International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

An Intelligent Framework for Network Traffic Intrusion Detection Using Machine Learning Techniques

Ms. Harshitha T A[1], Mr. Suthan R[2]

[1]MTech Student Department of Computer Science and Engineering Shridevi Institute of Engineering and Technology, Tumkur, India
2Assistant Professor, Department of Computer Science and Engineering Shridevi Institute of Engineering and Technology, Tumkur, India

ABSTRACT
The rapid evolution of large-scale network infrastructures has increased the difficulty of detecting cyber intrusions in real time. This paper proposes a machine learning–based Network Intrusion Detection System created utilizing the UNSW-NB15 dataset, which represents realistic network traffic and multiple attack categories. Data preprocessing techniques, including normalization, noise elimination, and class balance correction, are applied to improve learning reliability. Key traffic features are selected to reduce dimensional complexity and enhance computational efficiency. A Light Gradient Boosting Machine (LightGBM) classifier is employed due to its fast convergence, low resource consumption, and strong predictive capability. System Standard intrusion detection metrics are used to evaluate performance effective attack detection with a low false alarm rate, confirming the suitability of LightGBM for efficient and scalable real-time network security applications.

Keywords: Intrusion Detection System, Network Traffic Classification, Gradient Boosting, LightGBM, Cybersecurity Analytics, Data-Driven Security

1. INTRODUCTION

Networks for digital communication have evolved into fundamental to modern society, supporting continuous data exchange across sectors such as healthcare, finance, transportation, energy systems, and government services [3]. As network connectivity expands, the increasing volume and complexity of transmitted data have significantly raised exposure to cyber threats, making network security a critical concern for organizations worldwide [1].

The landscape of cyber threats has changed from simple, isolated attacks to sophisticated, fast-executing, and multi-stage intrusion strategies that aim to bypass conventional defenses [1]. Traditional security mechanisms, including firewalls, antivirus software, and access control systems, rely primarily on predefined rules and known signatures which restricts their efficiency against unknown or rapidly evolving attack patterns [2].

To enhance network protection, Intrusion Detection Systems (IDS) were introduced to monitor network traffic and identify suspicious or abnormal behavior in real time [3]. However, conventional IDS implementations depend heavily on manually crafted rules and static thresholds, reducing their adaptability

and leading to increased false alarms and missed attacks when facing dynamic adversarial behavior [1].
Machine learning–based intrusion detection has become a subject of considerable interest as it enables data-driven modeling of network behavior rather than reliance on fixed rules [4]. By learning complex and non-linear relationships from historical traffic data, machine learning models can detect subtle anomalies and generalize to previously unseen attacks, offering improved detection performance over traditional approaches [5].
The efficiency of machine learning–based IDS solutions strongly depends regarding the quality of training data, feature representation, and algorithm selection. Earlier benchmark datasets such as KDD'99 and NSL-KDD have been extensively employed, yet suffer from outdated traffic patterns, redundant records, and limited attack diversity, reducing their relevance for modern intrusion detection research [2].
To address these limitations, the UNSW-NB15 dataset was developed to provide realistic network traffic with a diverse set of contemporary attack categories generated using modern traffic simulation tools [5]. However, the dataset introduces challenges such as high dimensionality, mixed feature types, and class imbalance, necessitating efficient preprocessing and robust classification techniques [4].
Among recent machine learning algorithms, the Light Gradient Boosting Machine (LightGBM) has emerged as an effective solution due to its great precision, quick training pace, and ability to model complex feature interactions using a leaf-wise tree growth strategy [5]. Its scalability and efficiency make it particularly suitable for large-scale network traffic analysis where timely intrusion detection is essential.
Motivated by in light of these findings, this paper proposes an intelligent intrusion detection algorithm that incorporates the UNSW-NB15 dataset. with an optimized LightGBM classifier. The framework incorporates systematic data preprocessing, feature optimization, and class imbalance handling techniques. Experimental evaluation and comparison with baseline models demonstrate improved detection accuracy and reduced false alarms, highlighting the effectiveness of machine learning-driven intrusion detection systems for contemporary network security environments.

