A PROJECT REPORT
on
"Network - Intrusion Detection System"

Submitted to
KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN COMPUTER SCIENCE AND ENGINEERING

BY

Ashlok Prasad Yadav	22054022
Aditya Kumar Thakur	22054270
Samir Kumar	22054355
Yuvraj Singh	22054366
Krish Mahaseth	22054047

UNDER THE GUIDANCE OF

ARADHANA BEHURA

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY,
BHUBANESWAR, ODISHA - 751024

KIIT Deemed to be University
School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

“Network Intrusion Detection System”

Submitted by

Ashlok Prasad Yadav	22054022
Aditya Kumar Thakur	22054270
Samir Kumar	22054355
Yuvraj Singh	22054366
Krish Mahaseth	22054047

- This is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during the year 2024-2025, under our guidance.

Date : 08/04/2025

Project Guide - Aradhana Behura

ACKNOWLEDGEMENTS

- We are profoundly grateful to **Aradhana Behura** of Affiliation for his expert guidance and continuous encouragement throughout to see that this project meets its target since its commencement to its completion.

Ashlok Prasad Yadav
Aditya Kumar Thakur
Samir Kumar
Yuvraj Singh
Krish Mahaseth

ABSTRACT

- This paper proposes the development of a machine learning-based **Network Intrusion Detection System (NIDS)** to identify and classify network-borne malicious activity. Trained and tested on benchmark datasets **CICIDS-2017, CICIDS-2018, and UNSW-NB15** we identified several types of network intrusions using a variety of classification models. The objective is to enhance cybersecurity systems with **supervised learning models** that adapt to evolving threats. Our results form the basis of an intelligent and scalable intrusion detection infrastructure available for real-time deployment.

- In this paper, models such as **Logistic Regression, Random Forest, Support Vector Machine, K-Nearest Neighbors, and Multilayer Perceptron** were developed and tested using performance measures such as **accuracy, precision, recall, and F1-score**. With ample preprocessing and model tuning, the system performed satisfactorily in differentiating normal from malicious traffic patterns. Use of multiple datasets contributes towards power and diversity in detection capability, with a chance for developing more efficient intrusion detection systems.

Keywords:

I. Network-borne malicious activity,
II. Supervised Learning, Cybersecurity,
III. Intrusion Detection, Traffic Patterns.

Content

List Of Figures:

Chapter 1

Introduction:-

- In light of growing frequency and sophistication of **cyber-attacks**, there is an urgent need for next-generation security solutions to safeguard digital infrastructure and confidential data. Among them, **Network Intrusion Detection Systems (NIDS)** have a key function to observe network traffic for indicators of malicious activity. Our **mini project** is focused on the design and testing of a strong NIDS that not only detects suspicious behavior but does so efficiently and accurately. The main objective is to enhance the network security foundation by applying and analyzing intelligent detection techniques.

- Our project will involve the study of various machine learning classification algorithms that can be trained to recognize patterns associated with cyber threats. We are interested in **supervised learning models**, which depend on labeled datasets to differentiate normal from malicious network traffic. Precisely, we use datasets like **CIC-IDS 2017 & 2018 and UNSW NB-15** because they are popularly used in intrusion detection studies for their realistic and complete portrayal of contemporary attack scenarios.

- With the increasing digital interconnectivity of the world today, computer **network security** is now a priority. With the rise in cyber-attacks and their sophistication, conventional security measures such as firewalls and antivirus programs are no longer adequate to **safeguard systems** from emerging threats. This has resulted in greater dependence on smart security solutions such as **Network Intrusion Detection Systems (NIDS)**, which are intended to observe and scan network traffic for indications of malicious behavior.

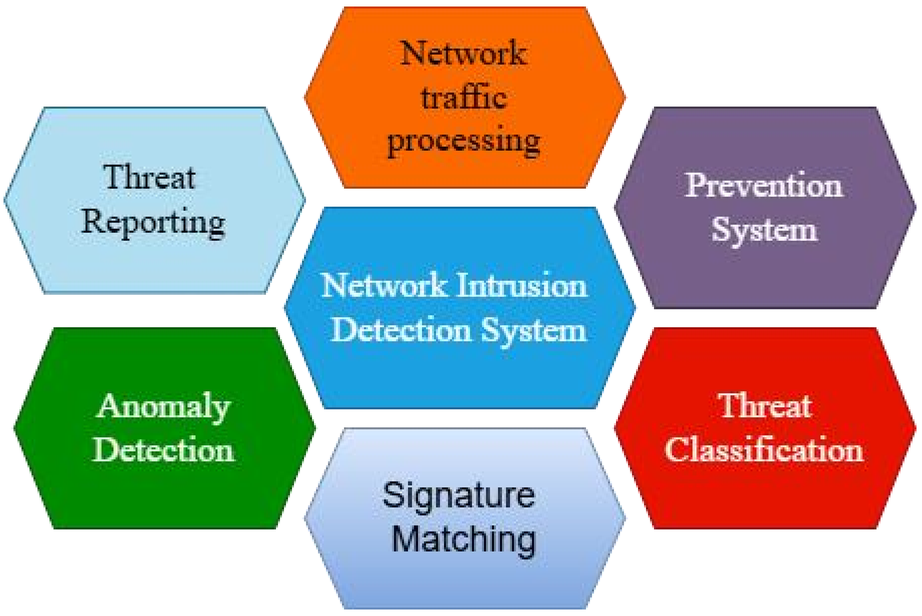


Figure 1.1: Applications of Network Intrusion Detection System.

- The performance of a NIDS relies greatly on how well it can detect intrusions and keep false positives to a minimum. **Machine learning** advancements have introduced new directions in the creation of intelligent NIDS that learn from past patterns of attacks and can adapt to novel, never-before-seen attacks. High-quality, exhaustive datasets are required to train and test these models.

- In this work, we discuss the development and testing of machine learning-based NIDS on three well-known benchmark datasets: **CIC-IDS 2017, CIC-IDS 2018, and UNSW NB-15**. These datasets contain various types of network traffic, both benign and malicious, and represent actual attack scenarios like **Distributed Denial of Service (DDoS)**, brute force, infiltration, and botnet behavior.

- Through experimentation and comparison of various **classification algorithms** on these data sets, we seek to determine their efficiency in identifying diverse network intrusions. This research not only increases knowledge about data-driven methods for **network security** but also adds to the process of developing more accurate, adaptive, and scalable **intrusion detection** systems.

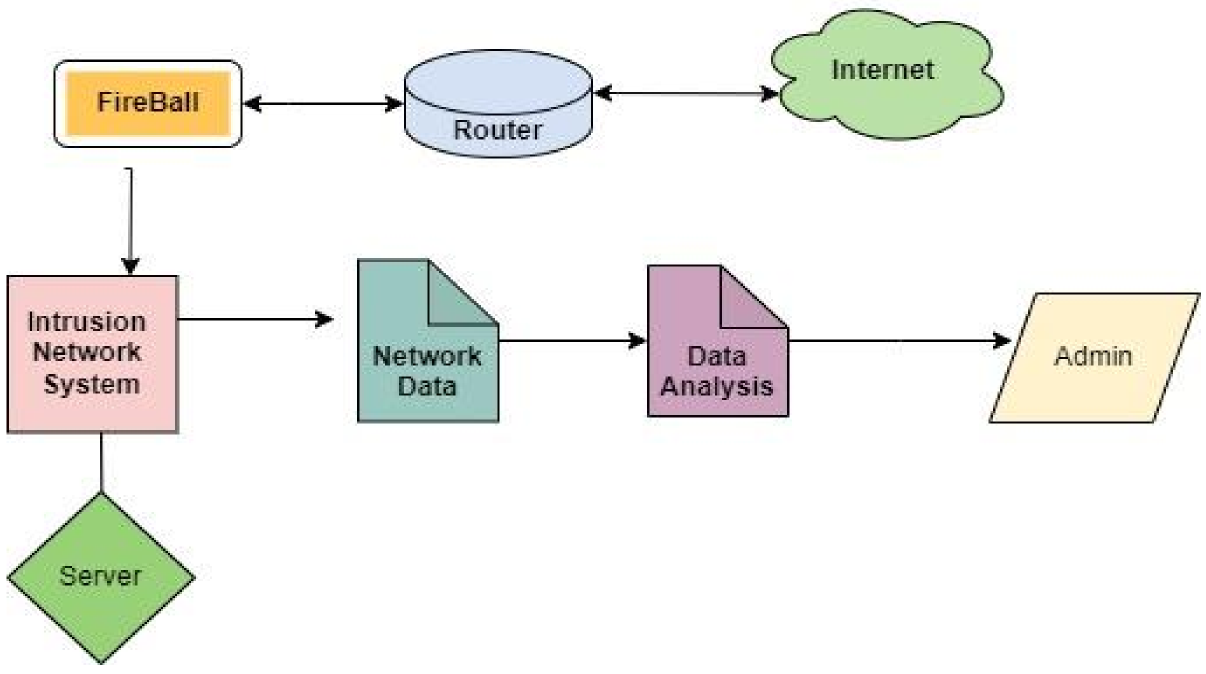


Figure 1.2: Network intrusion detection deployment.

