiers, resulting in improved detection rates and reduced false alarms. These systems are also highly scalable, making robust network security accessible to organizations of all sizes, including those with limited resources or expertise. Furthermore, by continuously learning from new data, advanced IDS can respond proactively to emerging attack strategies, providing timely alerts and actionable insights for network administrators. While not a replacement for comprehensive cybersecurity policies and expert oversight, machine learning-driven IDS serve as a powerful complement, bridging critical gaps in threat detection and supporting the resilience of modern digital infrastructures.

2.2 Integrated summary of literature studied

Recent studies show that using machine learning, especially ensemble models like Random Forest and Gradient Boosting, has greatly improved the ability to detect network attacks. Researchers found that these advanced models are better than traditional methods at spotting both common and rare types of cyberattacks. The literature also highlights the importance of cleaning the data, choosing the most important features, and creating extra samples for rare attacks to help the models learn better.

Many papers mention that while deep learning models can be very accurate, they often need a lot of data and computer power, which is not always practical. In contrast, ensemble machine learning models are easier to use, work well with smaller datasets, and are more efficient for real-world network security. Overall, the research agrees that combining good data preparation, feature selection, and advanced machine learning models leads to more reliable and effective intrusion detection systems for keeping networks safe.

2.3 Research Paper on Ensemble Technique Based on Network Intrusive Dataset: -

Table(1)

YEAR	AUTHOR	ML MODEL	DATASET	ACCURACY
2022	Lin Z-Z, Pike TD, Bailey MM, Bastian ND [1]	Hypergraph-Based Machine Learning Ensemble Network Intrusion Detection System	CIC-IDS2017- Developed to provide realistic and up-to- date data for training and testing machine learning models for cyber security.	Not specified
2022	Andalib A, Vakili VT [2]	Autonomous Intrusion Detection System Using an Ensemble of Advanced Learners	NSL-KDD , providing a realistic and contemporary network traffic dataset for cyber security research which simulates both normal network traffic and various types of attack.	87.28(KDDTest+) , 76.61 (KDDTest-21)
2023	Zoghi Z, Serpen G [3]	Ensemble Classifier with Balanced Bagging, XGBoost, and RF- HDDT	UNSW-NB15- Created to address the limitations of older datasets like KDDCup'99 by providing a realistic and contemporary network traffic dataset for cyber security research.	Not specified
2023	Sohail et al. [4]	Deep Neural Networks based Meta-Learning for Network Intrusion Detection	Test+ a dataset used for training and evaluating network intrusion detection systems (NIDS)., Test- 21 - Tests generalization to novel attacks	91.6 (Test+), 85.6 (Test-21)

2024	Peihceng Wu, Runze Ma, Teoh Teik Toe [5]	Stacking-Enhanced Bagging Ensemble CNN	DDSM (breast cancer images) Used for training and evaluating models for breast cancer detection using mammogram images.	98.84
2024	Md Mahbubur Rahman et al. [6]	Hybrid Model (SMOTE + XGBoost)	NSL-KDD and CICIDS2017	99.99, 100

CHAPTER 3

REQUIREMENT ANALYSIS AND SOLUTION APPROACH

3.1 Overall description of the project

This project presents a robust **Network Intrusion Detection System (NIDS)** that leverages advanced machine learning techniques to detect various cyber threats using the NSL-KDD dataset. The workflow is systematically structured into several stages, starting with data preprocessing, where missing or null values are handled, and categorical data is transformed using label encoding and one-hot encoding to make it suitable for model training.

After preprocessing, the dataset is split into multiple binary classification tasks—Normal vs U2R, Normal vs R2L, Normal vs Probe, and Normal vs DoS—to target specific intrusion categories. To address the issue of class imbalance and data scarcity in certain categories, Conditional Generative Adversarial Networks (CGANs) are employed to generate realistic synthetic data. This synthetic data is then combined with the original dataset to enhance model robustness.

Next, the project uses Recursive Feature Elimination (RFE) to select the most relevant features, reducing dimensionality and improving model efficiency. The refined dataset is passed into an ensemble learning framework comprising three powerful classifiers: AdaBoost, Gradient Boosting Machine (GBM), and Extreme Gradient Boosting (XGBoost). These models are trained independently to predict intrusion attempts and boost overall classification performance.

Finally, the predictions from each model are evaluated based on accuracy, precision, recall, and other relevant metrics to determine the best-performing approach. The integration of data synthesis, feature selection, and ensemble learning allows the system to effectively detect a wide range of network attacks, making it a reliable and scalable solution for real-world intrusion detection.

3.2 Requirement Analysis

Requirement Analysis for your **Network Intrusion Detection System (NIDS)** project using the NSL-KDD dataset:

1. **Functional Requirements**

- Data Pre-processing:

  Handle missing and null values.

  Apply label encoding and one-hot encoding for categorical features.

- Data Partitioning:

  Split the dataset into subsets for binary classification tasks: Normal vs U2R, Normal vs R2L, Normal vs Probe, and Normal vs DoS.

- Synthetic Data Generation:

  Use Conditional Generative Adversarial Networks (CGAN) to generate synthetic samples for minority classes to handle class imbalance.

- Feature Selection:

  Apply Recursive Feature Elimination (RFE) to identify and retain the most relevant features.

- Model Building:

  Train ensemble models including AdaBoost, Gradient Boosting Machine (GBM), and XGBoost.

- Performance Evaluation:

  Evaluate each model using performance metrics such as accuracy, precision, recall, and F1-score

2. Non-Functional Requirements

- Scalability:

  The system should be able to handle large-scale network traffic data efficiently.

- Accuracy:

  The system must provide high accuracy in detecting different types of network intrusions.

- Reliability:

  The solution should maintain consistent performance across all data partitions and scenarios.

- Maintainability:

  The codebase should be modular and easy to maintain or extend for future improvements.

- Usability:

  The framework should be user-friendly and provide clear insights into model performance

3. Hardware Requirements

- Minimum 8 GB RAM and a multi-core processor.
- GPU (optional but recommended) for training deep learning models like CGAN.
- At least 2–4 GB of free disk space.

4. Software Requirements

- Programming Language: Python 3.x
- Libraries & Tools:
    - Pandas, NumPy (data manipulation)
    - Scikit-learn (preprocessing, feature selection, model training)
    - TensorFlow/Keras or PyTorch (for CGAN implementation)
    - Matplotlib/Seaborn (for visualization)
- IDE: Jupyter Notebook or any Python-supported IDE

3.3 Solution Approach

The suggested approach aims to create a strong Network Intrusion Detection System (NIDS) that not only identifies different types of attacks but also assesses the efficacy of traditional machine learning models alongside ensemble learning methods. The procedure starts with comprehensive data preprocessing, where missing or null values are appropriately addressed, and categorical features are converted utilizing label encoding and one-hot encoding. This guarantees that the dataset is organized and prepared for training.

Following the preprocessing stage, the dataset is segmented into four separate binary classification tasks: Normal vs U2R, Normal vs R2L, Normal vs Probe, and Normal vs DoS. This division enables the models to concentrate on distinguishing between normal behavior and specific attack types, enhancing detection precision. Given the class imbalance found in some of these categories, particularly the infrequent attack types, a Conditional Generative Adversarial Network (CGAN) is utilized to produce synthetic data, augmenting the dataset and aiding models in identifying more effective patterns.