2. RELATED WORK

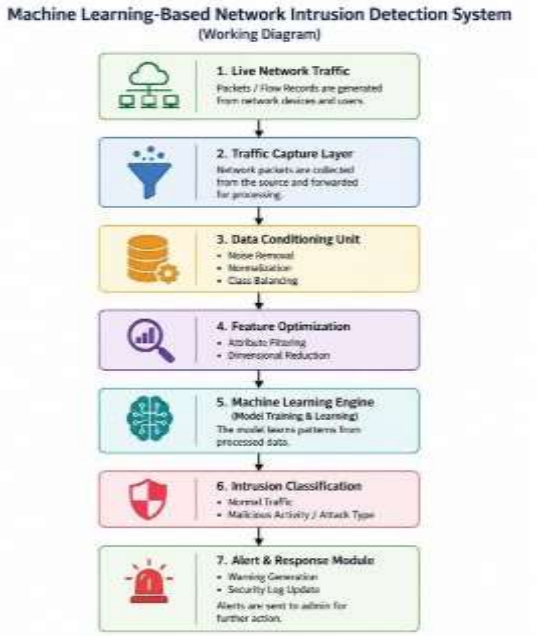
Recent research in network security has increasingly focused on modular and layered intrusion detection architectures that separate traffic acquisition, analysis, and response functions to improve scalability and detection efficiency [3]. Rather than relying on single-stage detection mechanisms, prior studies adopt stepwise processing pipelines that enhance interpretability and allow flexible integration of intelligent learning components. Figure X illustrates a generalized workflow commonly reported in learning-driven intrusion detection systems across the literature [1].
In most existing approaches, the detection The approach starts with ongoing observation of network activity, where communication events generated by users and devices are observed in real time. Network traffic is typically collected in the form of raw packets or aggregated flow-level records and forwarded to centralized processing units for further inspection [3]. This separation between traffic generation and analysis enables uninterrupted network operation while supporting comprehensive security monitoring [2].
Since Raw traffic data frequently includes noise, inconsistencies, and class imbalance Numerous studies highlight the significance of data preprocessing as a critical stage in intrusion detection systems [1]. Common preprocessing operations include normalization of numerical attributes, correction of anomalous values, and techniques to mitigate dominance of majority classes. Prior work has shown that effective data conditioning significantly improves learning stability and reduces false detections in large-scale network

environments [4].

Feature refinement is another widely discussed component in existing IDS frameworks. Instead of utilizing all available traffic attributes, many researchers focus on choosing characteristics that are most important in distinguishing normal and malicious behavior [5]. Feature selection and When exposed to a variety of changing attack patterns, dimensionality reduction approaches not only reduce computing overhead but also enhance machine learning models' capacity for generalization. [2].

At the core of these architectures lies the intelligent detection module, where statistical and machine learning algorithms are trained to identify intrusion characteristics. Previous studies have explored a variety of classifiers, incorporating conventional machine learning models and neural network–based approaches, enabling systems to learn complex and non-linear relationships from processed traffic data [4]. The trained models subsequently classify network connections as benign or malicious based on learned behavioral patterns [5].

The final stage of most reported intrusion detection architectures focuses on alert generation and response coordination. Once an intrusion is detected, alert mechanisms notify network administrators and update security logs for further investigation and auditing [3]. This response-oriented design ensures timely awareness of security incidents while supporting post-attack analysis. Overall, the layered IDS architecture depicted in Fig. X reflects a widely accepted design philosophy in prior research, offering adaptability, efficient processing, and seamless integration with organizational security infrastructures [1], [2].



3. Stages of the suggested system for detecting network intrusions

1. Network Traffic Collection

In this initial stage, raw network traffic is captured from live networks or benchmark datasets. The collected data consists of packet-level or flow-level information such as source and destination addresses, protocol types, packet sizes, and timing details. This phase guarantees that both normal and malicious traffic patterns are adequately represented for further analysis[1].

2. Data Pre-processing

The acquired network traffic redundant information. Pre-processing is therefore performed to clean the

data and improve its quality[3]. This includes handling missing attributes, removing irrelevant features, transforming numerical values from category values form, and normalizing feature ranges. Proper pre-processing enhances learning efficiency and reduces model bias.

3. Feature Selection and Extraction

From the pre-processed data The most instructive characteristics are chosen to represent network behavior effectively. This stage reduces dimensionality while retaining critical attack-related characteristics[5]. Statistical measures, correlation analysis, or model-based Techniques for feature importance are used to eliminate redundant and less significant attributes, resulting in faster training and improved detection accuracy.