Chapter 2

Literature Review

- This part looks at recent studies that explore ways to make network services more reliable and secure. It focuses on how machine learning and deep learning are being used to better detect threats and handle issues. Studies based on it are:

- Bhawana Sharma et al.(2023) suggested an anomaly-based IDS for IoT networks with a Deep Neural Network (DNN) with filter-based feature selection discarding highly correlated features. It was trained and optimized with the UNSW-NB15 dataset and had an accuracy of 84%. Synthetic minority attack data were generated using GANs to counter class imbalance, and accuracy increased to 91%.[1]

- Yan fang Fu et al.(2022) proposed a network intrusion detection model based on deep learning with CNN, attention, and Bi-LSTM to improve detection precision. The model learns sequence features, adjusts channel weights with attention, and discovers temporal patterns using Bi-LSTM.Sample increasing was performed employing ADASYN for handling class imbalance, and data dimensionality reduction was performed employing a stacked autoencoder with the architecture modified. DLNID was tested on the NSL-KDD dataset and was able to achieve 90.73% accuracy and 89.65% F1-score, which was superior to other solutions.[2]

- Yakub Kayode Saheed et al.(2022) proposed a machine learning-based intrusion detection system (ML-IDS) for attack detection in IoT networks. Min–max normalization was applied to the UNSW-NB15 dataset to prevent information leakage, and subsequently PCA was applied for dimensionality reduction. Six supervised ML models were compared on the basis of metrics such as accuracy, AUC, F1, precision, and MCC. The suggested system provided competitive results with 99.9% accuracy and 99.97% MCC which were better than some existing methods.[4]

- Emad-ul-Haq Qazi et al.(2022) presented a deep learning-based network intrusion detection system (NIDS) utilizing a non-symmetric deep autoencoder to address growing network threats. Deployed with TensorFlow and GPU acceleration, the model was tested on the KDD CUP '99 dataset and produced 99.65% accuracy. Although the system demonstrates robust potential for DL-based network security, its use of the antiquated KDD CUP '99 dataset can potentially restrict its efficiency against current and real-world cyber threats.[5]

- Khushnaseeb Roshan et al.(2024) investigated Machine Learning-based Network Intrusion Detection System (NIDS) vulnerability to adversarial attacks. Four adversarial methods like FGSM, JSMA, PGD, and C&W were used to evaluate the effect of adversarial attacks on NIDS performance. In efforts to enhance system robustness, three defense mechanisms were implemented.Adversarial Training, Gaussian Data Augmentation, and High Confidence. Their research provides proof-of-concept applications in a real-world network setting, providing insights into bolstering the resilience of ML/DL-based NIDS against adversarial attacks.[21]

- Kabir et al. (2020) proposed OA-LS-SVM, a two-stage intrusion detection technique integrating optimal sample allocation and LS-SVM classification. In the first stage, representative samples are chosen on the basis of intra-group variability, followed by intrusion detection based on LS-SVM. On testing on the KDD 99 dataset, the method exhibited high accuracy and efficiency in both binary and multiclass cases. The approach is also promising to be applied in the case of incremental datasets. Still, its operation can be constrained by initial gratuitous subgrouping, which lessens adaptability to dynamic or changing attack patterns in actual implementations.

- Ferrag et al. (2020) gave an extensive survey on deep learning-based intrusion detection systems, classifying 35 popular cybersecurity datasets into seven categories such as IoT, VPN, and network traffic datasets. They compared seven deep models such as CNNs, RNNs, DNNs, and deep autoencoders for binary as well as multiclass classification on the CSE-CIC-IDS2018 and Bot-IoT datasets. Evaluation was done using large performance measures such as accuracy, false alarm rate, and detection rate. Although it reveals valuable information concerning model performance and dataset selection, the work states the challenge in identifying a correct dataset and a deep learning approach for individual intrusion scenarios.

- Yogendra Narayan (2020) presented a comparative study of surface electromyography (sEMG) signal classification to identify six movements of the hand using SVM and Naive Bayes classifiers. Discrete Wavelet Transform (DWT) was utilized to denoise and extract time–frequency domain features, which were combined with time-domain features to form the feature vector. The SVM classifier proved 95.8% accurate, which outpaced the speed and error rate performance of Naive Bayes, and is thus more suitable to use in assistive robotic technology for the elderly and amputees.

- Ke He et al. (2021) gave an overview of DL-based Network Intrusion Detection Systems (NIDS) with emphasis on how susceptible they are to adversarial attacks. DNNs are useful in giving high accuracy, but it is possible that they can be susceptible to slight perturbations on inputs causing misclassification. The paper identifies the inconsistency of employing computer vision-based adversarial attacks on NIDS due to data and detection pipeline variability. It describes attack types, defence mechanisms, and taxonomy of DL-NIDS. One of the primary drawbacks is that there are no standardised, realistic adversarial test datasets, which restricts defence mechanisms testing to real grounds.

- Korium et al. (2023) introduced a machine learning-aided intrusion detection framework specifically for cyberattacks on Internet of Vehicles (IoV) settings to address attacks such as DoS, DDoS, botnets, and sniffing. The system uses Z-score normalization preprocessing, regression-based feature extraction, and high-performance classifiers—Random Forest, XGBoost, CatBoost, and LightGBM with hyperparameter optimization to improve detection accuracy and minimize overfitting. Tested on CIC-IDS-2017, CSE-CIC-IDS-2018, and CIC-DDoS-2019 datasets, both in isolation and combined, the system yielded more than 99.8% accuracy and low detection latency. Its efficacy notwithstanding, testing in real-time dynamic IoV environments with new types of attacks is yet to be established.

- Somanathan Pillai et al. (2023) proposed a sophisticated hybrid Network Intrusion Detection System (NIDS) to meet the challenges brought about by polymorphic malware and encrypted traffic. The proposed model combines Wavelet Transform, LSTM, and ANN for traffic classification with improved preprocessing and feature extraction using the Single Candidate Optimization (SCO) algorithm. The proposed model also employs SCO for the optimization of hyperparameters with outstanding robustness for dynamic cyberattacks. In the evaluation against the BoT-IoT dataset, the system scored 99.6% and outperformed other models in certain aspects such as up-to-99.4% F1-score. The versatility of the system for real-world deployment under mixed traffic conditions is yet to be established.

- Lonare et al. (2023) have carried out a comprehensive review of Network Intrusion Detection Systems (NIDS), reviewing methodologies and technologies employed to tackle emerging cyber threats. The research classifies NIDS into signature-based, anomaly-based, and hybrid models, reviewing their strengths and limitations. It reviews the strengths of the integration of machine learning and deep learning and reviews newer approaches like MCF-MVO-ANN and scalable K-means with Random Forest for intrusion detection. While presenting real-world advice to cybersecurity practitioners, the paper quotes difficulty in achieving detection accuracy with the requirements of scalability and flexibility in real-world contexts.

- Al-Haija et al. (2024) investigated the feasibility of Extreme Learning Machines (ELMs), or Single-Hidden-Layer Feedforward Neural Networks (SLFNs), in developing adaptive Intrusion Detection Systems (IDS) that can detect threats in real-time. Their research identifies ELM's fast learning capacity and efficiency in network anomaly classification, such as DDoS and phishing attacks. With a ten-year development history, ELM-based IDS models have demonstrated high adaptability and precision in various cybersecurity applications. Nonetheless, promising as it sounds, the use of randomly initialized weights could potentially impact model robustness and consistency in extremely dynamic network environments.

