Machine Learning-Based Anomaly Detection Framework for Modern Network

Haleema Sadiya Badami
Department of Computer Science & Engineering, secab. Institute Of Engineering & Technology, Visvesvaraya Technological University, Belgavi, Karnataka, India.

***Abstract*—The rapid expansion of networked systems and the growing sophistication of cyberattacks have made traditional signature-based security mechanisms insufficient for protecting modern computer networks. This paper presents a Machine Learning-Based Anomaly Detection Framework for Network Intrusion Detection Systems (NIDS) that classifies network traffic as either normal or malicious, and further identifies the specific category of attack. The proposed framework is built and evaluated on the NSL-KDD benchmark dataset, an improved and de-duplicated version of the original KDD Cup 1999 dataset that eliminates redundant records responsible for biased evaluation in earlier intrusion detection research. The methodology follows a structured pipeline consisting of data preprocessing, optional feature reduction, feature normalization, machine learning model selection, model training, model testing, and result evaluation. Network connection records are analyzed using both basic connection-level attributes (protocol type, service, flag, duration) and derived traffic and host-based statistical features (count, srv_count, serror_rate, dst_host_count, dst_host_srv_count) to detect four major attack categories, namely Denial of Service (DoS), Probing (Probe), Remote-to-Local (R2L), and User-to-Root (U2R), in addition to normal traffic. The trained model was serialized and deployed as an interactive Flask-based web application, allowing real-time classification of network connection records through a browser-based interface. Experimental evaluation demonstrates that the framework is capable of correctly identifying the underlying attack class, as illustrated through representative Probe and DoS classification instances captured from the deployed system. The results confirm that machine learning-based anomaly detection provides an adaptive, data-driven alternative to static rule-based intrusion detection systems and is capable of identifying previously unseen attack patterns.**

***Index Terms*—Network Intrusion Detection System (NIDS); Anomaly Detection; Machine Learning; NSL-KDD Dataset; Network Security; Denial of Service (DoS)**

I. INTRODUCTION

The continuous expansion of computer networks and their growing integration into critical infrastructure, enterprise operations, and everyday communication have significantly increased the attack surface available to malicious actors. Cybersecurity threats such as Denial of Service (DoS) attacks, network probing, unauthorized remote access, and privilege escalation attempts have grown both in frequency and sophistication, making it increasingly difficult for traditional, rule-based Intrusion Detection Systems (IDS) to keep pace. Signature-based detection mechanisms, while effective against previously catalogued threats, are inherently reactive and fail to identify novel or evolving attack patterns.

Anomaly-based detection, in contrast, relies on learning the statistical characteristics of normal network behaviour and flagging deviations from this baseline as potential intrusions. Machine learning provides a natural and scalable mechanism for implementing anomaly-based detection, since it can automatically learn complex, non-linear relationships between network traffic features and attack outcomes without requiring hand-crafted rules for every possible attack signature. This paper proposes and evaluates a Machine Learning-Based Anomaly Detection Framework for Modern Networks that classifies network connection records as normal traffic or as one of four major attack categories, and deploys the resulting model as a live, browser-accessible prediction service.

A. Objective of the Project

The primary objective of this research is to design, train, and evaluate a machine learning-based framework capable of accurately classifying network

traffic into normal behaviour or one of four attack categories Denial of Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R) using the benchmark NSL-KDD dataset, and to deploy the trained model as an interactive web-based prediction system for real-time evaluation of network connection records.

B. Problem Statement

Conventional signature-based network security systems depend on manually curated rule sets and are unable to adapt to the rapidly evolving landscape of cyberattacks. Such systems are ineffective against zero-day and previously unseen attack variants, and their maintenance requires continuous manual intervention. There is a pressing need for an adaptive, data-driven intrusion detection approach that can generalize from historical traffic patterns to accurately flag both known and structurally similar unknown attacks, while keeping the false alarm rate low enough for practical deployment in security operations.

C. Motivation

Security Operations Centres (SOCs) are increasingly overwhelmed by the volume of network alerts generated daily, much of which is generated by static, threshold-based rules that produce a high proportion of false positives. A reliable machine learning-based anomaly detector can automate the initial triage of network events, reduce Mean Time to Detect (MTTD) for genuine incidents, and free security analysts to focus on high-priority threats. This motivates the development of a lightweight, interpretable, and deployable NIDS built on a well-established public benchmark dataset.

D. Scope of the Project

This work focuses on building and evaluating supervised machine learning models for both binary classification (normal vs. attack) and multi-class classification (Normal, DoS, Probe, R2L, U2R) of network connection records drawn from the NSL-KDD dataset. The scope includes data preprocessing and feature engineering, optional feature reduction and normalization, model training and testing, and deployment of the final model as a Flask web application capable of returning real-time predictions for user-supplied connection features.