4. Machine Learning-Based Model Training

In this stage, the refined dataset is used to train machine learning models capable of distinguishing between legitimate and malicious traffic. The learning algorithm captures complex patterns and relationships within the data. The training process involves learning decision boundaries based on historical traffic behavior, enabling the system to generalize well to unseen network activities[4].

1. Intrusion Detection and Classification

Once trained, the model is deployed to classify incoming network traffic in real time or offline mode. Each traffic instance is analyzed and labeled as normal or intrusive[2]. For multi-class scenarios, the system further categorizes attacks into specific types such as denial-of-service, probing, or unauthorized access attempts.

2. Performance Evaluation

Standard evaluation measures like accuracy and precision are used to gauge how successful the intrusion detection system is, recall, F1-score, and false alarm rate[5]. This stage validates the reliability and robustness of the proposed framework and helps identify areas for optimization.

3. Alert Generation and Response

When malicious activity is detected, the system generates alerts for network administrators. These alerts provide actionable insights that support timely response and mitigation[2]. This stage ensures that security threats are identified early, reducing potential damage to network infrastructure.

4. Continuous Learning and Model Update

To adapt to evolving cyber threats, the system supports periodic retraining with new traffic data. Continuous learning enables the model to recognize emerging attack patterns and maintain long-term detection effectiveness in dynamic network environments[3].

5. PROPOSED METHODOLOGY

Data Collection and Loading

By eliminating irregularities and preparing traffic records for learning algorithms, preprocessing aims to improve the quality of the data. Missing values, redundant entries, noisy characteristics, and mixed data types are common in network datasets and might impair model performance if ignored [4]. This stage eliminates redundant or unnecessary records, converts category attributes into numerical representations, and handles missing values using appropriate handling techniques. To lessen the impact of extreme results that could skew the learning process, outlier identification is used. To ensure consistency among numerical attributes, feature scaling and normalization are carried out. To enable objective performance evaluation, the dataset is further divided into training and testing subsets. Reliable intrusion detection results are

enhanced by effective preprocessing, which also increases learning stability [5].

Engineering Features

Feature engineering aims to identify and construct the most informative attributes that effectively characterize network behavior. Since not all features contribute equally to intrusion detection, this step focuses on selecting attributes that exhibit strong discriminatory power between benign and malicious traffic [6]. Features related to traffic volume, protocol behavior, and session dynamics are prioritized, while redundant or weakly correlated attributes are eliminated. Transformations and combinations of features may also be applied to expose hidden behavioral patterns. By reducing dimensionality and noise, feature engineering improves detection accuracy and reduces computational overhead. Well-engineered features enable the model to learn meaningful representations of network activity and enhance generalization capability.

sbytes

The sbytes attribute represents the volume of data transmitted from the source host during a network session. This feature is especially helpful in identifying abnormal outbound traffic behavior. Malicious activities such as data exfiltration, botnet communication, or denial-of-service attacks often produce unusual source byte patterns compared to legitimate traffic [2]. Excessively high values may indicate large unauthorized data transfers, while unusually low values can be associated with reconnaissance or scanning attempts. By analyzing sbytes, the detection model can capture deviations in sending behavior and distinguish suspicious activities from normal communication patterns.

State

The State feature describes the status of a network connection based on protocol-specific session behavior. Legitimate connections typically follow a well-defined sequence of states during establishment and termination. In contrast, attack traffic often violates these patterns, resulting in incomplete or abnormal session states [3]. For example, flooding attacks may generate repeated half-open connections, while probing activities may terminate sessions prematurely. Incorporating the State feature allows the model to detect irregularities in session flow and identify manipulation attempts commonly used by attackers.

Protocol

The Protocol attribute specifies the communication protocol used in a network session, such as TCP, UDP, or ICMP. Different attack types are strongly associated with specific protocols, making this feature valuable for accurate intrusion classification [1]. For instance, TCP-based attacks exploit connection management mechanisms, while UDP and ICMP are frequently misused in flooding and reconnaissance attacks. Observing protocol usage helps the model understand traffic intent and detect deviations from expected protocol behavior. Inclusion of this feature enhances the system's ability to associate observed traffic patterns with known attack strategies.