- Azimjonov and Kim (2023) proposed a light-weighted and energy-aware IoT environment intrusion detection system utilizing a stochastic gradient descent classifier (SGDC) with ridge regression-based feature selection. The proposed solution eliminates the vulnerability of traditional heavyweight IDS models by achieving an average accuracy of 92.69% and significant reduction in feature dimensionality by 79.93%, making the system suitable for resource-constrained IoT devices. The system was evaluated on KDD-CUP-1999, BotIoT-2018, and N-BaIoT-2021 datasets. However, although it is efficient and adaptable, the system may have difficulty detecting more sophisticated or newer botnet attacks that require closer contextual analysis.

Chapter 3

Problem Statement / Requirement Specifications

3.1 Project Planning

- This project will in this phase plan on how to design and implement a Machine Learning-based **Network Intrusion Detection System** that is capable of detecting malicious activities in network traffic. The system used labeled real-world datasets such as **CIC-IDS 2017, CIC-IDS 2018, and UNSW NB-15.** These datasets have a complete mix of both benign and malicious traffic for training and evaluation purposes. There is a need for proper planning in such a way that would allow systematization in dealing with the data, models' training, and performance analysis.

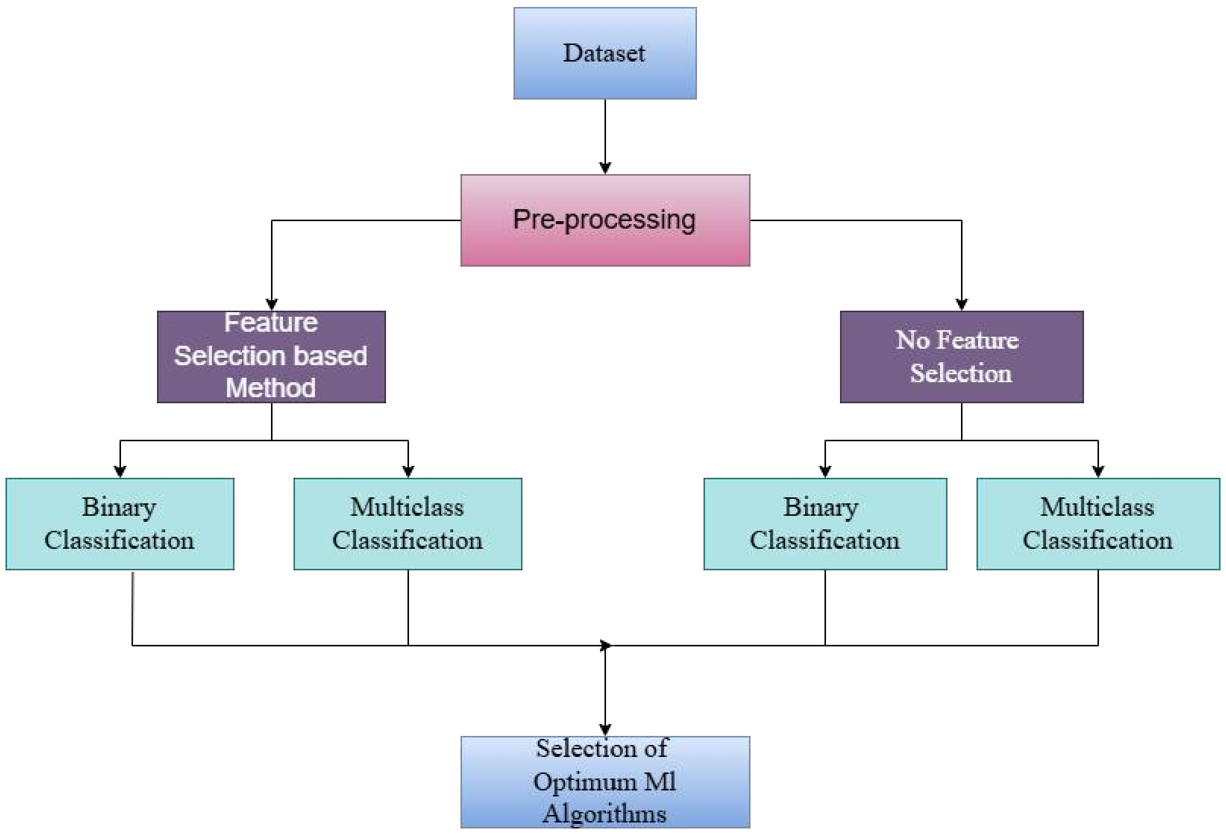

Dataset
Pre-processing
Feature Selection based Method
No Feature Selection
Binary Classification
Multiclass Classification
Binary Classification
Multiclass Classification
Selection of Optimum Ml Algorithms

Figure 3.1: Flowchart of Network intrusion detection .

<u>Project Aims:</u>

I. To study and learn about the form and the character of intrusion detection data present in real-world applications.

II. To preprocess data through procedures such as normalization, encoding, and class balancing to make data prep-ready.

III. To implement and train some of the standard machine learning classification algorithms to identify intrusions.

IV. To compare and test the performance of the algorithms in terms of standard classification evaluation metrics.

V. To create a modular and scalable detection system which could be simply updated or extended further in the future.

✧ **<u>Tools and Platforms Used:</u>**

I. <u>Development Environment:</u>

Google Colab - selected because it is very easy to use, cloud-based execution, and support for GPU/TPU.

II. <u>Programming Language:</u> Python - due to its rich ecosystem for data science and machine learning.

✧ **<u>Libraries and Frameworks:</u>**

I. <u>Data Handling:</u> Pandas, NumPy

II. <u>Machine Learning:</u> Scikit-learn, TensorFlow, Keras

III. <u>Visualization:</u> Matplotlib, Seaborn

IV. <u>Collaboration and Version Control:</u> GitHub or GitLab for maintaining code versions and enabling team collaboration.

Expected Outcomes:

- A functional Network Intrusion Detection System that can differentiate between normal and malicious traffic.
- A comparative study of several machine learning classifiers based on performance metrics including accuracy, precision, recall, and F1-score.
- A final report summarizing the system design, implementation, results, and conclusions, with visualizations and performance graphs.

3.2 Project Analysis (SRS)

The SRS aims to provide a comprehensive functional and non-functional description of the **Network Intrusion Detection System**. The system is based on machine learning models that identify and classify intrusions in a network based on publicly available labeled datasets. SRS ensures that the developed system will be guided by the following: meets performance criteria, executes in real-world environments with reliability, and is scalable to develop further.

Functional Requirements

- Data Handling:

I. Load and process CSV datasets. The datasets used are from CIC-IDS 2017, 2018, and UNSW NB-15.

II. Missing value handling, duplicates, noisy data.

III. Encoding categorical features and normalizing numerical data.

- **Preprocessing:**

I. Feature selection and extraction.

II. Imbalanced dataset balancing by method such as SMOTE.

- **Model Training and Evaluation:**

I. Training various machine learning classifiers such as Logistic Regression, Decision Tree, Random Forest, KNN, SVM, MLP, etc.

II. Model evaluation based on classification metrics: Accuracy, Precision, Recall, F1-score, ROC-AUC.

➢ **<u>Visualization</u>**

I. Produce graphical outputs like confusion matrices, ROC curves, and performance comparison plots.

➢ **<u>Output:</u>**

I. Make a prediction as to whether a network connection is benign or an intrusion.

II. Produce a summary of classification results and a report of performance metrics.

<u>Non-Functional Requirements:</u>

i. **<u>Scalability:</u>**

✓ The system must be able to cope efficiently with high-scale network traffic data.

ii. **<u>Modularity:</u>**

✓ The building blocks (data preprocessing, training, evaluation) must be distinct and reusable.

iii. **<u>Efficiency:</u>**

✓ Make use of GPUs/TPUs through Google Colab for training and testing a model with reasonable speed.

iv. **<u>Maintainability:</u>**

✓ It must have clean documented code which will be under some versioning at GitHub or GitLab

v **<u>Usability:</u>**

✓ It must have the capability of presenting results clearly to aid proper interpretation with satisfactory graphical support.

vi. **<u>Reliability:</u>**

✓ It is expected to generate the same output during runs through giving accurate repeatable results.