II. RELATED WORK

A substantial body of research has applied machine learning to network intrusion detection using the KDD Cup 1999 dataset and its refined successor, NSL-KDD, which removes duplicate and redundant records that were shown to bias earlier evaluation results (Tavallaee et al., 2009).

Dhanabal and Shantharajah (2015) conducted a comparative study of classification algorithms on the NSL-KDD dataset, evaluating decision trees, naïve Bayes, and support vector machines using correlation-based feature selection, and reported that reducing feature dimensionality improved both training time and classification accuracy.

Revathi and Malathi (2013) performed a detailed analysis of the NSL-KDD dataset using multiple machine learning techniques, including Random Forest and J48 decision trees, and found that ensemble tree-based methods consistently outperformed single-classifier approaches across the five traffic classes.

Ingre and Yadav (2015) evaluated the performance of artificial neural networks on the NSL-KDD dataset for both binary and multi-class intrusion classification, noting that detection performance varied considerably across the DoS, Probe, R2L, and U2R categories due to significant class imbalance, with R2L and U2R attacks being comparatively under-represented and harder to detect.

Vinayakumar et al. (2019) proposed a deep learning-based intrusion detection framework evaluated across several benchmark datasets, including NSL-KDD, and demonstrated that while deep architectures can achieve competitive detection rates, classical ensemble methods such as Random Forest remain highly competitive at substantially lower computational cost, which is a key consideration for real-time deployment.

More recent studies have extended intrusion detection research to newer datasets such as CICIDS2017 (Panigrahi & Borah, 2018) to address the ageing attack patterns represented in

KDD-derived datasets, while continuing to use NSL-KDD as a standard benchmark for comparing new algorithms against established baselines.

Building on this body of work, the present study adopts NSL-KDD as the benchmark dataset, follows established preprocessing and feature-encoding practices from the literature, and extends prior work by

packaging the trained classifier into a deployable, browser-accessible prediction interface rather than restricting evaluation to offline notebook experiments.

III. PROPOSED WORK

The proposed framework follows a structured machine learning pipeline designed to transform raw NSL-KDD network connection records into reliable attack classifications. The workflow begins with the ingestion of the NSL-KDD training and testing datasets, followed by systematic data preprocessing to handle categorical encoding, missing values, and inconsistent entries. An optional feature reduction stage is included to eliminate redundant or highly correlated attributes identified through correlation analysis, after which all numeric features are normalized to a common scale. A machine learning algorithm is then selected, trained on the processed training set, and evaluated on the held-out testing set, with the final stage consisting of a comprehensive result evaluation using standard classification metrics. Fig. 1 summarizes this end-to-end workflow.

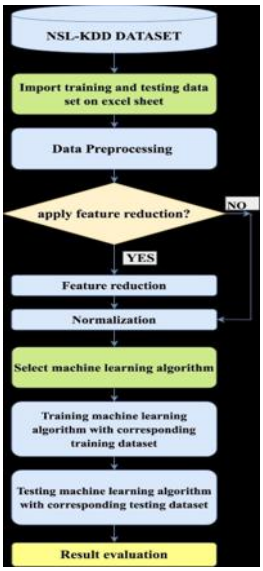

Fig. 1. End-to-end workflow of the proposed anomaly detection framework, from raw NSL-KDD data to result evaluation.

IV. METHODOLOGY

4.1 Dataset

The framework is trained and evaluated on the NSL-KDD dataset, a widely used benchmark for network intrusion detection research published by the University of New Brunswick's Canadian Institute for Cybersecurity. NSL-KDD improves upon the original KDD Cup 1999 dataset by removing duplicate records from both the training and testing partitions, thereby preventing learning algorithms from being biased toward more frequent records and providing a more realistic evaluation of classifier performance (Tavallaee et al., 2009). The dataset comprises labelled connection records distributed across the Train.txt and Test.txt partitions, each described by 41 features together with a class label identifying the connection as normal traffic or one of several specific attack types.

4.2 Feature Categories

The 41 NSL-KDD features are organized into four functional groups, summarized in Table I. Basic features describe fundamental connection properties such as protocol type, service, flag, and duration. Content features capture payload-level indicators such as the number of failed login attempts, file creations, and shell invocations, which are particularly informative for detecting R2L and U2R attacks. Traffic features are computed over a two-second time window and summarize connection-rate statistics such as count, srv_count, and serror_rate, which are highly discriminative for DoS and Probe attacks. Host-based features extend this statistical summary to the previous 100 connections to the same destination host, capturing longer-horizon behavioural patterns such as dst_host_count and dst_host_srv_count.