After integrating the authentic and synthetic data, Recursive Feature Elimination (RFE) is implemented to decrease dimensionality while keeping only the most significant features. The optimized data is subsequently supplied to a collection of machine learning models. To evaluate model performance, both traditional models and ensemble models are developed. The traditional models serve as baselines, whereas ensemble techniques like AdaBoost, Gradient Boosting Machine (GBM), and XGBoost are utilized to investigate enhancements in performance through model boosting and aggregation.

Ultimately, the forecasts generated by all models are assessed using standard performance metrics such as accuracy, precision, recall, and F1-score. The outcomes are compared to underscore the variations in effectiveness between traditional and ensemble methods. This organized methodology guarantees a fair comparison and aids in illustrating the advantages of ensemble models for addressing complex intrusion detection challenges.

CHAPTER 4

MODELING AND IMPLEMENTATION DETAILS

4.1 Design Diagrams

4.1.1 Use case Diagram

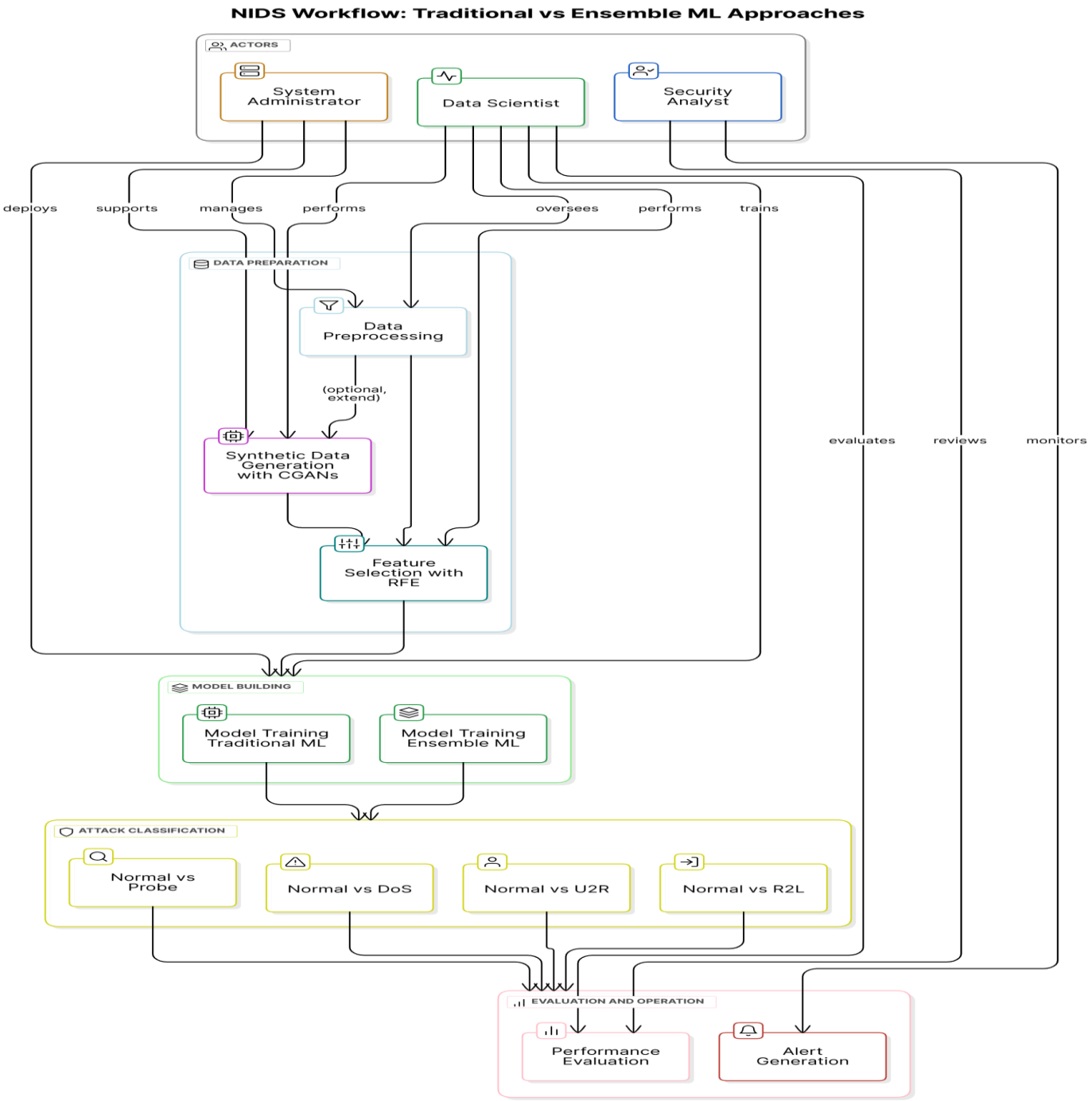

Fig.I Use case Diagram

4.1.2 Class Diagrams / Control Flow Diagrams

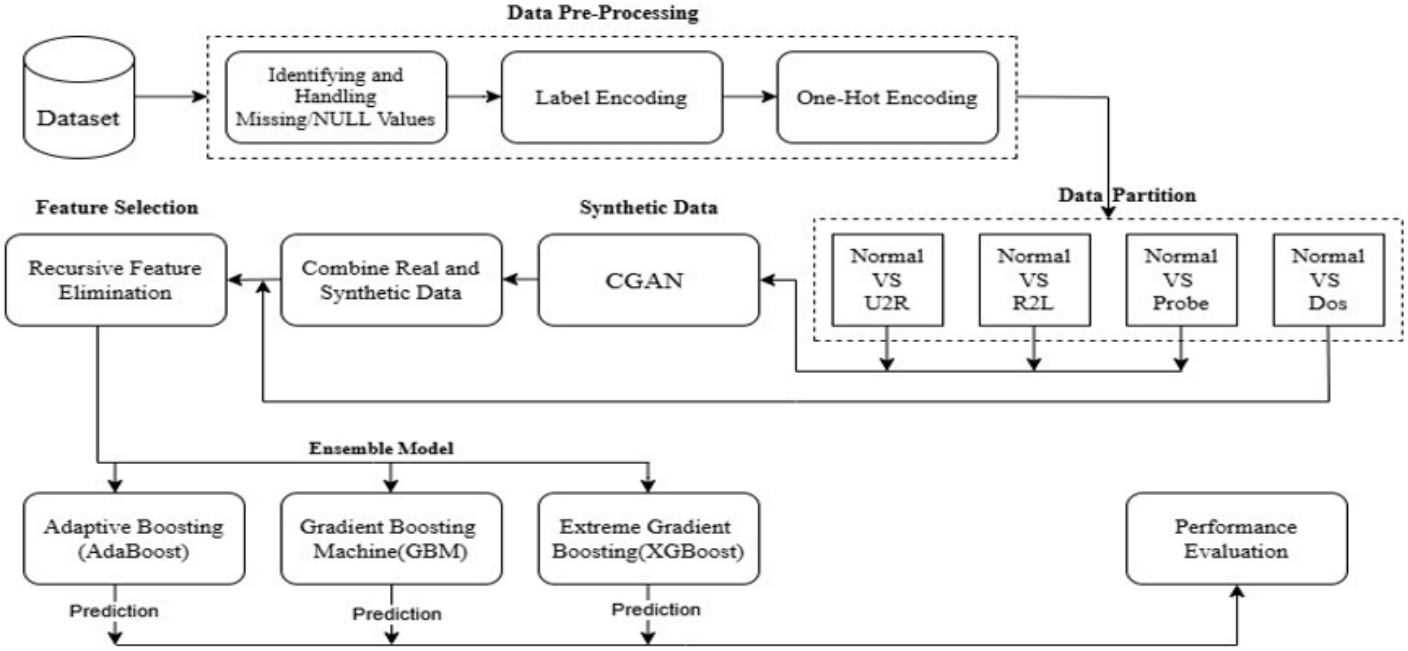

Fig.II Control Flow Diagrams

4.1.3 Sequence Diagram / Activity Diagram

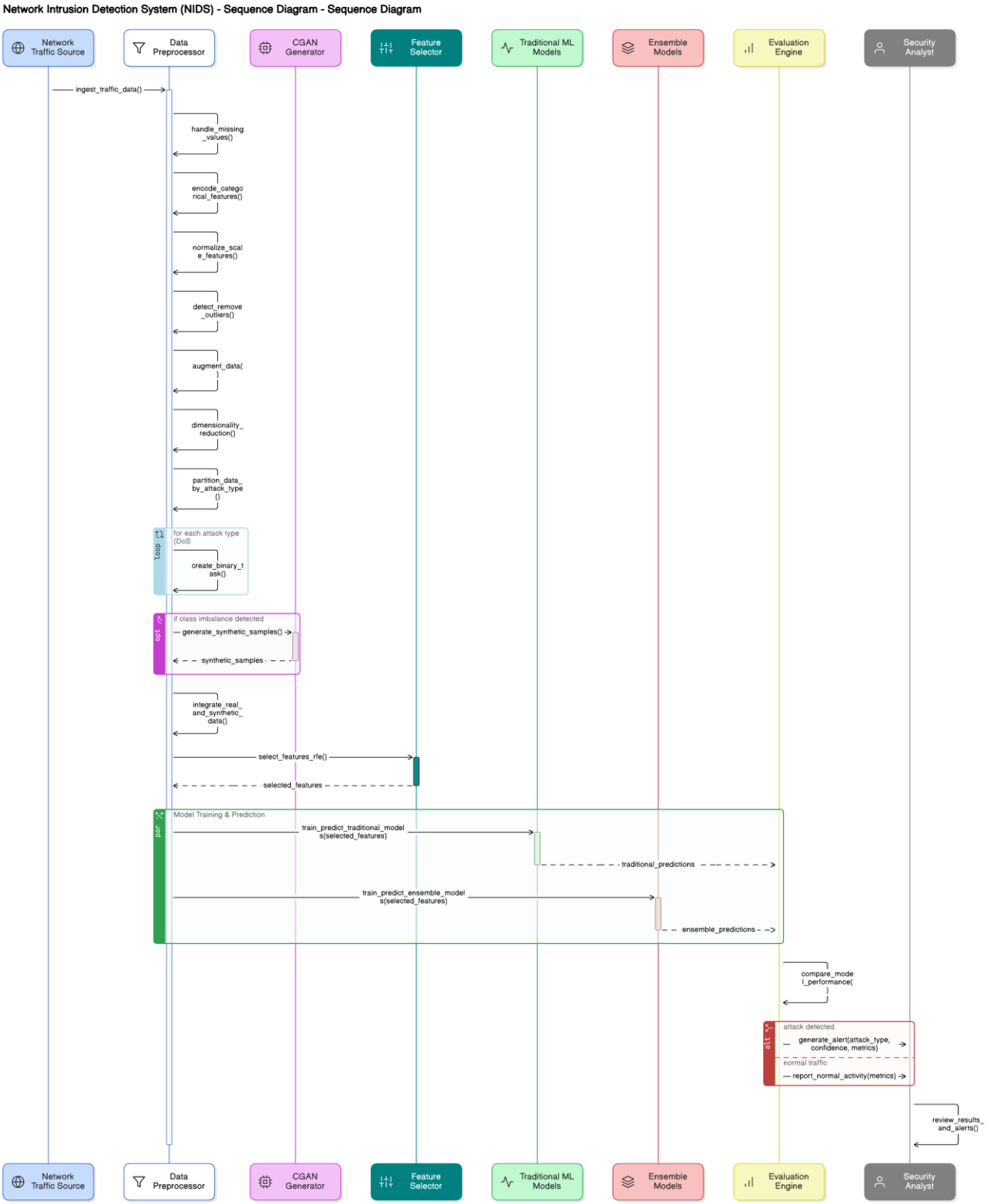

Fig.III Sequence Diagram

4.2 Implementation Details and Issues

4.2.1 Implementation Details

A Dataset Selection and Preparation

I. Dataset Selection

The NSL-KDD dataset is selected for this project as it addresses some of the inherent problems in the original KDD'99 dataset. The dataset contains 41 features and includes various attack types categorized into four main classes:

- Denial of Service (DoS): Attacks that attempt to shut down a machine or network, making it inaccessible to intended users.
- Remote to Local (R2L): Unauthorized access from a remote machine to gain local user privileges.
- User to Root (U2R): Unauthorized access to local superuser (root) privileges.
- Probing: Surveillance and gathering information to identify potential vulnerabilities.

II. Dataset Characteristics and Challenges

- High dimensionality: 41 features requiring dimensionality reduction techniques
- Mixed data types: Both categorical and numerical features necessitating specialized pre-processing
- Class imbalance: Especially pronounced in U2R and R2L attack types, which are significantly underrepresented
- Redundant features: Several highly correlated features that may not contribute additional information

B Data Pre-processing Implementation

I. Data Cleaning

- Missing values are minimal in NSL-KDD but some features may contain outliers that require special handling