3.3 System Design

- System design is a description of the overall structure, data processing, and components involved in designing an intelligent Machine Learning-based **Network Intrusion Detection System**. The system needs to be modular in design so that there are independent modules like **data preprocessing**, model training, and evaluation and can be independently modified without affecting the entire system. This makes it more flexible, maintainable, and is scalable.

The architecture consists in of several key modules:

- **Data Ingestion a Module:**

I. Gathers charges and structures the data.

II. Accepts input in the form of CSV from in UNSW NB-15, CIC-IDS 2018, and CIC-IDS 2017 datasets.

III. Processes proper parsing in and processing of large data.

- **Data Preprocessing Module:**

I. Truncates null values and duplicates from the data.

II. The categorical features are encoded and the numerical features are normalized.

III. Features are chosen in order to achieve dimensionality reduction and improve the models performance.

IV. Handles class imbalance through oversampling methods such as SMOTE.

- **Training of Model in Module:**

Installs machine learning classifier suite:

1. Logistic Regression:

✓ Consider this as a basic tool that assists us in making a choice between two alternatives—such as whether network traffic is safe or not based on what it observes. It provides us with a probability and makes a decision accordingly.

2. **Decision Tree:**

✓ It functions as a decision flowchart. It poses a series of yes/no questions to the data and follows down branches until it comes to a conclusion. It is easy to follow and suitable for describing decision-making processes.

3. **Random Forest:**

✓ Picture having a bunch of decision trees that all chime in and then vote on the answer. That's Random Forest. It's more precise because it takes the averages of individual trees' biases.

4. **K- Nearest Neighbors (KNN):**

✓ This one categorizes data based on its "neighbors" in the data. If all its nearest points are marked as attacks, it probably is an attack as well. It's a judgment based on who one associates with.

5. **Support Vector Machine (SVM):**

✓ SVM is intelligent to create a line of demarcation between categories. It discovers the optimal possible line or curve that discriminates between safe traffic and malicious ones, even when the data isn't linearly separable.

6. **Multilayer Perceptron (MLP):**

✓ This is a form of neural network, a sort of mini-brain. It consists of layers of "neurons" that learn from the information and are able to recognize very intricate patterns. It's useful for those jobs where simple models aren't up to it.

➢ **A Module's Evaluation:**

Performance of models against:

1. **Accuracy:**

✓ That is telling us the ratio of time the model is correct overall. You show it 100 examples and it gets 90 of them right, and its accuracy is 90%. It's a great aggregate measure, but not always—particularly if the data is skewed.

2. **Recall:**

✓ Also known as sensitivity, recall is concerned with how well the model can detect actual attacks. Recall tells us: "Of all the actual attacks, how many did the model successfully capture?" The greater the recall, the fewer attacks the model will fail to detect.

3. **<u>F1 Score:</u>**

✓ This is recall times precision, or the balance between precision and recall. It's a harmonic score that tells us how well the model is performing on catching attacks and on not catching a lot of false positives. A good F1-score shows us that the model is well-balanced.

4. **<u>ROC-AUC (Receiver Operating Characteristic - Area Under Curve):</u>**

✓ It is a metric that indicates how well the model discriminates at different thresholds between attacks and normal traffic. The higher the AUC, the more accurate the model in distinguishing between the two regardless of where you place the line.

5. **<u>Confusion Matrix:</u>**

✓ This is a chart that displays what the model got right and wrong—how many actual attacks it identified, how many it missed, and how many false alarms it generated. It's like a fine-grained report card for classification.

➢ **<u>Visualization & Reporting Module:</u>**

<u>Plots performance with:</u>

1. **<u>ROC Curves</u>**

✓ ROC (Receiver Operating Characteristic) curves allow us to visualize how well our model performs in separating two classes—such as attack and normal traffic. The curve shows the true positive rate (recall) against the false positive rate at varying threshold settings.A curve tightly following the top-left corner of the plot indicates a good model.

2. **<u>Bar Plots of Evaluation Metrics</u>**

✓ Bar plots provide a simple and quick comparison of performance of different models using important metrics such as accuracy, precision, recall, and F1-score.One bar per metric for a given model.These plots are easy to identify with which model is best in which region (e.g., high recall but low precision).

3. **Confusion Matrices**

- ✓ A confusion matrix is a table that displays what the model predicted against what actually happened.It divides it into: True Positives, False Positives, True Negatives, and False Negatives.This informs you not only about how many predictions were correct, but about the type of errors the model is doing.

- ➢ Prints an end of stream summary report with conclusions and model performance.This modularity guarantees that optimizations can be introduced at any moment e.g. trying new models hyperparameter search, or introducing new data stream without impacting the balance of the rest of the pipeline.

3.3.1 Constraints of Design:

- ✧ During the creation of our **Network Intrusion Detection System (NIDS),** we encountered several constraints in real practice that had impacts on the completion of the project. The constraints were from varying sources and some from the platform and tool selection, and the rest due to the scale and complexity of data we dealt with. Each had an effect on our design choice during development.

- ➢ **Software Constraints:**

- ✓ We decided to develop using **Google Colab** as our primary development environment due to the flexibility and free **GPU and TPU** access that it provided, which proved to be an immense assistance in training models of large sizes. But developing in a cloud-based environment had its own set of constraints such as session timeouts and memory limits that we had to navigate around.

- ✓ The whole system was developed in **Python 3**, providing us with access to a multitude of libraries. For data processing, **Pandas and NumPy** were used mostly. For constructing and testing within our machine learning in a model we employed in a framework like **Scikit learn** and **TensorFlow**. Visualizations are done in and handled by tools like **Matplotlib** and **Seaborn**, enabling us to make sense of our results visually.

- ✓ In order to collaborate and keep track of code versions so we utilized **GitHub** to ensure that everything remained neat and in sync particularly when collaborating over multiple sessions and contributors.

➢ **<u>Hardware Constraints</u>**

✓ As all our work is done in **Google Collab** so we were constrained by its inbuilt limitations. For instance, we couldn't keep sessions open endlessly, which means we had to re-run and save progress every now and then. Also, the **RAM** was limited to approximately **12 GB**, and **GPU or TPU** resource access wasn't always available. This meant that we needed to be judicious about how much data we would load and how compute-intensive our training steps were. We couldn't afford to use powerful local machines or specialized hardware.

➢ **<u>Dataset Constraints</u>**

✓ We used three prominent datasets: **CIC-IDS 2017, CIC-IDS 2018, and UNSW NB-15**. They're all very large and feature-rich, which was wonderful for training a robust model but also brought a bit of a challenge or two.

✓ To start, because the datasets were so large and **Colab** has memory constraints, we frequently had to sample the data or split it into pieces. Secondly the data formats are not always uniform so we spent a considerable amount of time on cleaning and standardizing everything before we could even start modeling. One of the biggest issues was class imbalance, most of the records are normal traffic, with very few percent being attacks. To deal with this we used balancing techniques such as **SMOTE** so that our models can learn from both sets of data more effectively.

Chapter 4

Implementation:

1.1 Methodology:

- Our primary objective in this project was to develop a machine learning-based intrusion detection system that would be able to classify normal and malicious network traffic with good accuracy. We built our system upon popular datasets such as **CIC-IDS 2017, CIC-IDS 2018, and UNSW NB-15**, all of which have real-world instances of network attacks and benign traffic.

