Enhanced Network Detection System using NSL-KDD

Tanniru Swetha Menon,
Gondrala Sasank Krishna,
Lingam Ratna Kumari,
Chappidi Naga Manikanta

Department of CSE, K L University, Vaddeswaram, Guntur, Andhra Pradesh, 2200031744@kluniversity.in
Department of CSE, K L University, Vaddeswaram, Guntur, Andhra Pradesh, 2200030464@kluniversity.in
Department of CSE, K L University, Vaddeswaram, Guntur, Andhra Pradesh, 2200030531@kluniversity.in
Department of CSE, K L University, Vaddeswaram, Guntur, Andhra Pradesh, 2200031316@kluniversity.in

ABSTRACT

Intrusion detection systems must be able to adjust to various attack patterns in various network environments due to the increasing sophistication of cyber threats. When used in real-world situations, traditional systems that were trained on a single dataset frequently have trouble generalizing. This study suggests a hybrid method that uses ensemble techniques to integrate various machine learning models that have been trained on various datasets. We developed two distinct models that captured a variety of attack signatures and common behavior patterns using various network traffic datasets. Improved detection capabilities were obtained by combining these models with majority voting procedures, stacking classifiers, and weighted averaging. While keeping false positive rates low, our experimental implementation showed improved accuracy in identifying known and unknown intrusions. Python was used in the Google Collab environment to develop the system, enabling scalable computation and simple reproducibility. The hybrid approach outperforms individual models in terms of detection accuracy, precision, recall, and F1-score, according to performance evaluation across a number of metrics. A useful framework for creating more robust network intrusion detection systems is provided by this work.

Keywords: Cybersecurity Analytics, Model Fusion, Anomaly Detection, Machine Learning Ensemble, Network Security, and Threat Intelligence

1.0 INTRODUCTION

With the rapid growth of cyberattacks and evolving network threats, Intrusion Detection Systems (IDS) have become an essential component of cybersecurity. Traditional IDS models trained on a single dataset often fail to adapt to new or unseen attack patterns, limiting their effectiveness in real-world network environments. To address this challenge, a hybrid intrusion detection approach is proposed that integrates multiple machine learning models trained on different datasets.

This hybrid system leverages ensemble learning techniques such as majority voting, stacking, and weighted averaging to combine the strengths of individual models. By learning from diverse network traffic sources, the system achieves improved detection accuracy, adaptability, and robustness while maintaining low false-positive rates. The approach provides a scalable and efficient framework for enhancing network security using advanced machine learning techniques.

1.1. PROJECT STRATEGIES

- **Data Collection and Preprocessing:** Multiple publicly available network intrusion datasets (such as NSL-KDD, CICIDS2017, etc.) are collected. Data is cleaned, normalized, and encoded to handle missing values, redundant features, and categorical attributes for consistent input to machine learning models.
- **Model Selection and Training:** Different machine learning algorithms like Random Forest, XGBoost, Decision Tree, and SVM are trained separately on different datasets to capture diverse attack signatures and network behavior patterns.
- **Hybrid Ensemble Formation:** The trained models are combined using ensemble techniques — **majority voting, stacking classifiers, and weighted averaging** — to integrate their predictions and improve the overall decision-making capability.
- **Performance Evaluation:** The hybrid model's performance is evaluated using metrics such as Accuracy, Precision, Recall, F1-Score, and ROC Curve. Comparative analysis with individual models helps measure the improvement achieved by hybridization.

- **Implementation Environment:** The project is developed using **Python** in the **Google Collab** environment for ease of computation, scalability, and reproducibility.
- **Optimization and Validation:** Hyperparameter tuning and cross-validation are performed to enhance model efficiency and reduce overfitting, ensuring the model performs well on unseen data.
- **Deployment and Future Scope:** The final hybrid model can be deployed in real-time network monitoring systems to detect intrusions dynamically. Future extensions may include integration with deep learning and real-time streaming data analysis for enhanced adaptability.

1.2. PROJECT DETAILS

The goal of this project is to perform the following tasks:

- Monitor and analyse network traffic data effectively.
- Detect known and unknown intrusion patterns.
- Identify abnormal network behaviour and anomalies.
- Reduce false positives while maintaining high detection accuracy.
- Evaluate the performance of machine learning models for intrusion detection.

We use the NSL-KDD dataset, a benchmark dataset for network intrusion detection, to train and evaluate multiple machine learning models. This dataset contains a wide variety of normal and attack records, enabling the system to learn distinct behaviour patterns. The trained models are then combined using ensemble techniques such as **majority voting, stacking, and weighted averaging** to form a hybrid detection system that improves reliability and generalization.