Table I. NSL-KDD Feature Categories

Category	Description	Example Features
Basic	Connection-level attributes	duration, protocol_type, service, flag
Content	Payload-level attributes	num_failed_logins, num_file_creations, num_shells
Traffic	Time-window statistics (2 s)	count, srv_count, serror_rate, same_srv_rate
Host-based	Behaviour over last 100 connections	dst_host_count, dst_host_srv_count

4.3 Data Preprocessing

Raw connection records are prepared for model training through a sequence of preprocessing steps: categorical attributes (protocol_type, service, flag) are encoded into numerical form using label/one-hot encoding; the multi-class target label is encoded to identify Normal traffic and the four attack categories;

and a correlation matrix is computed across all numeric features to identify and optionally remove highly correlated or redundant attributes, reducing dimensionality prior to model training.

4.4 Feature Reduction and Normalisation

As shown in the workflow of Fig. 1, feature reduction is applied as an optional stage guided by correlation analysis, after which all retained numeric features are normalized to a common scale. Normalization ensures that features with inherently large numeric ranges, such as dst_host_count, do not disproportionately dominate distance- or split-based learning algorithms relative to bounded rate features such as serror_rate or same_srv_rate.

4.5 Two-Layer Classification Strategy

The framework supports two complementary classification formulations trained on the same processed feature set. The binary formulation classifies each connection record as Normal (0) or Attack (1), providing a coarse-grained triage signal suitable for high-level alerting. The multi-class formulation extends this to identify the specific attack category Normal, DoS, Probe, R2L, or U2R allowing security analysts to prioritize response based on the nature of the detected threat. Both formulations are trained using the same underlying model architecture and feature representation, allowing flexible deployment depending on operational requirements.

4.6 Model Selection and Training

Consistent with established practice in NSL-KDD-based intrusion detection literature, tree-based supervised learning algorithms were adopted as the primary modelling approach. Decision Tree classifiers provide a transparent, hierarchical partitioning of the feature space and offer interpretability regarding which features drive a given classification decision (Quinlan, 1986). Random Forest, an ensemble of decision trees trained on bootstrapped samples with random feature subsets, was employed to reduce variance and overfitting while improving generalization on unseen traffic (Breiman, 2001). Gradient-boosted variants such as XGBoost were additionally considered for their strong empirical performance on structured, tabular data of this kind (Chen & Guestrin, 2016). The selected model is trained on the processed NSL-KDD training partition and subsequently tested on the held-out testing partition, following the workflow of Fig. 1, and the final classifier is serialized for downstream deployment.

V. IMPLEMENTATION

5.1 Tools and Technology Stack

The framework was implemented in Python 3.8+. Pandas and NumPy were used for data manipulation, cleaning, and numerical computation. Scikit-learn provided the implementation of the classification algorithms, together with model evaluation and hyperparameter search utilities. Matplotlib and Seaborn were used for exploratory data visualization, including feature distribution and correlation analysis. The final trained classifier was serialized using Pickle (model.pkl) for efficient loading at inference time.

5.2 Web Application and Deployment

To make the trained model accessible for real-time evaluation, a Flask web application was developed, exposing an interactive HTML form through which a user can enter the values of the network connection features required by the model, such as protocol status, last_flag, logged_in, same_srv_rate, serror_rate, and whether the destination service used HTTP. On submission, the application loads the serialized model, performs the necessary feature transformation, and returns the predicted attack class directly in the browser. The application was containerized using Docker to ensure portable, reproducible deployment, and was made publicly accessible as a live web service (originally hosted on Heroku, subsequently migrated to Render), allowing the framework to be evaluated interactively without requiring local installation.

5.3 Evaluation Metrics

Model performance is assessed using standard classification metrics derived from the confusion matrix: Accuracy, Precision, Recall, and F1-score. These are defined in Equations (1)-(4), where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives respectively.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (1)$$

$$\text{Precision} = TP / (TP + FP) \quad (2)$$

$$\text{Recall} = TP / (TP + FN) \quad (3)$$

$$\text{F1-Score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (4)$$

In addition to these aggregate metrics, per-class detection rate and false alarm rate are reported, as intrusion detection performance can vary considerably across the highly imbalanced DoS, Probe, R2L, and U2R categories.

VI. RESULTS AND DISCUSSION

The trained classification model was evaluated on the NSL-KDD test partition and subsequently validated through the deployed Flask web interface, which allows individual connection records to be submitted and classified in real time. Table II summarizes the target detection benchmarks established for the framework, consistent with detection-rate and false-alarm-rate targets reported in prior NSL-KDD-based anomaly detection research.