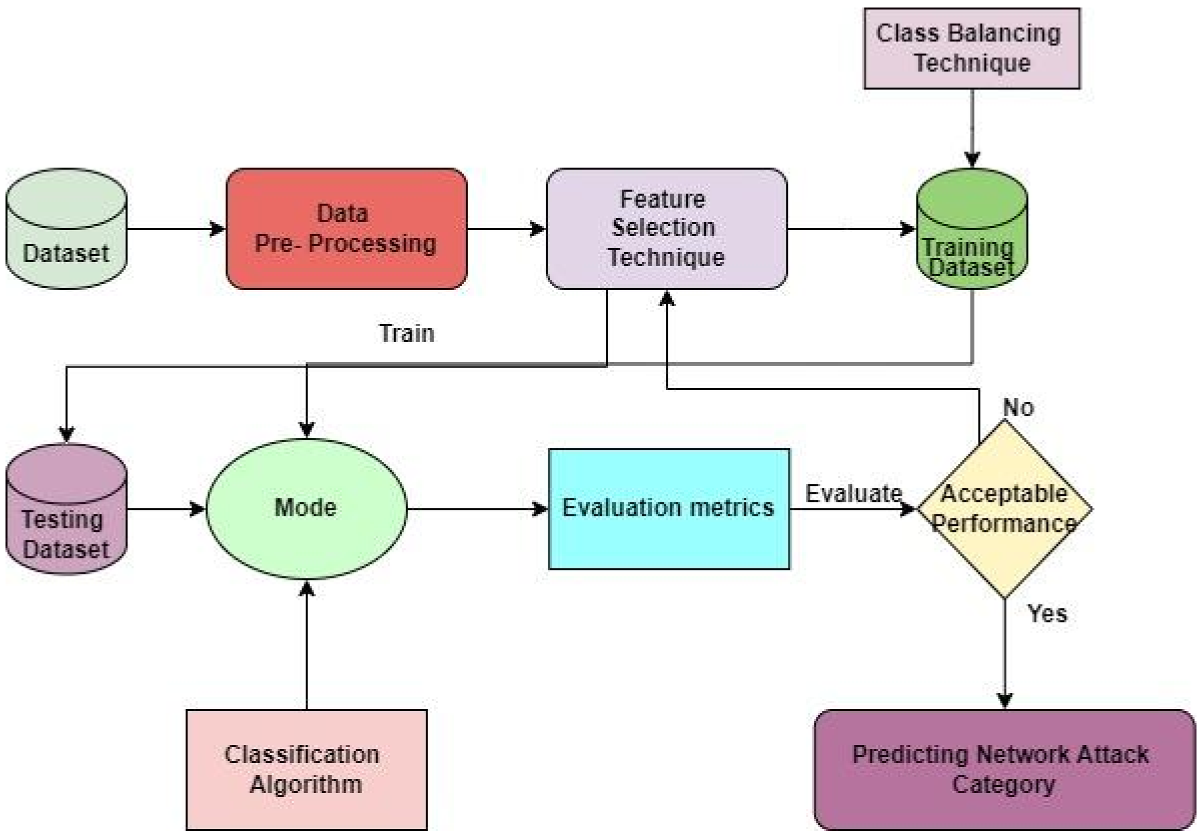


Figure 4.1: Network Intrusion Detection Methodology.

To achieve this, we employed **Python** and a number of data processing and machine learning libraries. The overall steps in our approach were:

1. **Data preprocessing:**

- Preparing the datasets, encoding categorical features, scaling the values, and dealing with imbalanced classes used in methods such as **SMOTE**.

2. Model training:

- We trained a number of classification models such as in Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, Support Vector Machine, and Multilayer Perceptron.

3. Evaluation:

- Accuracy, precision, recall, F1-score, and ROC-AUC were used to evaluate each model for its performance in intrusion detection.

This pipeline enabled us to compare and choose the most performing model for our NIDS.

4.2 Testing OR Verification Plan:

- Once we trained the models, we emphasized testing in a how well they perform under various conditions. We divided our datasets into test and training sets (usually applying a 70/30 split), making sure that the models were tested on unseen data.

- **To test a model performance:**

I. We employed confusion matrices to ensure how many of the predictions were correct versus incorrect.

II. We calculated precision, recall, and F1-scores to gain a better idea of how well the model is performing particularly when working with imbalanced datasets.

III. We utilized cross-validation in order to confirm that the outcome was not an isolated event but consistent across different segments of the dataset.

4.3 Result Analysis OR Screenshots:

- These are the performance results for the trained models on the given datasets.

CICIDS 2017

- Monday

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.9997	1.00/0.99	1.00/0.98	1.00/0.99
Random Forest	0.9995	1.00/1.00	1.00/0.96	1.00/0.98
Logistic Regression	0.9941	1.00/0.77	1.00/0.61	1.00/0.68
SVM	0.9952	1.00/0.89	1.00/0.60	1.00/0.72
Gaussian Naive Bayes	0.7973	1.00/0.05	0.80/1.00	0.89/0.09
CNN	0.9962	0.9963	0.9962	0.9962

- Tuesday

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.9954	0.67	0.73	0.69
Random Forest	0.9953	0.7	0.65	0.67
Logistic Regression	0.9913	0.63	0.46	0.42
SVM	0.9905	0.62	0.46	0.42
Gaussian Naive Bayes	0.9878	0.42	0.77	0.47
CNN	0.9932	0.9933	0.9932	0.9932

-Wednesday

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree (DT)	0.9961	0.9962	0.9961	0.9961
Random Forest (RF)	0.9997	0.9997	0.9997	0.9997
Logistic Regression	0.9719	0.9731	0.9719	0.9719
SVM	0.981	0.981	0.981	0.981
Gaussian NB (GNB)	0.7627	0.8054	0.7627	0.742
CNN	0.9801	0.9802	0.9801	0.9801

-Thursday

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.9954	0.67	0.73	0.69
Random Forest	0.9953	0.7	0.65	0.67
Logistic Regression	0.9913	0.63	0.46	0.42
SVM	0.9905	0.62	0.46	0.42
Gaussian Naive Bayes	0.9878	0.42	0.77	0.47
CNN	0.9932	0.9933	0.9932	0.9932

- Friday

Model	Accuracy	Precision	Recall	F1-Score
Decision Tree	0.9997	1	1	1
Random Forest	0.9995	1	1	1
Logistic Regression	0.9941	1	1	1
SVM	0.9952	1	1	1
Gaussian Naive Bayes	0.7973	1	0.8	0.89
CNN	0.9962	0.9963	0.9962	0.9962

CICIDS 2018

❖ 02-15-2018

Model	Accuracy	Precision	Recall	F1-Score	G-Mean	MCC	Kappa
Logistic Regression	0.9994	0.9994	0.9994	0.9994	0.9887	0.9936	0.9936
Decision Tree	1	1	1	1	1	0.9999	0.9999
Random Forest	1	1	1	1	0.9998	1	1
K-Nearest Neighbors	0.9999	0.9999	0.9999	0.9999	0.9997	0.9994	0.9994
Support Vector Machine	0.9996	0.9997	0.9996	0.9997	0.9966	0.9964	0.9964
Multilayer Perceptron	0.9999	0.9999	0.9999	0.9999	0.9995	0.9991	0.9991

❖ 02-21-2018

Model	Accuracy	Precision	Recall	F1-Score	G-Mean	MCC	Kappa
Logistic Regression	0.9999	0.9999	0.9999	0.9999	0.9931	0.9997	0.9997
Decision Tree	1	1	1	1	1	1	1
Random Forest	1	1	1	1	1	1	1
K-Nearest Neighbors	1	1	1	1	1	1	1
Support Vector Machine	1	1	1	1	0.9986	1	1
Multilayer Perceptron	1	1	1	1	1	1	1

❖ 02-28-2018

Model	Accuracy	Precision	Recall	F1-Score	G-Mean	MCC	Kappa
Logistic Regression	0.888	0.79	0.89	0.84	0.65	0.63	0.65
Decision Tree	0.9434	0.94	0.94	0.94	0.91	0.88	0.88
Random Forest	0.9182	0.91	0.92	0.92	0.85	0.85	0.85
K-Nearest Neighbors	0.8527	0.81	0.85	0.83	0.67	0.67	0.67
Support Vector Machine	0.9992	0.9994	0.9992	0.9994	0.9962	0.9961	0.9961
Multilayer Perceptron	0.9998	0.9998	0.9998	0.9998	0.9994	0.9990	0.9990

UNSW NB-15

Model	Accuracy	Precision	Recall	F1-Score	G-Mean	MCC	Kappa
ANN	0.8812	0.8776	0.8812	0.8761	0.9327	0.8343	0.834
RF	0.9007	0.9001	0.9007	0.8999	0.9445	0.8616	0.8614
SVM	0.8708	0.8506	0.8708	0.8555	0.8554	0.821	0.8189
CNN	0.8868	0.894	0.8868	0.8839	0.9375	0.8443	0.8427
DNN	0.8858	0.8856	0.8858	0.8821	0.9355	0.8408	0.8405
RNN	0.878	0.8738	0.878	0.8737	0.9318	0.8307	0.8301
LSTM	0.8809	0.8806	0.8809	0.8748	0.9329	0.8342	0.8337
Bi-LSTM	0.8778	0.8809	0.8778	0.8741	0.931	0.8296	0.8293
GRU	0.8763	0.865	0.8763	0.8683	0.9301	0.8277	0.8271
Bi-GRU	0.8845	0.8847	0.8845	0.8805	0.9352	0.8393	0.8389
GNN	0.8785	0.8517	0.8785	0.8565	0.9303	0.8324	0.8295

4.4 Quality Assurance:

- In order to have our system functioning and also safe, we installed quality assurance steps:

- Code checks regularly through GitHub maintained that inputs coming from a diverse range of contributors were clean as well as standardized.