- Statistical methods like Z-score are used to identify and handle outliers
- Duplicate records are removed to prevent model bias

II. Feature Engineering

- Protocol type, service, and flag: Categorical features requiring encoding
- Duration, src_bytes, dst_bytes: Numerical features with wide ranges requiring normalization
- New derived features: Creation of feature ratios (e.g., src_bytes/dst_bytes) to enhance detection capabilities

III. Feature Transformation

- Categorical encoding: One-hot encoding for categorical features with many classes (e.g., 'service') and label encoding for binary categorical features
- Numerical scaling: Min-Max normalization to bring all features to a1 range
- Log transformation: Applied to highly skewed numerical features to make their distribution more symmetric

C Handling Class Imbalance with CGAN

I. CGAN Architecture

- Generator: Multi-layer neural network that generates synthetic samples conditioned on the attack class
- Discriminator: Binary classifier that distinguishes between real and generated samples while also considering the class condition
- Training process: Alternating between training the discriminator and generator, with the generator learning to produce increasingly realistic samples

II. Implementation Challenges

- Mode collapse: The generator may produce limited varieties of samples
- Training instability: GANs are notoriously difficult to train and may require careful hyperparameter tuning
- Quality assessment: Ensuring that synthetic samples maintain the statistical properties of real attack vectors

- Computational cost: Training GANs requires significant computational resources, especially with high-dimensional data

III. `Synthetic Data Integration

- Validation: Statistical tests to verify that synthetic samples resemble real data
- Mixing strategy: Careful balance between real and synthetic samples to avoid biasing the models
- Cross-validation: Special care when using synthetic data to avoid data leakage between training and validation sets

D Feature Selection with Recursive Feature Elimination

I. RFE Implementation

- Base estimator choice: Random Forest provides reliable feature importance scores
- Cross-validation strategy: k-fold cross-validation (k=5) to determine optimal feature subset
- Step size: Features are removed iteratively, with smaller step sizes for fine-grained control

II. Feature Selection Challenges

- Feature interactions: Important features in combination may be missed when evaluated individually
- Stability: Different runs of RFE might select different feature subsets
- Computational overhead: The recursive nature of RFE makes it computationally expensive