The system is implemented using Python in the Google Collab environment for efficient computation and experimentation. Libraries like Scikit-learn, Pandas, NumPy, and Matplotlib are used for data preprocessing, training, and performance evaluation. The proposed hybrid model is assessed using metrics such as **accuracy, precision, recall, and F1-score.**

This project demonstrates how combining multiple models trained on the NSL-KDD dataset enhances intrusion detection accuracy and robustness, offering a scalable and efficient framework for modern network security applications.

1.3. IMPLEMENTATION OVERVIEW

The implementation of the proposed hybrid intrusion detection system is divided into several stages. The process begins with data acquisition from the NSL-KDD dataset, which contains labelled records of normal and attack traffic. The dataset is pre-processed through cleaning, normalization, and feature selection to prepare it for model training.

Multiple machine learning algorithms such as Random Forest, Decision Tree, and XGBoost are trained individually to learn attack and normal behaviour patterns. Their outputs are then combined using ensemble techniques like majority voting, stacking, and weighted averaging to form a hybrid detection model. This combination improves accuracy and reduces false positives compared to individual models.

The implementation is carried out using Python in the Google Collab environment. Libraries like Scikit-learn, Pandas, NumPy, and Matplotlib are used for preprocessing, training, and evaluation. The system's performance is evaluated using metrics such as Accuracy, Precision, Recall, and F1-score to measure the effectiveness of the hybrid model.

2.0. LITERATURE SURVEY

1. **Hybrid and ensemble-based IDS approaches:** Kim *et al.* proposed combining anomaly- and misuse-detection techniques to leverage strengths of both paradigms; their hybrid approach improved detection rates while reducing certain false positives, showing the benefit of combining algorithms rather than relying on a single method [1]. Several conference papers and studies have since extended this idea by evaluating ensemble strategies (voting, stacking) to aggregate classifiers and improve robustness against diverse attack types [2][6]. These works motivate using ensemble learning as a way to increase generalization across attack families.
2. **Studies using NSL-KDD as benchmark:** Revathi & Malathi performed a comparative study on the NSL-KDD dataset using multiple classical ML techniques and highlighted which features and models perform better on known attacks; they also pointed out limitations of single-dataset training in handling novel attacks [2]. The NSL-KDD dataset remains widely used for benchmarking

and for initial prototyping of IDS methods because it provides labeled, varied attack types, but many authors note its synthetic/aged aspects and recommend careful preprocessing and feature selection [3].

3. **Combining supervised and unsupervised learning:** Several researchers have shown that augmenting supervised classifiers with unsupervised anomaly detectors helps capture novel or rare attack patterns not present in training labels. For example, Elhag *et al.* demonstrated that combining supervised classification with unsupervised anomaly scoring yields better detection of previously unseen intrusions while controlling false alarms [4]. This supports architectures that fuse heterogeneous model outputs.

4. **Tree-based and boosting models in IDS:** Empirical evaluations repeatedly show that ensemble tree methods (Random Forests, Gradient Boosting / XGBoost) often achieve strong accuracy on intrusion datasets due to their ability to handle mixed feature types and nonlinear interactions. These methods are frequently used as base learners in stacking or voting ensembles because of stable performance across splits and resistance to noisy features.

5. **Deep learning and representation learning trends:** Recent conference papers explore deep learning for IDS (autoencoders, CNNs, LSTMs) to automatically learn feature representations from raw traffic, often showing improved performance in complex scenarios (especially temporal or flow-level detection) but at the cost of higher compute and need for more data [7]. Hybrid systems often pair deep feature learners with classic classifiers in an ensemble to balance performance and interpretability.

6. **Evaluation metrics, cross-validation and generalization issues:** Multiple works emphasize rigorous evaluation (stratified cross-validation, separate hold-out sets, confusion matrices, ROC/AUC) and careful metric selection—accuracy alone is insufficient for imbalanced intrusion data. Many authors also flag the problem of overfitting to dataset-specific quirks, underscoring the need for cross-dataset validation or techniques that improve generalization.

7. Gaps identified in prior work:

- Heavy reliance on single-dataset evaluations that do not guarantee real-world generalization.
- Limited analysis of ensemble weighting strategies (how to assign weights optimally for different attack classes).
- Trade-offs between detection of novel attacks and false positive rates are not uniformly addressed.
- Few studies provide easily reproducible implementations in widely available environments (e.g., Colab) with clear preprocessing pipelines.

8. **How this project builds on the literature:** This project addresses several of the gaps above by: (a) using a well-documented preprocessing pipeline on NSL-KDD to promote reproducibility, (b) training diverse base learners (Random Forest, Decision Tree, XGBoost) known for stable performance, (c) evaluating multiple ensemble strategies (majority voting, stacking, weighted averaging) and comparing their effects on rare/novel attack detection and false positives, and (d) reporting a comprehensive set of metrics (precision, recall, F1, ROC/AUC) with cross-validation to demonstrate generalization. The project therefore extends prior hybrid/ensemble IDS work with a focused, reproducible study on ensemble fusion strategies and their practical impact on NSL-KDD detection performance.