- Cross-validation techniques and persistent testing were undertaken to reduce the likelihood of overfitting, as well as bias.

- Performer benchmarks facilitated us to systematically test and evaluate models.

- The system was also tested on some samples and subsets of the data to validate stability and flexibility.

Overall, these efforts allowed us to deliver a sound and reliable intrusion detection system that could be further developed or actually tested in real-world environments.

Chapter 5

Standards Adopted

5.1) Standards of Design

- We received a scalable, user-friendly, and modular design at the design stage. Some of our design principles and standards utilized below:

1) Modularity:

- The system was separated into independent modules such as data preprocessing, model training, evaluation, and visualization. It made the project simple to debug and maintain.

2) **Reusability:**

- Data cleaning code, feature scaling, and model scoring code was written as reusable functions and scripts to prevent duplication.

3) Clarity and Simplicity:

- Easy-to-read and comprehensible data flow, especially in model pipeline development and testing, was employed in a way that other members of the team or reviewers would be able to easily comprehend the system design.

4) UML Guidelines:

- Block diagrams or easy UML diagrams were employed in the definition of the interaction of the components and workflow in making the design concrete before coding.

5.2) Coding Standards

- When we were writing readable code and quality code, we followed standard Python coding conventions:

1) PEP 8:

- We always followed the official Python style guide when we coded consistent and clean code. That also included good variable names, proper indentation, and short functions.

2) **<u>Documentation:</u>**

➢ Functions and modules were properly documented with inline comments and docstrings explaining how to call them and why.

3) **<u>Version Control:</u>**

➢ Git and GitHub were used for managing changes, branch management, and collaboration in an efficient way of time. Commits were always made on a routine basis with explanatory commit messages to maintain project history clean.

4) **<u>Code Reusability and Optimisation:</u>**

➢ Code duplicate copies of code, wherever possible, were re-coded in function form and resolved the problem of coding optimization so that it executed faster, particularly at model training.

5.3 Testing Criteria

➢ Testing was needed to ensure the NIDS functions as needed and reproducibly:

➢ **<u>Unit Testing:</u>**

✓ Core features like data normalisation, encoding, and evaluation metrics were tested manually with controlled input/output tests to see expected behavior.

1) **<u>Cross-Validation:</u>**

✓ Generality and performance of models on diverse datasets was achieved through k-fold cross-validation.

2) **<u>Performance Metrics:</u>**

✓ Models were checked on commonly utilized metrics of Accuracy, Precision, Recall, F1-score, and AUC-ROC widely used to check performance on firm ground for the task of classifications.

3) **<u>Confusion Matrices:</u>**

✓ Utilized for visually viewing output of classifications as well as checking class-specific mistakes, mainly false negatives and false positives.

4) **<u>Management of Data Sets:</u>**

✓ We managed data ethically by dividing data in the ratio of 70/30 or 80/20 so that leakage or overfitting is not done.

Chapter 6:

Conclusion and Future Scope:

6.1 Conclusion

➢ During the course of this project, we designed and developed a machine learning-based **Network Intrusion Detection System (NIDS)** from publicly available datasets such as **CIC-IDS 2017, CIC-IDS 2018,** and **UNSW NB-15.** Each of these datasets consisted of a full dataset of normal as well as malicious network traffic on which we successfully trained and tested various classification models.

➢ We tested and experimented with several algorithms like **Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, Support Vector Machine, and Multilayer Perceptron** throughout the implementation phase. We evaluated all the models using standard performance metrics to identify which one is suitable for intrusion detection. Among them, models like **Random Forest and SVM** were found to be more accurate and trustworthy.

➢ This paper has illustrated that machine learning can be used to contribute positively to the detection of malicious network activity with the right preprocessing and model choice. Our system is a step towards developing smart and autonomous intrusion detection systems that can improve cybersecurity systems.

6.2 Future Scope

➢ Though the current system shows promising results, there is great scope for further development:

1. **Real-Time Detection:**

✓ Current implementation is grounded on static datasets. Incorporating real-time packet capture and live detection would make the system more deployable.

2. **<u>Deep Learning Models:</u>**

- ✓ State-of-the-art models such as **LSTM, CNN**, and transformer-based models can be explored to determine temporal and spatial patterns in network traffic.

3. **<u>Feature Engineering:</u>**

- ✓ Sophisticated and self-service feature extraction techniques can be included to improve detection rates and dimensionality reduction.

4. **<u>Deployment:</u>**

- ✓ Subsequent work can then focus on deploying the system as a light service or application that can be integrated into enterprise security infrastructures.

5. **<u>Hybrid Models:</u>**

- ✓ Blending rule-based and machine learning approaches could lead to stronger detection systems that detect known threats in addition to labeling new, zero-day threats.

Reference :

1) Bhawana Sharma, Lokesh Sharma, Chhagan Lal, Satyabrata Roy (2023). Anomaly-based Intrusion Detection System (IDS) for IoT networks using Deep Learning Techniques. Computers and Electrical Engineering, 107, 108626.

2) Yanfang Fu, Yishuai Du, Zijian Cao, Qiang Li, Wei Xiang (2022). A Deep Learning Approach for Network Intrusion Detection with Imbalanced Data. Electronics 2022, 11, 898.

3) Sk. Tanzir Mehedi, Adnan Anwar, Ziaur Rahman, Kawsar Ahmed, Rafiqul Islam(2023). Dependable Intrusion Detection System for IoT: A Deep Transfer Learning-Based Approach. IEEE Transaction on Industrial Informatics, Vol. 19, No. 1.

4) Yakub Kayode Saheed, Aremu Idris Abiodun, Sanjay Misra, Monica Kristiansen Holone, Ricardo Colomo-Palacios(2022) . A machine learning-based intrusion detection system for detecting Internet of Things network attacks. Alexandria Engineering Journal(2022) 61, 9395–9409.

5) Emad-ul-Haq Qazi, Muhammad Imran, Noman Haider, Muhammad Shoaib, Imran Razzak(2022). An intelligent and efficient network intrusion detection system using deep learning. Computers and Electrical Engineering 99 (2022) 107764.

6) Kabir, E., Hu, J., Wang, H., & Zhuo, G. (2018). A Novel Statistical Technique for Intrusion Detection Systems. Future Generation Computer Systems, 79, 303–318.

7) Ferrag, M. A., Maglaras,L.,Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Journal of Information Security and Applications, 50, 102419.

8) Yogendra Narayan and published in (2021). Comparative Analysis of SVM and Naive Bayes Classifier for EEG Signal Classification.Materials Today: Proceedings 37 (2021) 3242-3245.

9) Şevval Şolpan, Hakan Gündüz, and Kerem Küçük. 2024. Enhanced Feature Extraction and Machine Learning Algorithm-Based Wi-Fi Network Intrusion Detection. 8th International Artificial Intelligence and Data Processing Symposium (EDP24), Sept 21-22, 2024, Malatya, Türkiye.

10) Ke He, Dan Dongseong Kim and Muhammad Rizwan Asghar (2023). Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey. IEEE Communications Surveys and Tutorials, Vol.25, No. 1.

11) Nardelli, Pedro & Korium, Mohamed & Saber, Mohamed & Beattie, Alexander & Narayanan, Arun & Sahoo, Subham. (2023). Intrusion detection system for cyberattacks in the Internet of Vehicles environment. Ad Hoc Networks. 153. 16. 10.1016/j.adhoc.2023.103330.

12) Vadakkethil Somanathan Pillai, Sanjaikanth & Vallabhaneni, Rohith & Pareek, Piyush & Dontu Ph.D., Sravanthi. (2024). Strengthening Cybersecurity using a Hybrid Classification Model with SCO Optimization for Enhanced Network Intrusion Detection System. 1-9. 10.1109/ICDCOT61034.2024.10516247. .