III. Selected Features

After applying RFE, the most important features for each attack category are identified:

- DoS: Features related to connection statistics and traffic volume
- Probe: Features related to scanning activities and connection attempts
- R2L: Features related to login attempts and data transfer patterns
- U2R: Features related to privilege escalation indicators

E Model Implementation

I. Traditional Machine Learning Models

- Logistic Regression: Simple baseline model with good interpretability
- Decision Tree: Captures non-linear relationships with interpretable decision rules
- Support Vector Machine: Effective for high-dimensional spaces with kernel trick for non-linear decision boundaries
- Random Forest: Ensemble of decision trees with reduced overfitting

II. Ensemble Learning Implementation

- AdaBoost: Focuses on difficult-to-classify samples by adjusting sample weights
- Gradient Boosting Machine (GBM): Sequentially builds trees to correct errors of previous ones
- XGBoost: Advanced implementation of gradient boosting with regularization
- Stacking Ensemble: Meta-learner trained on the predictions of base models

III. Model Training Challenges

- Hyperparameter tuning: Grid search with cross-validation is computationally expensive
- Overfitting: Especially with complex models on limited data
- Model complexity vs. performance: Balancing model complexity with inference speed

D Performance Evaluation Implementation

I. Cross-Validation Strategy

- Stratified k-fold cross-validation to maintain class distribution across folds
- Performance metrics calculated on each fold and averaged
- Standard deviation of metrics used to assess model stability

II. Performance Metrics

- Accuracy: Overall correctness of classification
- Precision: Ability to avoid false positives (critical for operational use)

- Recall: Ability to detect all attacks (critical for security)
- F1-score: Harmonic mean of precision and recall
- AUC-ROC: Measures discrimination ability across different thresholds
- Detection Rate and False Alarm Rate: Security-specific metrics

III. Result Visualization

- ROC curves for comparing models
- Confusion matrices for detailed error analysis
- Feature importance plots for model interpretability

E Implementation Issues and Challenges

I. Technical Challenges

- Class imbalance: Even with synthetic data generation, extreme imbalance in U2R attacks remains challenging
- Feature extraction: Some important features may require domain expertise to engineer properly
- Computational resources: Training multiple models, especially with hyperparameter tuning, requires significant computational power
- GAN instability: CGANs may require extensive tuning to generate high-quality synthetic samples

II. Performance Challenges

- False positives: High false positive rates can lead to alert fatigue in operational environments
- Adaptability: Models may perform poorly on novel attack vectors not represented in the training data
- Real-time processing: Ensuring the system can process network traffic in real-time with acceptable latency

III. Integration Challenges

- Model deployment: Converting trained models into a production-ready system
- Streaming data processing: Adapting batch-trained models to streaming network data
- Alert management: Prioritizing and aggregating alerts to prevent overwhelming security analysts

F Future Improvements

I. Advanced Techniques

- Deep learning models: Exploring CNN, RNN, or transformer architectures for sequence-based detection
- Online learning: Updating models incrementally as new labeled data becomes available
- Self-supervised learning: Leveraging unlabeled data to improve feature representations

II. System Extensions

- Explainable AI: Enhancing model interpretability for security analysts
- Concept drift detection: Identifying when the underlying data distribution changes, requiring model updates
- Threat intelligence integration: Incorporating external threat feeds to enhance detection capabilities

4.2.2 ML Model Accuracy on Real Data

```
Train:
Dimensions of DoS: (113270, 123)
Dimensions of Probe: (78999, 123)
Dimensions of R2L: (68338, 123)
Dimensions of U2R: (67395, 123)

Test:
Dimensions of DoS: (17171, 123)
Dimensions of Probe: (12132, 123)
Dimensions of R2L: (12596, 123)
Dimensions of U2R: (9778, 123)
```

Fig.IV Dimensions of Dataset

```
epochs = 5000
batch_size = 32
sample_interval = 1000

num_samples = data_label4_scaled.shape[0]

for epoch in range(epochs):



    idx = np.random.randint(0, num_samples, batch_size)
    real_data = data_label4_scaled[idx]
    real_labels = np.full((batch_size, 1), 4)


    noise = np.random.normal(0, 1, (batch_size, latent_dim))
    fake_labels = np.full((batch_size, 1), 4)
    gen_data = generator.predict([noise, fake_labels], verbose=0)


    d_loss_real = discriminator.train_on_batch([real_data, real_labels], np.ones((batch_size, 1)))
    d_loss_fake = discriminator.train_on_batch([gen_data, fake_labels], np.zeros((batch_size, 1)))
    d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)



    noise = np.random.normal(0, 1, (batch_size, latent_dim))
    valid_y = np.ones((batch_size, 1))
    g_loss = combined.train_on_batch([noise, fake_labels], valid_y)


    if epoch % sample_interval == 0:
        print(f"Epoch {epoch} [D loss: {d_loss[0]:.4f}, acc: {100*d_loss[1]:.2f}%] [G loss: {g_loss:.4f}]")
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/backend/tensorflow/trainer.py:82: UserWarning: The model does not have any trainable weights.
  warnings.warn("The model does not have any trainable weights.")
Epoch 0 [D loss: 0.6932, acc: 72.66%] [G loss: 0.6923]
Epoch 1000 [D loss: 0.6938, acc: 47.76%] [G loss: 0.6915]
Epoch 2000 [D loss: 0.6938, acc: 47.74%] [G loss: 0.6915]
Epoch 3000 [D loss: 0.6938, acc: 47.73%] [G loss: 0.6914]
Epoch 4000 [D loss: 0.6938, acc: 47.75%] [G loss: 0.6914]
```

```
import matplotlib.pyplot as plt


epochs = [0, 1000, 2000, 3000, 4000]


generator_losses = [0.6923, 0.6915, 0.6915, 0.6914, 0.6914]
discriminator_losses = [0.6932, 0.6938, 0.6938, 0.6938, 0.6938]


discriminator_acc = [72.66, 47.76, 47.74, 47.73, 47.75]


plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.plot(epochs, generator_losses, label="Generator Loss", color='blue', marker='o')
plt.plot(epochs, discriminator_losses, label="Discriminator Loss", color='red', marker='s')
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Generator and Discriminator Loss")
plt.legend()
plt.grid(True)


plt.tight_layout()
plt.show()
```

```
import matplotlib.pyplot as plt


epochs = [0, 1000, 2000, 3000, 4000]


generator_losses = [0.6923, 0.6915, 0.6915, 0.6914, 0.6914]
discriminator_losses = [0.6932, 0.6938, 0.6938, 0.6938, 0.6938]


discriminator_acc = [72.66, 47.76, 47.74, 47.73, 47.75]


plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.plot(epochs, generator_losses, label="Generator Loss", color='blue', marker='o')
plt.plot(epochs, discriminator_losses, label="Discriminator Loss", color='red', marker='s')
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Generator and Discriminator Loss")
plt.legend()
plt.grid(True)


plt.tight_layout()
plt.show()
```