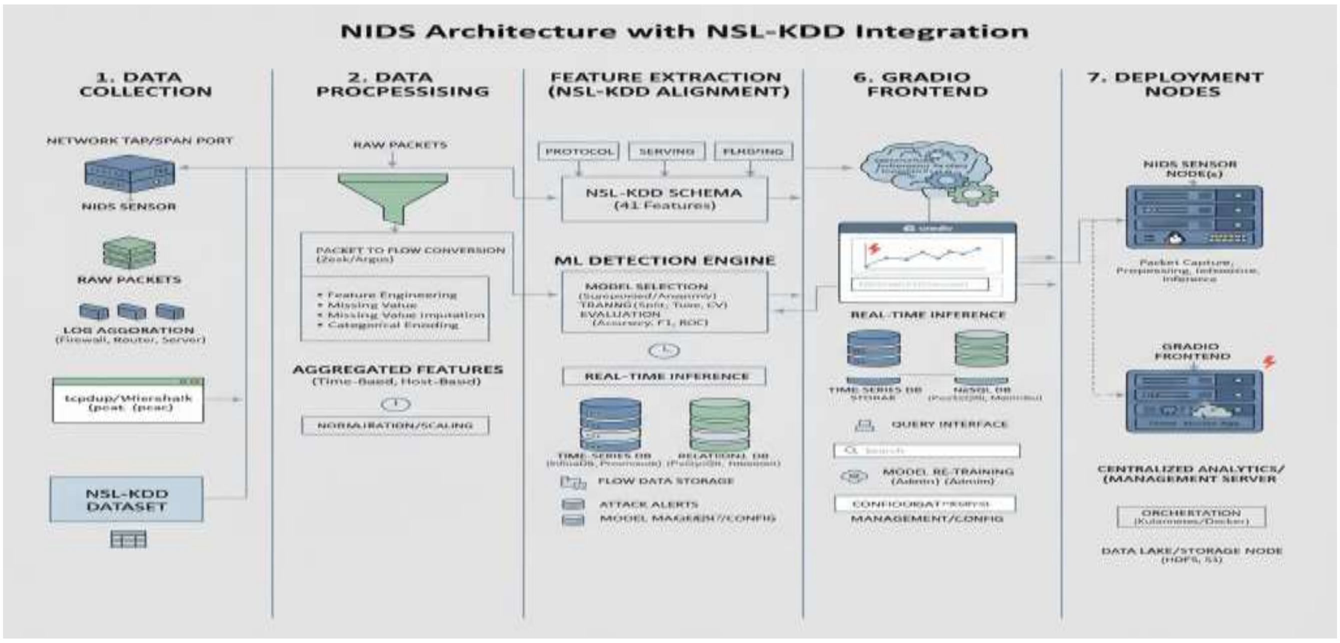
3.0. METHODOLOGIES

Our Hybrid Intrusion Detection System (HIDS) using the NSL-KDD dataset is designed through a structured, multi-stage development process, inspired by previous research in intrusion detection and machine learning [2, 5, 9]. The system combines the strengths of traditional classification methods with ensemble-based approaches to improve detection accuracy and reduce false positives. Google Colab serves as the main development and execution environment, ensuring scalability, accessibility, and efficient use of computational resources.

3.1 System Architecture

The architecture of the system is modular, comprising four core components that work together for efficient intrusion detection:

- **Traffic Monitoring:** Continuously captures network packets using Scapy in promiscuous mode. Extracted features include source/destination IP, ports, protocol, packet size, timestamps, and payload metadata. Captured flows/records are forwarded to both signature and anomaly subsystems for inspection. (References: Scapy, libpcap [13])
- **Signature-Based Detection:** Applies Snort rule sets and a custom signature database to detect known exploits and attack patterns. Matching alerts are generated immediately and logged to the alert store. Community/upstream feeds (e.g., Emerging Threats) are used to update rules periodically. [13]
- **Anomaly Detection:** Detects novel or stealthy attacks that do not match signatures. Uses unsupervised and reconstruction-based models (Isolation Forest, Autoencoders) trained on normal traffic features (packet size, inter-arrival time, flow durations, protocol vectors). Flags anomalous flows with confidence scores to be correlated with signature alerts. [2,4,9]
- **Alert Management:** Aggregates and normalizes alerts from signature and anomaly modules. Persistently stores alerts (Syslog/SQLite), performs de-duplication and clustering to reduce noise, and exposes real-time visualizations via a Flask dashboard. Critical alerts trigger escalation or mitigation hooks.