13) Lonare, M B et al. “Real-Time Network Monitoring and Reporting Using Network Intrusion Detection System.” *2024 IEEE 9th International Conference for Convergence in Technology (I2CT)* (2024): 1-6.

14) Abu Al-Haija, Qasem & Al-Tamimi, Shahad & Alwadi, Mazen. (2024). Analysis of Extreme Learning Machines (ELMs) for intelligent intrusion detection systems: A survey. Expert Systems with Applications. 253. 124317. 10.1016/j.eswa.2024.124317.

15) Jahongir Azimjonov, Taehong Kim,Stochastic gradient descent classifier-based lightweight intrusion detection systems using the efficient feature subsets of datasets,Expert Systems with Applications,Volume 237, PartB,2024,121493,ISSN09574174,https://doi.org/10.1016/j.eswa.2023.12143

16) Md. Alamin Talukder, Khondokar Fida Hasan, Md. Monowarul Islam, Md Ashraf Uddin, Arnisha Akhter, Mohammad Abu Yousuf, Fares Alharbi, Mohammad Ali Moni.

17) Muhammad Azmi Umer, Khurum Nazir Junejo, Muhammad Taha Jilani, Aditya P. Mathur, 2022.

18) Nitasha Sahani, Ruoxi Zhu, Chen-Ching Liu, 2023.

19) Muhammad Asif, Sagheer Abbas, M.A. Khan, Areej Fatima, Sang-Woong Lee, 2022.

20) Emad E. Abdallah, Ahmed Fawzi (2022).

21) Untargeted White Box Adversarial Attack with Heuristic Defense Methods in Real-Time Deep Learning Based Network Intrusion Detection System Khushnaseeb Roshan .Aasim Zafar, Sheikh Burhan Ul Haque.

22) Complex Adaptive Systems Conference Theme: Big Data, IoT, and AI for a Smarter Future Malvern, Pennsylvania, June 16-18, 2021 Network Intrusion Detection System using Deep Learning Lirim Ashiku1 Cihan Dagli.

23) Wang, Xiaosong & Qiao, Yuxin & Xiong, Jize & Zhao, Zhiming & Zhang, Ning & Feng, Mingyang & Jiang, Chufeng. (2024). Advanced Network Intrusion Detection with TabTransformer. Journal of Theory and Practice of Engineering Science. 4. 191-198.10.53469/jtpes.2024.04(03).18.

24) Sajid, M., Malik, K.R., Almogren, A. *et al.* Enhancing intrusion detection: a hybrid machine and deep learning approach. *J Cloud Comp* **13**, 123 (2024). https://doi.org/10.1186/s13677-024-00685-x.

25) Stochastic Gradient Descent Intrusions Detection for Wireless Sensor Network Attack Detection System Using Machine Learning HADEEL M. SALEH 1, HEND MAROUANE2, AND AHMED FAKHFAKH3

26) (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

SAMPLE INDIVIDUAL CONTRIBUTION REPORT :

Network - Intrusion Detection System

SAMIR KUMAR
22054355

Abstract:

➢ This project will implement and compare an effective Network Intrusion Detection System (NIDS) using machine learning models. Having labeled network traffic datasets such as CIC-IDS 2018, we had tried and compared several classification models for determining malicious activity in network environments. The goal is to enhance cyber threat detection and towards automated security systems.

Individual Contribution and Findings:

➢ My primary responsibility for this project was to do the data preprocessing and train and test classification models on the CIC-IDS 2018 dataset. My responsibility was to clean the dataset, select suitable features, handle imbalanced classes and one-hot encode the categorical features. The major portion of my work was to normalize the input data and prepare it to be fed into machine learning models.

➢ I tried and tested some models like Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, Support Vector Machine, and Multilayer Perceptron with Scikit-learn and TensorFlow libraries. All the models were trained on a training set split from the CIC-IDS 2018 dataset and evaluated based on metrics such as accuracy, precision, recall, F1-score, and confusion matrix evaluation.

➢ One of the primary challenges that I faced was how to handle memory limitations in Google Colab with a huge dataset. I managed to handle this by means of data sampling and splitting in order to accommodate available resources.

➢ Through this project, I gained direct experience in supervised learning, model selection techniques, and real-world data set handling for cybersecurity. It also raised my awareness with respect to the advantages and limitations between different machine learning approaches in application to intrusion detection.

Individual Contribution to Preparation of Project Report:

➢ I assisted in writing the Introduction (Chapter 1) and the Implementation (Chapter 3) sections of the project report. Specifically, I assisted in explaining the model training process describing the design constraints and showing the classification of outputs in an appropriate visual and interpretation. I assisted with the testing and result analysis sections as well by generating and interpreting the evaluation metrics and confusion matrices.

Full Signature of Supervisor: Full Signature of the Student:

…………………………………… ………………………………

SAMPLE INDIVIDUAL CONTRIBUTION REPORT :

Network - Intrusion Detection System

Aditya Kumar Thakur
22054270

Abstract:

- The purpose of the project is to examine the CICIDS 2017 dataset in order to detect and classify different network intrusions. Processing data from a series of days with attack traffic and benign traffic, the purpose is to use machine learning models for detection and classification with precision. Feature engineering, preprocessing, and performance measurement are prominent phases of the project.

Individual contribution and findings:

- In order to do this **for Thursday's data**, I had performed an intensive analysis to feel the attack-type distribution. I have encoded the labels into numeric by LabelEncoder followed by visualization techniques (histograms, and value counts) to check class imbalances. I had also maintained data consistency while merged data frames as well as taken care of missing values appropriately by techniques such as.dropna() and.fillna().

- **For Friday's traffic**, which had both benign and attack traffic (i.e., DDoS) intermixed, I used a similar preprocessing pipeline. The biggest issue was the size of the dataset, which I addressed by using chunk loading and sampling techniques. I also checked that post-cleaning, each data frame had the same structure and labeling. I trained and developed machine learning models, namely Random Forest and Logistic Regression, to compare the performance of the cleaned data in intrusion detection. I implemented hyperparameter optimization and result checking through accuracy and confusion matrices. I found Random Forest to work better at all times, especially with processed Friday traffic, and with high accuracy even for class-imbalanced data.

- During this work, I virtually gained hands-on experience in how to preprocess massive cyber data, encoding strategies, and model performances. I also gained hands-on experience in handling multi-label as well as imbalanced data, which is commonly a problem with intrusion detection systems.

Individual contribution towards report generation of project :

- I was assigned to collect all the datasets and Implementation (Chapter-4) of the project report. It involved recording what was done when cleaning and processing Thursday and Friday data and results of model performance reporting. I also checked and standardized figures, tables, and results for uniformity in these sections.

Full Signature of Supervisor:	Full Signature of the Student:
……………………………………	………………………………

SAMPLE INDIVIDUAL CONTRIBUTION REPORT :

Network - Intrusion Detection System

Ashlok Prasad Yadav
22054022

Abstract:

➢ The objective of the project is to analyze the CICIDS 2017 dataset in trying to create an efficient network intrusion detection system. The objective is to preprocess network traffic data, identify patterns, and employ machine learning algorithms to detect malicious traffic. Extensive exploratory data analysis, feature selection, and classification model performance measurement form the project.

Individual Contribution and Findings:

➢ I was mostly dealing with exploratory data analysis (EDA) and training classification models from the **CICIDS 2017 dataset,** i.e., Monday traffic logs, Tuesday traffic logs, and Wednesday traffic logs. My activities involved preprocessing the data such as missing value handling, categorical feature conversion, numerical feature scaling, and plotting the distribution of attack labels. I have utilized libraries such as pandas, NumPy, seaborn, and matplotlib for EDA and skillearn for training machine learning models.

➢ **For Monday's data**, I had investigated the kind of benign traffic, knowing about the types of network services employed and inspecting the protocol and flag feature distribution. I utilized some plots such as heatmaps and distribution plots to determine the correlations between features.

➢ **For Tuesday's** traffic that contained brute force SSH and FTP attacks, I made contributions towards removing attack samples, detection of anomalies in connection size, bytes, and packet rate. I used attack feature importance and normal traffic feature importance which helped with the feature selection process.

➢ **For Wednesday** data, I worked with mitigating DoS and Heartbleed attacks. I oversampled class imbalance and developed classification models such as Random Forest and Decision Tree classifiers. I evaluated the models using accuracy, precision, recall, and F1-score metrics. I found that Random Forest performed better than other models in recognizing these kinds of attacks.