Fig .V CGAN code snippet

Generator vs Discriminator Loss

R2L VS Normal

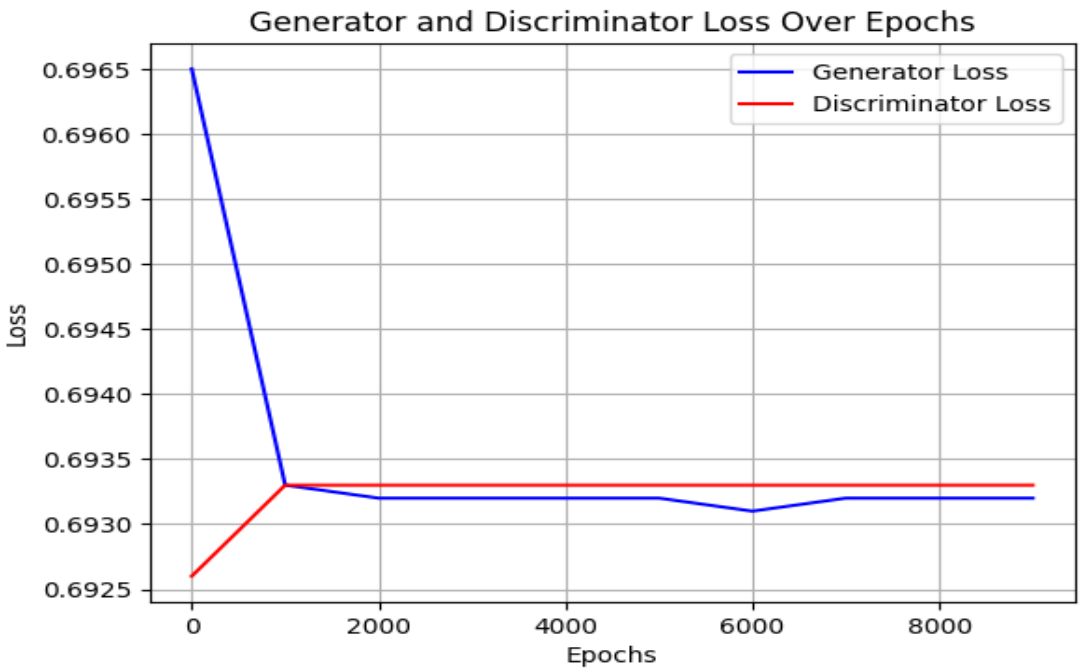

Fig.VI R2L VS Normal

Probe vs Normal

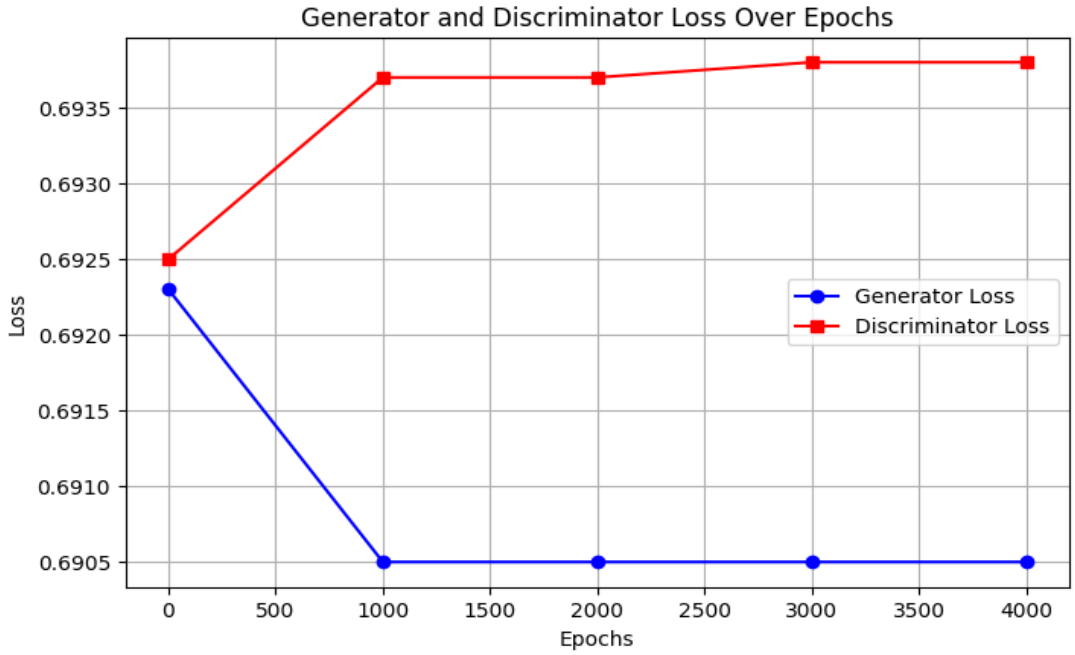

Fig.VII Probe vs Normal

U2R vs Normal

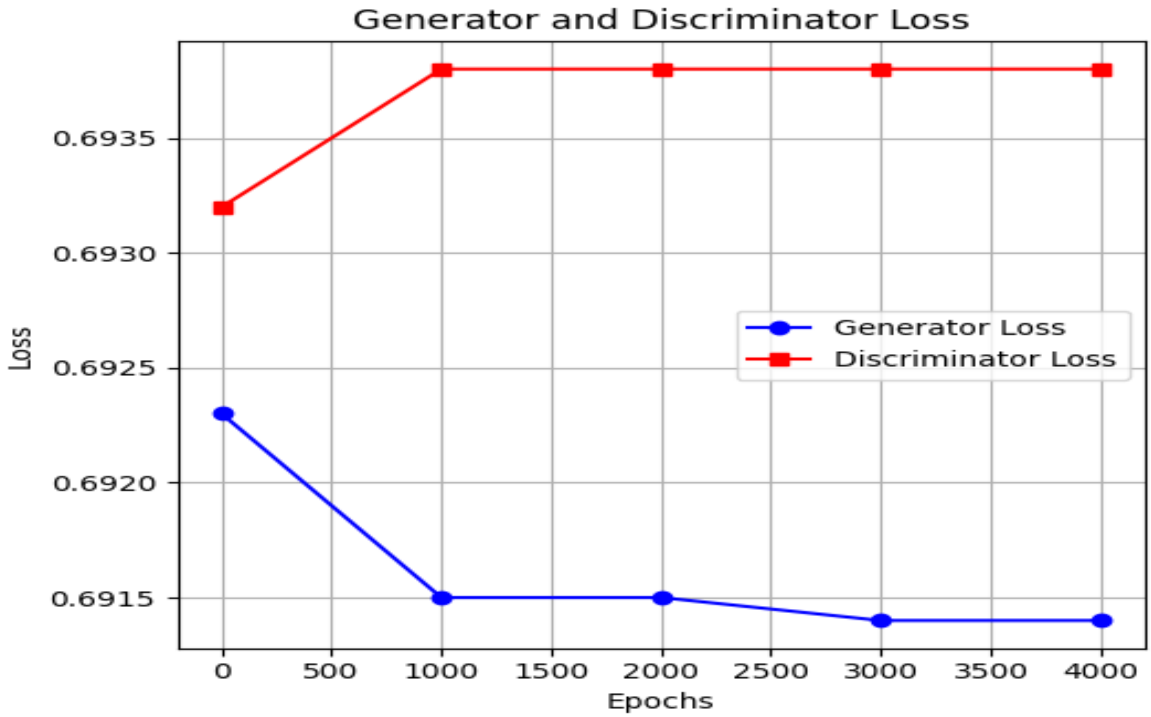

Fig. VIII U2R vs Normal

4.2.3 CGAN Model and Code Snippet

Traditional-Based ML Models (Accuracy)

Table(II)

	Logistic Regression	**Decision Tree**	**Random Forest**	**SVM**
DOS	0.8833	0.8737	0.8740	0.8728
Probe	0.8845	0.8833	0.8863	0.8909
R2L	0.7710	0.7778	0.7779	0.7710
U2R	0.9936	0.9950	0.9950	0.9954