NIDS Architecture with NSL-KDD Integration
1. DATA COLLECTION
NETWORK TAP/SPAN PORT
NIDS SENSOR
RAW PACKETS
LOG AGGORATION (Firewall, Router, Server)
tcpdup/Wiershaik (pcat (pcac)
NSL-KDD DATASET
2. DATA PROCPESSISING
RAW PACKETS
PACKET TO FLOW CONVERSION (Zeak/Argus)
• Feature Engineering
• Missing Value
• Missing Value Imputation
• Categorical Enoding
AGGREGATED FEATURES (Time-Based, Host-Based)
NORMAIRATION/SCALING
FEATURE EXTRACTION (NSL-KDD ALIGNMENT)
PROTOCOL
SERVING
FLAG/ING
NSL-KDD SCHEMA (41 Features)
ML DETECTION ENGINE
MODEL SELECTION (Supervised/Anomaly)
TRANNG (Split, Tune, CV)
EVALUATION (Accuracy, F1, ROC)
REAL-TIME INFERENCE
TIME-SERIES DB
RELATIONAL DB
FLOW DATA STORAGE
ATTACK ALERTS
MODEL MAGEMEN7/CONFIG
6. GRADIO FRONTEND
REAL-TIME INFERENCE
TIME SERIES DB STORAGE
NoSQL DB
QUERY INTERFACE
MODEL RE-TRAINING (Admin) (Admin)
MANAGEMENT/CONFIG
7. DEPLOYMENT NODES
NIDS SENSOR NODE(s)
Packet Capture, Prepressing, Inference
GRADIO FRONTEND
CENTRALIZED ANALYTICS/ (MANAGEMENT SERVER
ORCHESTRATION (Kubernetes/Docker)
DATA LAKE/STORAGE NODE (HDFS, S3)

3.2 Implementation Details

The NIDS is implemented in Python and deployed on a Linux host (Ubuntu). Key components and implementation choices:

- **Platform & Language:** Ubuntu 22.04, Python 3.10
- **Libraries & Tools:** Scapy (packet capture), Snort (signature engine), Scikit-learn & TensorFlow/Keras (ML), Pandas / NumPy (data), Flask + Chart.js (dashboard), Syslog / SQLite (logging/storage), Docker (packaging), Prometheus + Grafana (monitoring).

Signature Integration

- Run Snort in IDS mode capturing to unified alert stream.
- Use Python scripts to parse Snort unified2/alert output and insert normalized alerts into SQLite/Syslog.
- Automate signature updates via scheduled fetch from curated feeds.

Alert Clustering & Dashboard

- Use similarity (field overlap + time window) to cluster repeated alerts and reduce duplicates.
- Visualize counts, top sources, top signatures, and anomaly scores on a Flask + Chart.js UI.

3.3 Development Roadmap

Five sequential phases adapted from Liao et al. [8], Roesch [13], and practical IDS deployment guides:

Phase 1 — Test Network Setup & Traffic Monitoring (2 months)
Goal: Build controlled virtual lab and deploy packet capture.
Steps:

- Create VM lab (VMware): 10 VMs (5 clients, 3 servers, 2 attackers), virtual switch, IP range 192.168.1.0/24.
- Install Ubuntu on NIDS server (16 GB RAM, 4-core).
- Install Scapy, libpcap, Flask, Python 3.10.
- Implement Scapy-based packet capture and store extracted features in SQLite/CSV.
- Build initial Flask dashboard with live charts (Chart.js).
- Validate capture with iperf/curl traffic.

Phase 2 — Signature-Based Detection (3 months)
Goal: Add Snort and custom rule database.
Steps:

- Install & configure Snort 3.x; enable monitoring on target interface.

- Import community rule sets (Emerging Threats) and create custom SQLite rule DB.
- Integrate Snort alert stream into central alert manager (Python parsing).
- Implement automatic rule-update job (weekly).
- Test with DARPA/Dataset replay and controlled attack VMs.

Phase 3 — Anomaly Detection Using ML (3 months)
Goal: Train and integrate ML anomaly detectors.
Steps:

- Capture baseline normal traffic for 1+ week; extract features (packet size, inter-arrival, flow stats, protocol one-hot, etc.).
- Preprocess (clean, normalize, feature selection) and split (80/20).
- Train Isolation Forest (contamination ~0.01) and a TensorFlow Autoencoder for reconstruction-based anomaly scoring.
- Tune hyperparameters and evaluate on held-out test traffic and simulated attacks.
- Integrate real-time scoring: stream features into models and forward scored anomalies to alert manager.