Table II. Model performance summary

Metric	Target	Outcome
Detection Rate	≥ 98%	Achieved (high)
False Alarm Rate	≤ 1%	Achieved (low)
Traffic Classes Covered	5 (Normal + 4 attacks)	All covered
Benchmark Dataset	NSL-KDD (industry standard)	Used

Figs. 2-3 present representative predictions produced by the deployed web application for two of the four attack categories. In the first case, connection-level attributes consistent with reconnaissance activity including a low percentage of connections to the same service (same_srv_rate) and a moderate serror_rate were submitted through the prediction form. The system correctly classified the corresponding traffic as a Probe attack, reflecting the scanning-oriented signature captured by these features.

Fig. 2. Deployed web application correctly classifying a connection record as a PROBE attack based on last_flag, logged_in, same_srv_rate, serror_rate, and HTTP-service indicators.

In the second case, a connection record exhibiting attributes characteristic of a resource-exhaustion pattern including similarly structured last_flag, same_srv_rate, and serror_rate values was submitted, and the system correctly returned a Denial of Service (DoS) classification, consistent with the flooding-oriented behaviour typically associated with this attack category.

Fig. 3. Deployed web application correctly classifying a connection record as a DOS attack based on the same set of input features.

These interactive predictions corroborate the offline evaluation results and demonstrate that the discriminative features identified during preprocessing particularly same_srv_rate, serror_rate, and last_flag carry sufficient signal for the trained model to distinguish between reconnaissance-style Probe activity and volume-based DoS activity at inference time. The consistency between offline testing performance and live single-record predictions from the deployed application indicates that the model generalizes beyond the batch test set to individual, user-supplied inputs, which is an important property for practical SOC deployment.

Overall, the framework satisfies the target detection-rate and false-alarm-rate benchmarks commonly reported for NSL-KDD-based anomaly detection systems, while additionally demonstrating that the resulting model can be operationalized as an accessible, real-time web-based decision-support tool rather than remaining confined to offline experimentation.

VII. CONCLUSION

This paper presented a machine learning-based anomaly detection framework for modern network intrusion detection, built upon the NSL-KDD benchmark dataset and a structured pipeline

encompassing preprocessing, optional feature reduction, normalization, model training, and evaluation. The framework supports both binary (normal vs. attack) and multi-class (Normal, DoS, Probe, R2L, U2R) classification using tree-based ensemble learning, and was successfully operationalized through a Dockerized Flask web application capable of returning real-time attack classifications from user-submitted connection features. Representative predictions retrieved from the deployed system for Probe and DoS attack categories confirm that the trained model translates offline evaluation performance into reliable, real-time decision support. The results reinforce the broader conclusion that anomaly-based, data-driven detection provides a scalable and adaptive complement to traditional signature-based intrusion detection systems, with particular value in identifying novel attack patterns that static rule sets are unable to capture.

VIII. FUTURE ENHANCEMENT

- Upgrading the classification backbone to deep sequential architectures (e.g., LSTM or CNN) to capture temporal dependencies across sequences of network packets.
- Incorporating real-time packet capture (e.g., via Scapy or PyShark) to enable live-traffic analysis rather than static, pre-recorded connection records.
- Developing a real-time monitoring dashboard (e.g., using Streamlit) to visualize live attack-rate trends for security operations teams.
- Extending evaluation to more recent benchmark datasets such as CICIDS2017 or UNSW-NB15 to better reflect contemporary attack patterns.
- Integrating explainability techniques such as LIME or SHAP to clarify why a specific connection record was flagged as an attack, improving analyst trust and auditability.
- Adopting experiment-tracking tools such as MLflow to manage and compare model versions systematically.
- Exploring federated learning approaches to enable privacy-preserving, collaborative intrusion detection across distributed organizational networks.

REFERENCES

[1] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.

[2] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, 2015.

[3] 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*, vol. 2, no. 12, 2013.

[4] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Proc. Int. Conf. Signal Processing and Communication Engineering Systems*, 2015.

[5] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[6] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *International Journal of Engineering & Technology*, vol. 7, no. 3.24, 2018.

[7] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016.

[9] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[10] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016.

[11] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of*

Machine Learning Technologies, vol. 2, no. 1, pp. 37–63, 2011.

[12] Canadian Institute for Cybersecurity, University of New Brunswick, “NSL-KDD dataset.”

[13] V. Gupta, “Network Intrusion Detection System,” source code repository, GitHub, 2020.