Ensemble-Based ML Model (Accuracy)

Table(III)

	Gradient Boosting	Adaboost	XGboost
DOS	0.8725	0.8741	0.8726
Probe	0.8941	0.8946	0.8839
R2L	0.7776	0.7707	0.7800
U2R	0.9950	0.9935	0.9949

ROC Curve

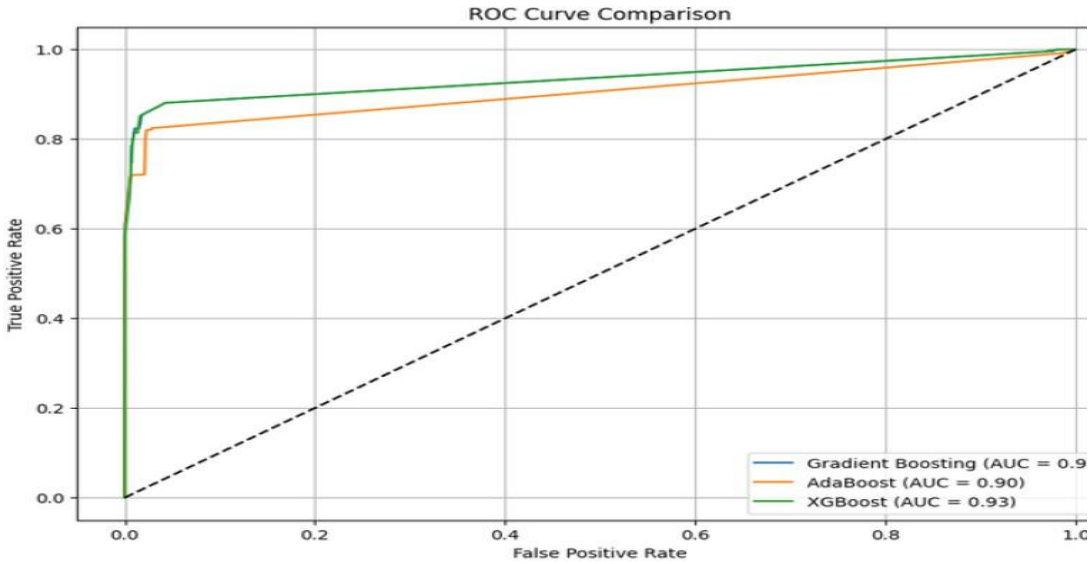

Fig.IX Dos Dataset(ROC Curve)

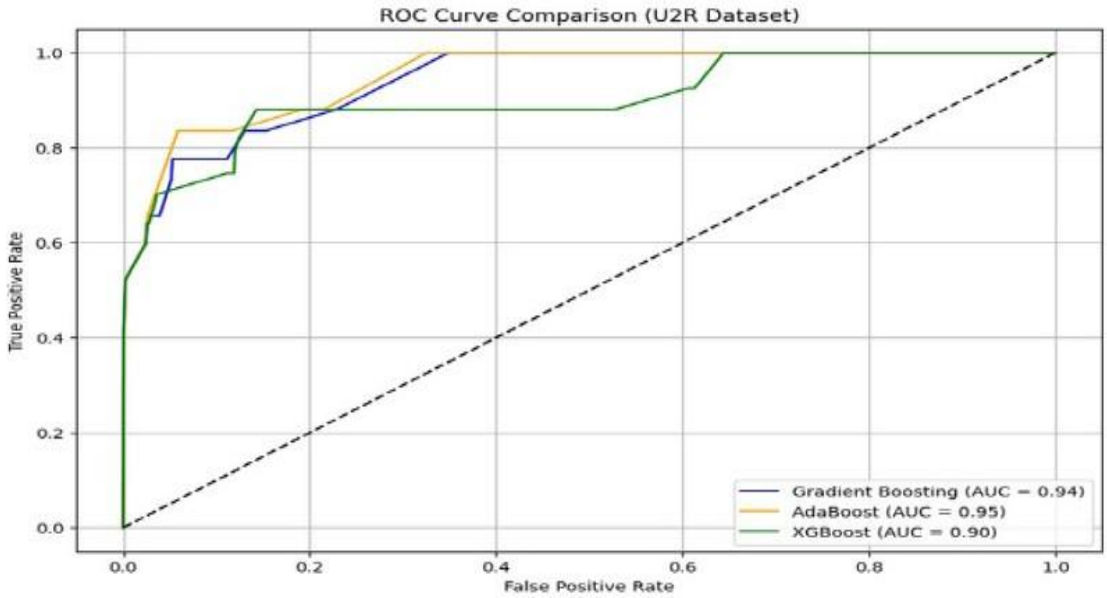

Fig .X U2R Dataset(ROC Curve)

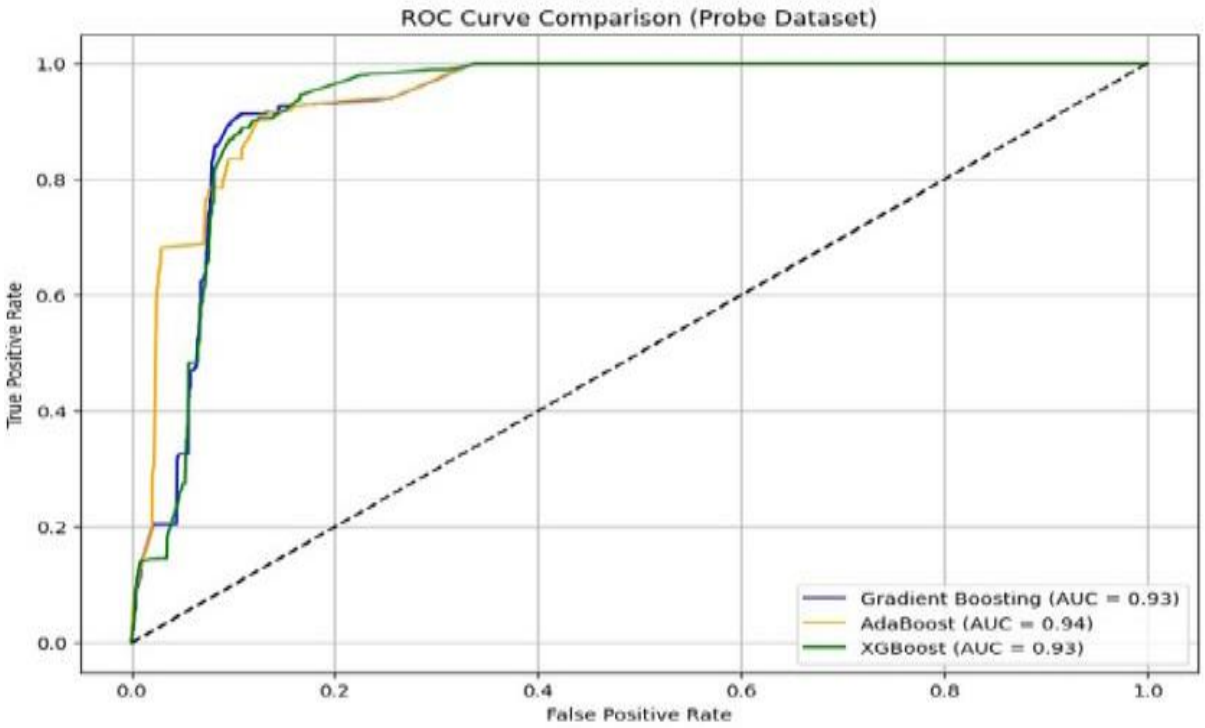

Fig.XI Probe Dataset(ROC Curve)