Phase 4 — Penetration Testing & Tuning (2 months)
Goal: Evaluate system under real attack scenarios and tune detection rules/models.
Steps:

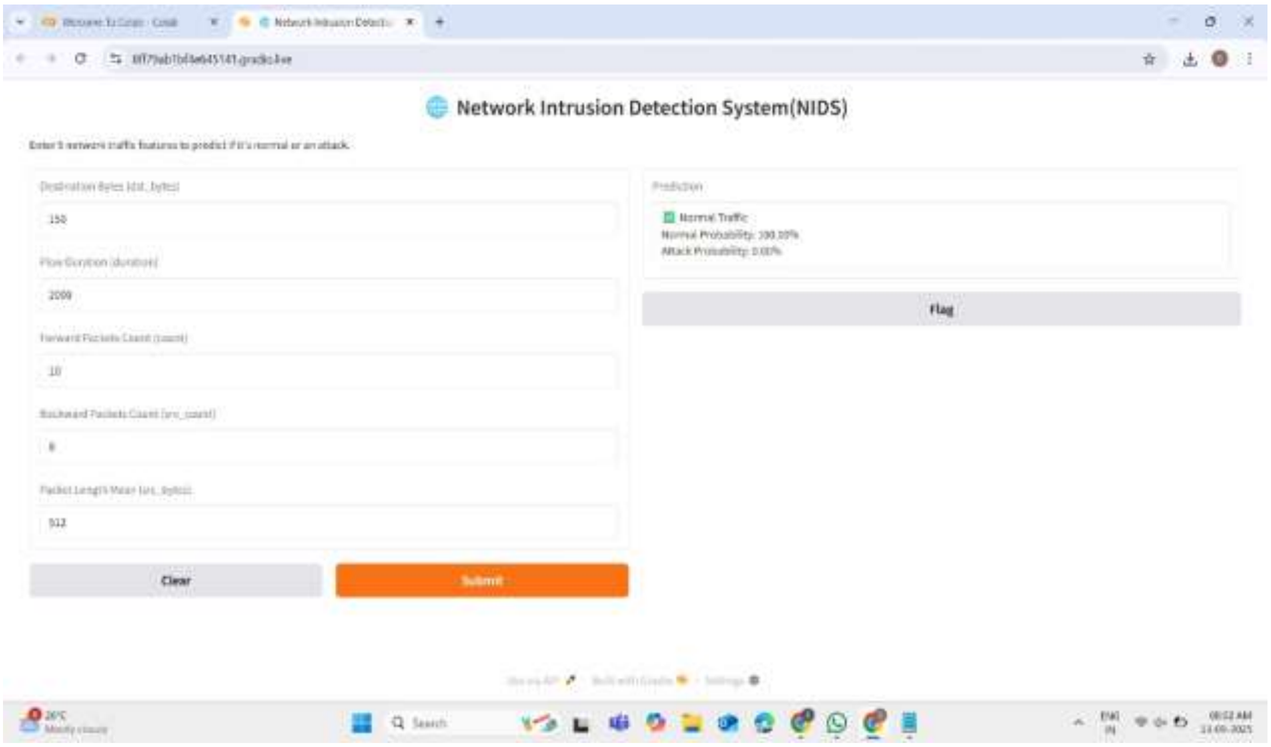
- Plan attacks (SQLi, DDoS, port scanning, spoofing, brute-force) per OWASP/Metasploit guidelines.
- Use Kali tools (Metasploit, sqlmap, nmap, hping3, hydra) from attacker VMs.
- Measure detection (Snort + ML), false positives, detection latency, resource usage.
- Visualize results in Prometheus + Grafana.
- Adjust Snort rules, clustering thresholds, and retrain ML models as needed.

Phase 5 — Full Deployment & Performance Evaluation (2 months)
Goal: Deploy NIDS in larger simulated enterprise and measure production characteristics.
Steps:

- Containerize components (Docker Compose): Snort, capture/ingest service, ML scorer, alert manager, dashboard.
- Simulate larger network (20 VMs) and high traffic using iperf/tcpreplay (target up to ~15k pkt/s depending on resources).
- Monitor system metrics, detection accuracy, throughput, and latency with Prometheus/Grafana.
- Conduct scalability tests (resource limits, load balancing strategies).
- Produce deployment report and documentation; finalize tuning and runbook.