➢ During the project, I had modular notebook development with reusable code functions, commented images, and inline comments. I also helped the team debug data loading and feature transformation problems on various traffic days.

Personal contribution towards preparation of project report:

➢ My role is to do Standards Adopted (Chapter-5) plays a critical role in the project report.The design followed separating components such as data preprocessing, model training, evaluation, and visualization for better clarity and debugging. We emphasized by implementing common functionalities as reusable scripts and functions, reducing redundancy were maintained throughout the development process to enhance team collaboration and ease of review.

Full Signature of Supervisor: Full Signature of the Student:

………………………………… ……………………………..

SAMPLE INDIVIDUAL CONTRIBUTION REPORT :

Network - Intrusion Detection System

Yuvraj Singh
22054366

Abstract :

- This notebook presents a machine-learning pipeline to detect NETWORK INTRUSION DETECTION from the UNSW-NB15 dataset. Binary classification and multi-class classification approaches are provided to classify normal and malicious traffic, along with various types of attacks. The process encompasses data preprocessing methods such as one-hot encoding, label encoding, data normalization, and feature selection. Different supervised learning models are used, including Linear Regression, Logistic Regression, Support Vector Machines, Decision Trees, Random Forests, K-nearest neighbors, and Neural Networks. Models are evaluated based on accuracy and visualization of predicted vs actual results. Trained models and preprocessed data are saved for future use. This study depicts the effective use of machine learning techniques to enhance cybersecurity threat detection.

Individual Contribution and Findings:

- For this project, I created a complete machine learning pipeline to detect network intrusions based on the UNSW-NB15 dataset. I was mainly in charge of preparing the data and building the machine learning models. I cleaned the dataset, handled missing values, and used SMOTE to balance the classes. Then I trained models like Decision Tree, Random Forest, and XGBoost. XGBoost gave us the best results in terms of accuracy.

- This project helped me get hands-on experience with real data and taught me how important data preprocessing is. I also learned how to evaluate and compare different models effectively.

Individual Contribution to Project Report Preparation :-

- I was responsible for writing the parts related to data preprocessing, model implementation, and the results section. I explained how we cleaned the dataset, handled class imbalance using SMOTE, and described how each machine learning model was built and evaluated. I also helped in proofreading the final report and making sure the technical sections were clear and accurate.

Full Signature of Supervisor:	Full Signature of the Student:
…………………………………	……………………………..

SAMPLE INDIVIDUAL CONTRIBUTION REPORT :

Network - Intrusion Detection System

Krish Mahaseth
22054047

Abstract :

➢ The project is centered around the development of a machine learning-driven Network Intrusion Detection System (NIDS) using publicly accessible data sets like UNSW NB-15. The goal is to discover malicious network traffic through the training and testing of different classification models. The system intends to enhance cybersecurity through the detection of attacks more efficiently.

Individual Contribution and Findings:

➢ My primary task in this project was the data preprocessing and model development phases, which I embarked on alongside the other stages. I began by pre-processing the dataset, detecting and replacing missing values, encoding categorical features, and using SMOTE to address class imbalance, which enhanced model performance for minority classes.

➢ After data preparation, I applied and tested machine learning algorithms including Decision Tree, Random Forest, and XGBoost. I worked and assisted with data set UNSW-NB15.I organized my work by initially separating it into phases, data preparation, model testing, and evaluation ,such that each phase was properly executed prior to advancing to the next. I further developed confusion matrices as well as classification reports for the sake of comprehending model performance.

Individual Contribution to Project Report Preparation :-

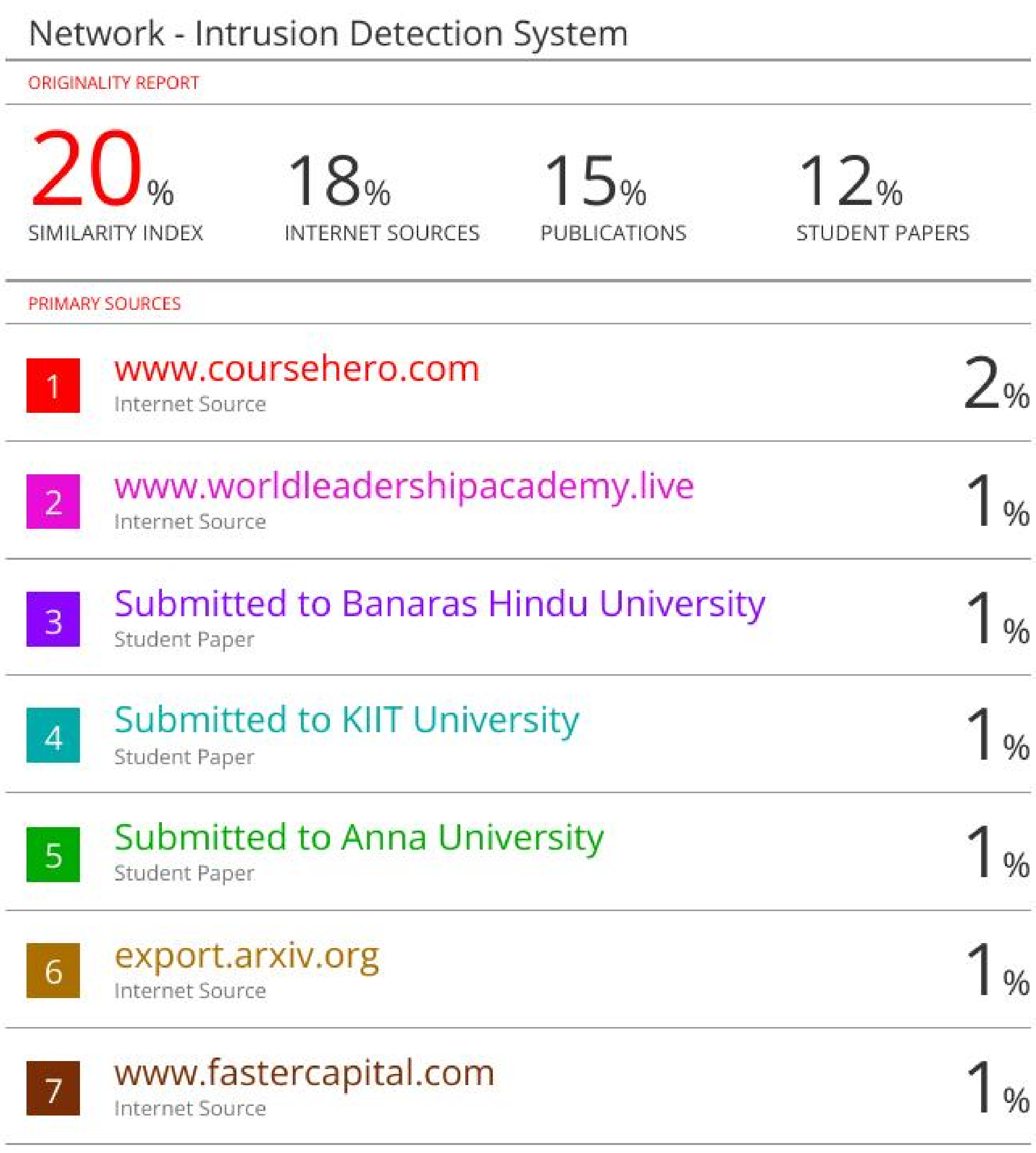
➢ I was responsible for the parts with chapter 6 and assisted with the data set UNSW-NB15 .This project helped me learn about machine learning applications in practice in real life, particularly in cybersecurity. I realized how data quality, preprocessing, and model selection are all essentials in constructing a stable detection system.

Full Signature of Supervisor: Full Signature of the Student:

………………………………… ……………………………..

TURNITIN PLAGIARISM REPORT

(This report is mandatory for all the projects and plagiarism must be below 25%)



Network - Intrusion Detection System

ORIGINALITY REPORT

20%	18%	15%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	2%
2	www.worldleadershipacademy.live Internet Source	1%
3	Submitted to Banaras Hindu University Student Paper	1%
4	Submitted to KIIT University Student Paper	1%
5	Submitted to Anna University Student Paper	1%
6	export.arxiv.org Internet Source	1%
7	www.fastercapital.com Internet Source	1%