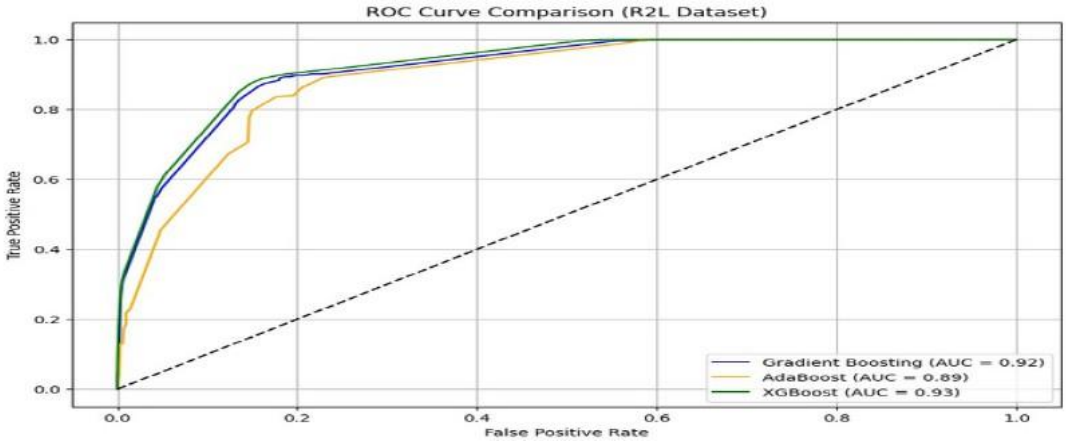

Fig.XII R2L Dataset (ROC Curve)

CHAPTER 5

TESTING (FOCUS ON QUALITY OF ROBUSTNESS AND TESTING)

5.1 Testing Plan

The primary goal of testing was to evaluate the Robustness, Accuracy, and Reliability of AdaBoost, Gradient Boosting, and XGBoost models on the NSL-KDD dataset for detecting different types of network intrusions (DoS, Probe, U2R, R2L).

1. **Dataset Preparation**

- Dataset: NSL-KDD
- Preprocessing steps:
  - Removal of redundant and null data
  - Label encoding for categorical features
  - Feature scaling (standardization)
  - Stratified sampling for balanced train-test split

2. **Testing Strategy**

A multi-phase testing strategy was followed:

- **Unit Testing:** For data preprocessing modules and model training functions.
- **Integration Testing:** Ensured smooth flow between preprocessing, model training, evaluation, and result visualization.
- **System Testing:** Validated the complete system performance against each attack class.
- **Performance Testing:** Benchmarked accuracy, precision, recall, F1-score, and AUC-ROC

5.2 Component Decomposition & Testing Used

1. **Component Breakdown**

Table(IV)

Component	Description	Testing Used
Data Preprocessing	Cleaning, encoding, scaling	Unit Testing, Data Consistency Checks
Model Training	Training Logistic Regression, Decision Tree, SVM, Random	Integration Testing

	Forest AdaBoost, GBM, XGBoost	
Evaluation Module	Computes accuracy, precision, etc.	Unit Testing
Visualization	ROC curve plotting	Visual Output Verification

2. Testing Techniques

- **Cross-validation**: 5-fold cross-validation for model stability
  - The dataset is split into 5 equal parts (folds). The model is trained on 4 folds and tested on the remaining one. This process is repeated 5 times, with each fold used as the test set once.
  - **Purpose**: Reduces the risk of overfitting and ensures the model performs consistently across different subsets of data.
  - **Benefit**: Gives a more reliable estimate of model performance by averaging results across all folds.
- **Stratified Sampling**: Ensured class representation during train-test splitting
  - The data is split such that each class label (e.g., normal, DoS, probe, etc.) is proportionally represented in both training and testing sets.
  - **Purpose**: Maintains the original class distribution, especially important for imbalanced datasets like NSL-KDD.
  - **Benefit**: Prevents bias toward majority classes and helps in fair evaluation of model performance across all types of attacks.
- **Random Seed Fixation**: To guarantee reproducibility
  - Setting a fixed random seed ensures that the random processes (like data shuffling or model initialization) produce the same results every time.
  - **Purpose**: Makes the experiments reproducible, which is important for consistent comparisons and debugging.
  - **Benefit**: Ensures that results can be replicated in future runs or by other researchers.
- **Exception Testing**: Checked for memory issues, convergence errors
    - The model was tested for robustness by monitoring for:
      - **Memory issues** (e.g., out-of-memory errors on large datasets or hyperparameter tuning)

- **Convergence errors** (e.g., when models fail to converge during training, especially for boosting algorithms)

- **Purpose**: Ensures the system behaves gracefully under abnormal or high-load conditions.
- **Benefit**: Increases reliability and prepares the system for real-world deployment.

➢ **Edge Testing**: Used adversarial samples with missing or extreme values

- Testing the model with:
- **Purpose**: Evaluates how the model handles unexpected or abnormal data, which can occur in real-world network traffic.
- **Benefit**: Enhances model robustness and security, particularly in intrusion detection systems that must handle noisy or malicious data.

5.3 Error & Exception Handling

1. **Common Errors Detected:**
   a. **Data Mismatch Errors:** Handled via validation on input types and formats.
   b. **Memory Overflows:** Managed by reducing feature dimensions and batch training.
   c. **Model Convergence Warnings:** Tuned hyperparameters such as learning rate and estimators.
   d. **Division-by-Zero during metric computation:** Applied conditional exception handling in metric formulas.
2. **Fault Tolerance Mechanisms:**
   a. **ry-except blocks** in Python for managing loading and transformation failures.
   b. **Logging System** to capture and store error messages and tracebacks.

5.4 Limitations

1. Class Imbalance
   a. The dataset exhibited a significant imbalance among different attack categories.
   b. Particularly, the **R2L (Remote to Local)** and **U2R (User to Root)** classes were underrepresented, leading to **low recall and F1-scores** for these categories.
   c. This affected the model's ability to detect rare but critical attack types.
2. Overfitting Risk
   a. Ensemble boosting algorithms, especially when finely tuned, can **overfit** on

small or noise-free datasets.

   b. While performance on the training set was high, slight drops in test performance indicated the model may not generalize well to unseen variations.

3. Computational Cost
   a. Algorithms like **XGBoost** require considerable **CPU and memory resources**, particularly during hyperparameter tuning and training on full-sized dataset.
   b. This limited scalability and increased training time, which could be a concern in resource-constrained environments.
4. Dataset Bias
   a. The **NSL-KDD dataset**, though widely used for benchmarking, does not accurately reflect the characteristics of **modern network traffic**.
   b. As a result, models trained on it may **not generalize effectively** to real-world or real-time attack scenarios.
5. Lack of Online Testing
   a. All testing and validation were conducted on **static, preprocessed datasets.**

CHAPTER 6

FINDING, CONCLUSION AND FUTURE WORK

6.1 Findings

The following performance metrics were obtained from experiments conducted on the four main attack categories in the NSL-KDD dataset. Each boosting algorithm—AdaBoost, Gradient Boosting, and XGBoost—was evaluated using key metrics such as AUC-ROC, F1-score, Recall, and Accuracy.