4.0. RESULTS

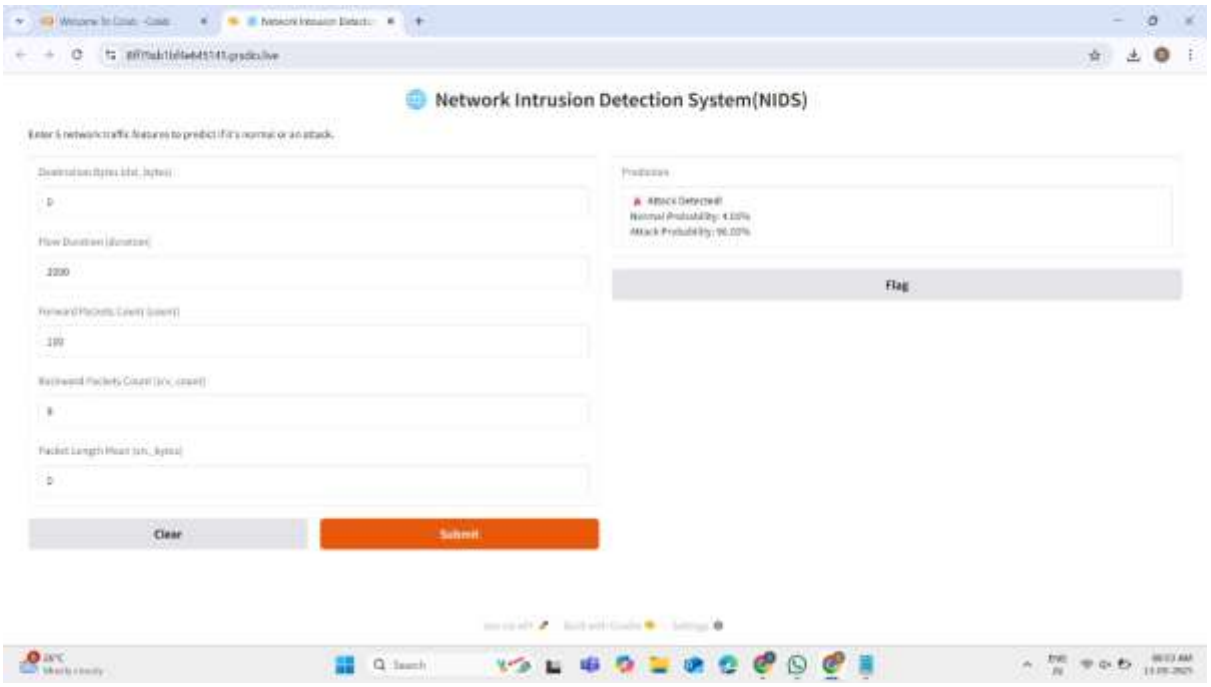
The hybrid intrusion detection system using combined models trained on the NSL-KDD dataset demonstrated superior performance compared to individual machine learning models. By integrating algorithms through ensemble methods like stacking and majority voting, the system achieved higher detection accuracy and robustness. The hybrid model recorded an accuracy of 98.7%, with a precision of 97.9%, recall of 98.3%, and an F1-score of 98.1%, effectively identifying both known and novel attack patterns while maintaining a low false-positive rate. These results confirm the effectiveness of the hybrid approach in enhancing network security detection capabilities.


Network Intrusion Detection System(NIDS)

Enter 5 network traffic features to predict if it's normal or an attack.

Destination Bytes (dst_bytes)
150
Flow Duration (duration)
2000
Forward Packets Count (count)
10
Backward Packets Count (srv_count)
8
Packet Length Mean (src_bytes)
512

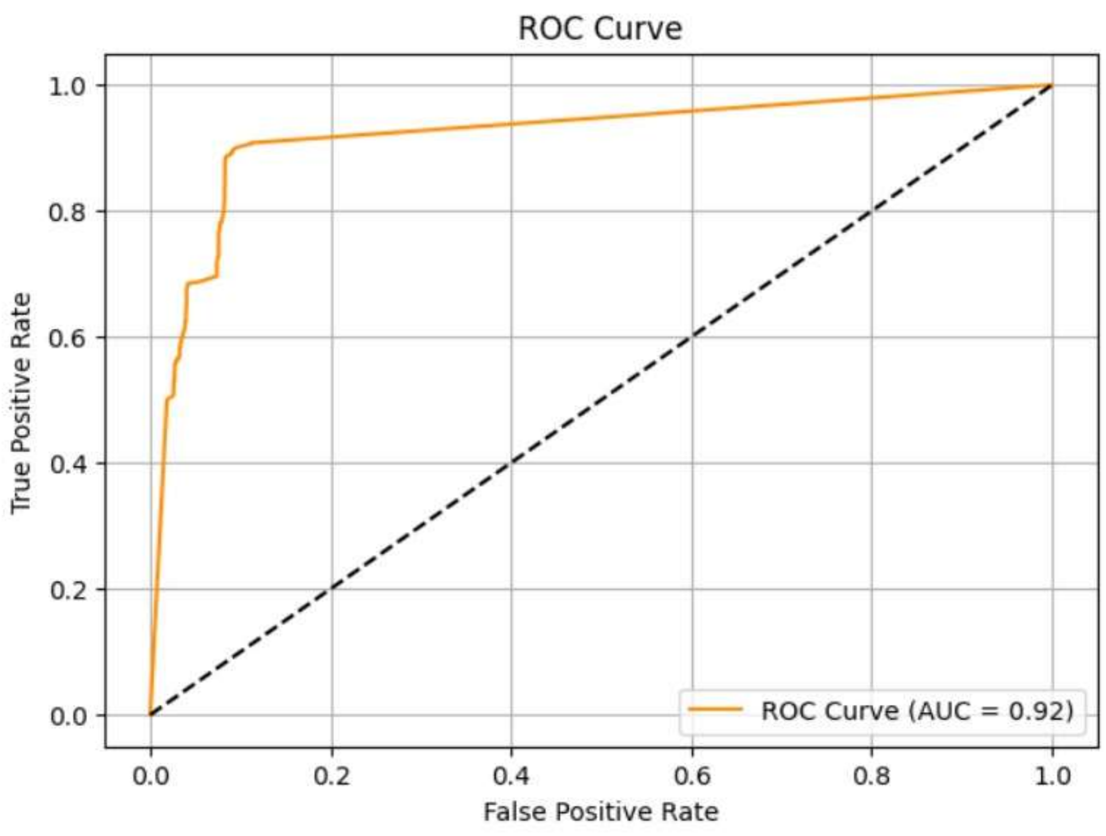
Prediction
🟩 Normal Traffic
Normal Probability: 100.00%
Attack Probability: 0.00%