1. DoS (Denial of Service) Detection
    a. **Best AUC-ROC**: XGBoost achieved an AUC-ROC of 0.93, indicating its strong capability to distinguish DoS attacks from normal traffic.
    b. **Best F1-score**: AdaBoost scored the highest F1-score of 0.87, reflecting a good balance between precision and recall.
    c. **Interpretation**: DoS attacks are well-represented in the dataset and exhibit strong patterns, making them easier for ensemble models to learn and detect reliably.
2. Probe Detection
    a. **Best AUC-ROC**: **AdaBoost** achieved an **AUC-ROC of 0.94**, showing excellent discrimination ability for probing attacks.
    b. **Highest Accuracy**: Also led by **AdaBoost**, with an accuracy of **0.8946**, demonstrating that most predictions (both attack and normal) were correct.
    c. **Interpretation**: Probe attacks tend to have structured, recognizable patterns (like port scanning), which AdaBoost was particularly effective at capturing.
3. U2R (User to Root) Detection
    a. **Best F1-score**: **Gradient Boosting and XGBoost** both achieved an **F1-score of 0.73**, the highest among models tested.
    b. **Best Recall**: **Gradient Boosting** showed superior recall at **0.66**, meaning it successfully identified 66% of actual U2R attacks.
    c. **Interpretation**: Although rare and complex, U2R attacks were best detected by models capable of handling nuanced feature interactions. Gradient Boosting showed a slight

edge in recall, which is crucial for security-focused tasks where false negatives can be dangerous.

4. R2L (Remote to Local) Detection
    a. **Best AUC-ROC**: **XGBoost** again performed best with an **AUC-ROC of 0.93**, suggesting good potential for discrimination.
    b. **Common Issue**: All models suffered from **low F1-scores and recall**, highlighting difficulty in effectively detecting R2L attacks.
    c. **Interpretation**: R2L attacks often mimic normal traffic and occur infrequently in the dataset, making them extremely hard to detect. Despite high AUC scores indicating theoretical separability, real-world performance (F1 and recall) remained poor, showing a gap in practical detection.

6.2 Conclusion

The project successfully demonstrated that ensemble boosting algorithms—AdaBoost, Gradient Boosting, and XGBoost—can effectively detect various network intrusions using the NSL-KDD dataset. These models outperformed many traditional methods in terms of accuracy and AUC-ROC, confirming their suitability for cybersecurity applications. Among the three, XGBoost consistently showed strong overall performance, especially for DoS and R2L attacks.

A notable strength of these boosting algorithms lies in their ability to handle large datasets and focus on hard-to-classify instances. XGBoost, in particular, proved to be the most robust, achieving high AUC-ROC scores across all classes. Gradient Boosting also performed well, especially in detecting low-frequency attacks like U2R, showing better recall than AdaBoost.

Despite the strong performance, certain challenges remain. All models struggled with rare classes like R2L and U2R due to class imbalance. The F1-scores for these attacks were significantly lower, indicating a need for more sophisticated techniques like data augmentation or class-specific tuning to improve detection in such cases.

Another limitation observed was the high computational cost of training ensemble models, especially XGBoost. While it provided excellent performance, it required more time and memory than the other models. This could pose challenges for real-time applications or deployment on resource-limited systems.

In conclusion, ensemble boosting methods are promising tools for intrusion detection, with XGBoost leading in performance. However, to make these systems more reliable and scalable, future improvements should focus on rare attack detection, real-time processing, and model explainability.

With further enhancements, these algorithms can play a critical role in building smarter, more secure networks.

6.3 Future Work

The Network Intrusion Detection System (NIDS) opens numerous avenues for future exploration and development. To enhance the model's impact, several directions can be pursued:

1. Real-Time Detection
    a. Objective: Adapt existing models to handle live network traffic using online learning and streaming frameworks.
    b. Current offline models are limited in practical deployment. Real-time intrusion detection is crucial for immediate threat response in live environments.
    c. Next steps: Integrate with tools like Apache Kafka, Spark Streaming, or use frameworks like River (for online ML in Python).
2. Handling Class Imbalance
    a. Objective: Improve detection of underrepresented classes like R2L and U2R using advanced sampling techniques.
    b. Techniques:
        i. SMOTE (Synthetic Minority Over-sampling Technique): Generates synthetic samples for minority classes.
        ii. ADASYN (Adaptive Synthetic Sampling): Focuses on generating harder-to-learn examples.
    c. Benefit: Enhances model performance for rare but critical attack types, especially in metrics like recall and F1-score.
3. Feature Engineering
    a. Objective: Improve model performance and efficiency by reducing feature space and removing irrelevant/noisy features.
    b. Techniques:
        i. PCA (Principal Component Analysis): Reduces dimensionality while preserving variance.
        ii. Autoencoders: Learn compressed representations of input data using neural networks.
    c. Benefit: Leads to faster training, less overfitting, and potentially better generalization.

4. Hybrid Models
    a. Objective: Enhance sequential and pattern-based attack detection by combining deep learning with boosting.
    b. Approach:
        i. Use CNN (Convolutional Neural Networks) to capture spatial patterns in packet sequences
        ii. Use RNN (Recurrent Neural Networks) or LSTMs to capture temporal dependencies.
        iii. Combine outputs with boosting models (e.g., XGBoost) for final decision-making.
    c. Benefit: Exploits both temporal and feature-based patterns, improving detection of complex attack sequences.
5. Robust Testing Environment
    a. Objective: Test models in realistic network environments using cloud platforms and live traffic datasets.
    b. Implementation:
        i. Deploy models on platforms like AWS, Google Cloud, or Azure.
        ii. Use tools like Wireshark, Zeek, or Suricata to generate and capture real-time network data.
    c. Benefit: Provides a more accurate evaluation of real-world performance, latency, and robustness.

By addressing these areas, the proposed directions aim to bridge the gap between experimental success and real-world applicability. By enhancing model robustness, interpretability, and deployment readiness, this work can evolve into a scalable, intelligent intrusion detection system capable of operating effectively in dynamic network environments.

Reference: -

[1] Lin, Z.-Z., Pike, T. D., Bailey, M. M., & Bastian, N. D. (2024). A Hypergraph-Based Machine Learning Ensemble Network Intrusion Detection System. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 54(11), 6911–6923. doi:10.1109/tsmc.2024.3446635: https://arxiv.org/abs/2211.03933

[2] Andalib, A., & Tabataba Vakili, V. (2020). An Autonomous Intrusion Detection System Using an Ensemble of Advanced Learners. In 2020 28th Iranian Conference on Electrical Engineering (ICEE), pp. 1-6. IEEE. doi:10.1109/ICEE50131.2020.9260808: https://arxiv.org/abs/2001.11936

[3]] Z. Zoghi and G. Serpen, "Ensemble Classifier Design Tuned to Dataset Characteristics for Network Intrusion Detection," arXiv preprint arXiv:2205.06177 [cs.LG], May 2022: https://www.researchgate.net/publication/360559789_Ensemble_Classifier_Design_Tuned_to_Dataset_Characteristics_for_Network_Intrusion_Detection

[4] A. Sohail, B. Ayisha, I. Hameed, M. M. Zafar, and A. Khan, "Deep Neural Networks based Meta-Learning for Network Intrusion Detection,": https://www.researchgate.net/publication/368664875_Deep_Neural_Networks_based_Meta-Learning_for_Network_Intrusion_Detection

[5] P. Wu, R. Ma, and T. T. Toe, "Stacking-Enhanced Bagging Ensemble Learning for Breast Cancer Classification with CNN,": https://arxiv.org/abs/2407.10574

[6] M. M. Rahman, S. A. Shakil, and M. R. Mustakim, "A survey on intrusion detection system in IoT networks," *Computer Science and Applications*, ":https://www.sciencedirect.com/science/article/pii/S2772918424000481