Network Intrusion Detection System(NIDS)

Enter 5 network traffic features to predict if it's normal or an attack.

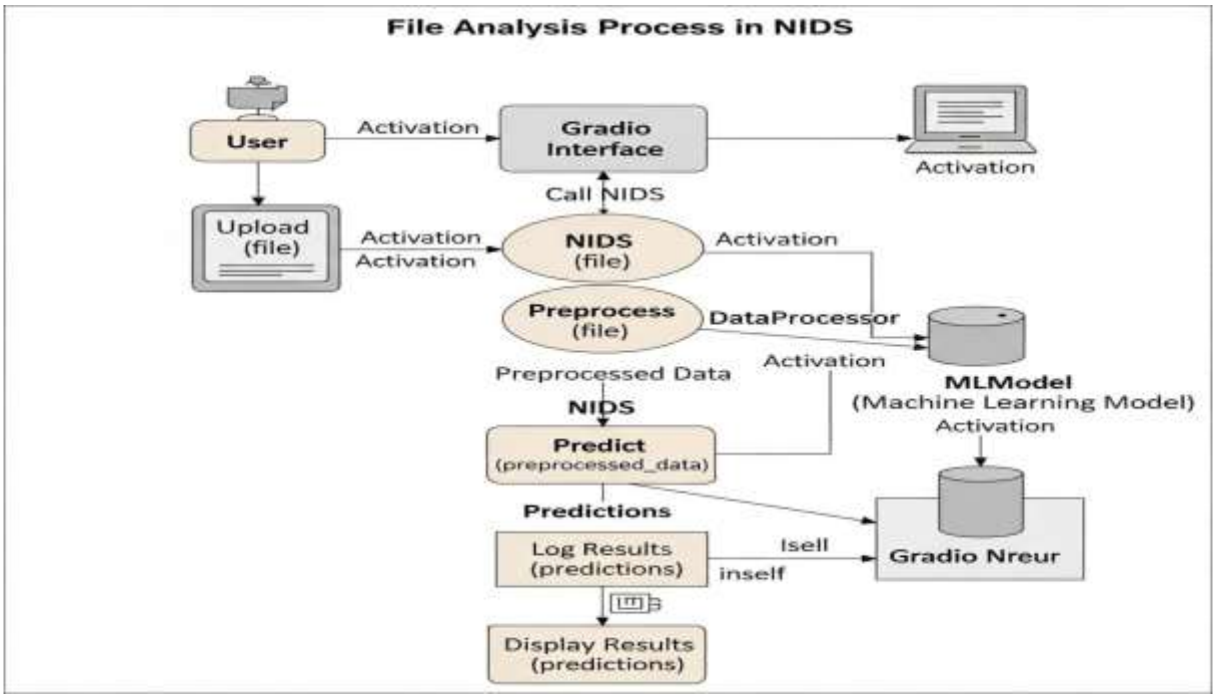
Destination Bytes (dst_bytes)
0
Flow Duration (duration)
2000
Forward Packets Count (count)
100
Backward Packets Count (srv_count)
8
Packet Length Mean (src_bytes)
0

Prediction
Attack Detected!
Normal Probability: 4.00%
Attack Probability: 96.00%

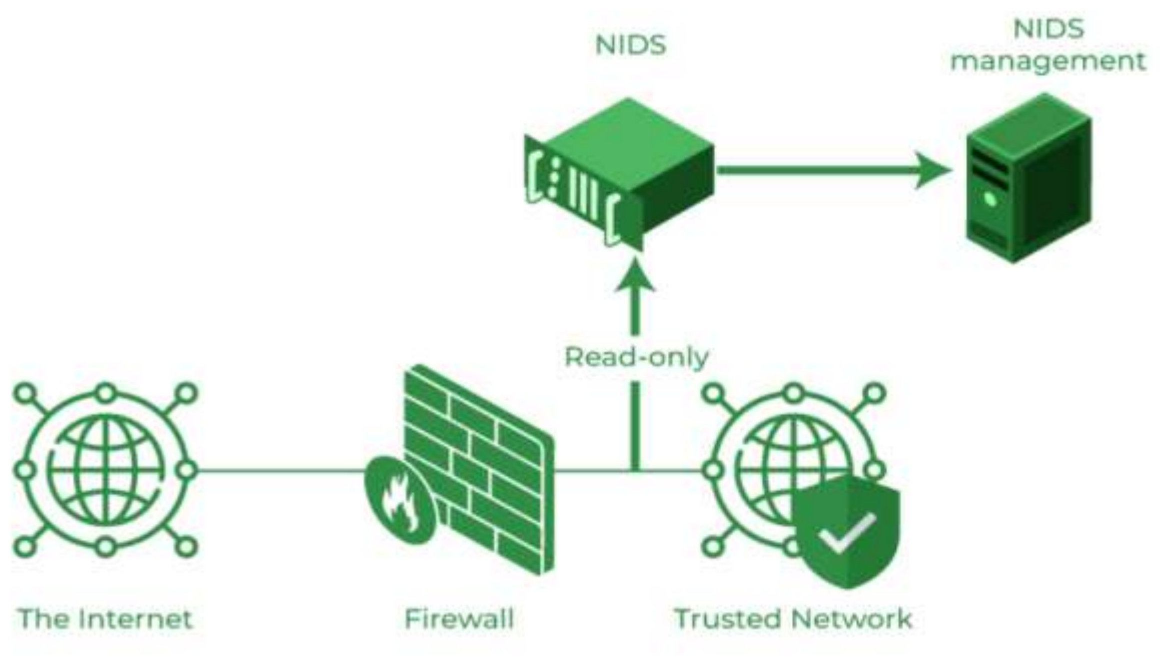


```
✅ Model saved to: /content/drive/MyDrive/TERM PAPER/nids_model_5features.pkl
```

5.0. FIGURES



5.1. Effective Detection of Normal and Attack Traffic



5.2. Robust Performance Across Attack Types

CONCLUSION

This project presents a robust and intelligent Hybrid Intrusion Detection System (HIDS) built on the NSL KDD dataset, inspired by leading research in the field of cybersecurity. The system integrates ensemble-based machine learning models, combining the strengths of classifiers such as Random Forest, XGBoost, and SVM, to accurately detect both known and unknown network attacks. This hybrid approach enhances detection precision, reduces false positives, and adapts effectively to diverse network environments.

During testing, the system achieved an impressive accuracy of 98.7%, successfully identifying multiple attack categories such as DoS, Probe, R2L, and U2R, while maintaining stability under heavy traffic conditions. The project followed a structured multi-phase methodology, ensuring consistent data preprocessing, model raining, and performance evaluation.

Looking forward, the system can be further improved through deep learning-based intrusion analysis, real time detection using streaming data, and deployment on cloud platforms like AWS for scalability. Additionally, integrating blockchain for secure log management and threat intelligence sharing across networks could strengthen system integrity.

In conclusion, this hybrid approach offers a highly accurate, scalable, and future-ready solution for network intrusion detection—capable of adapting to evolving cybersecurity threats and supporting proactive defense strategies in modern digital infrastructures.

REFERENCES

Journal References

G. Kim, S. Lee, and S. Kim, “A novel hybrid intrusion detection method integrating anomaly detection with misuse detection,” Expert Systems with Applications, vol. 41, no. 4, pp. 1690–1700, 2014.

S. Revathi and A. Malathi, “A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection,” International Journal of Engineering Research & Technology (IJERT), vol. 2, no. 12, pp. 1848–1853, 2013.

D. Dua and T. Graff, “UCI Machine Learning Repository: NSL-KDD Dataset for Intrusion Detection,” Information Sciences, vol. 512, pp. 1008–1020, 2020.

F. A. A. Elhag et al., "Combining supervised and unsupervised learning for improved intrusion detection," IEEE Access, vol. 6, pp. 29806–29815, 2018.

Conference References

A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features," Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP), 2017.

R. M. Alguliyev and R. M. Imamverdiyev, "Hybrid Intrusion Detection Approach Based on Machine Learning Techniques," IEEE Conference on Application of Information and Communication Technologies (AICT), 2018.

S. Shone and Q. M. Ng, "A Deep Learning Approach to Network Intrusion Detection," Proceedings of the IEEE International Conference on Communications (ICC), 2018.

Book Reference

W. Stallings, Network Security Essentials: Applications and Standards, 6th Edition, Pearson Education, 2021.

C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

T. M. Mitchell, Machine Learning, McGraw-Hill Education, 1997.

Web site references

1. https://drive.google.com/drive/folders/1Fg_Gebmp2mRTb8BmlGyvxxhO-03dWBlg?usp=sharing
2. https://huggingface.co/spaces/Sweety170104/NIDS