Dbytes

The dbytes feature indicates the amount of data received by the destination host during a connection. Abnormal inbound traffic volumes often signal malicious activity such as amplification attacks, reflection attacks, or unauthorized data delivery [4]. High destination byte counts may reveal traffic flooding, whereas unusually low values can indicate failed or suspicious connection attempts. When analyzed together with sbytes, this feature helps
identify asymmetric traffic patterns that are characteristic of many intrusion scenarios. Thus, dbytes plays a vital part in understanding inbound traffic behavior and improving detection reliability.

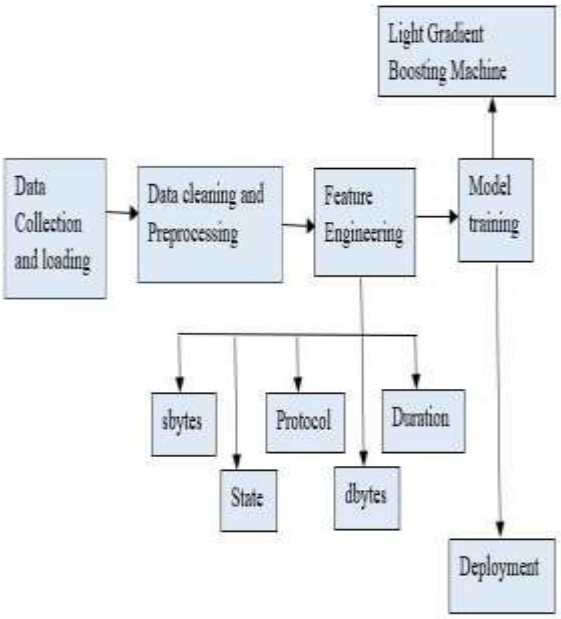
Duration

The Duration feature represents the total time span of a network session. Normal user activities typically exhibit predictable duration ranges, whereas attack traffic often deviates from these norms [5]. Very short sessions may indicate scanning or brute-force attempts, while prolonged connections may suggest persistent malware communication or unauthorized remote access. Temporal analysis using session duration helps detect time-based anomalies that are not easily captured through volume-based features alone. Including this attribute strengthens the model's ability to identify stealthy and long-lasting attacks.

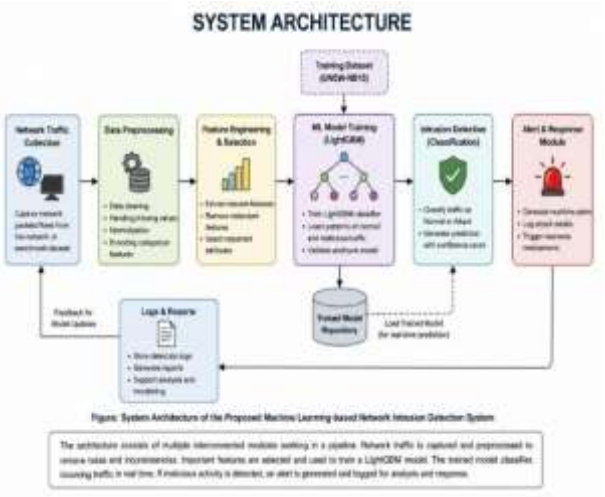
Model Training

Model training involves learning the distinction between normal and malicious traffic using the processed and feature-optimized dataset. The data is separated into training and validation sets during this stage.

to ensure fair assessment of performance. The learning algorithm extracts patterns by minimizing classification errors through iterative optimization [6]. Hyperparameters are tuned to balance detection precision and capacity for generalization. Performance indicators including F1-score, recall, accuracy, and precision are utilized to evaluate effectiveness. Proper training ensures that the intrusion detection model can accurately classify unseen traffic and remain robust under real-world network conditions.



SYSTEM ARCHITECTURE



The proposed system architecture illustrates the complete operational flow of the Machine Learning–Based Network Intrusion Detection System (ML-NIDS). The architecture is designed in a layered manner to ensure modularity, scalability, and efficient real-time detection of cyber threats[1]. Each architectural component performs a specific function, collectively enabling accurate intrusion detection, classification, and response[2].

The architecture consists of the following major components:

1. Network Traffic Source
2. Traffic Capture and Data Collection Module
3. Preprocessing and Feature Engineering Module
4. Machine Learning Detection Engine
5. Explainability and Decision Support Module
6. Alerting, Logging, and Response Module
7. User Interface and Monitoring Dashboard

6. Network Traffic Source

This layer represents the origin of network data. It includes traffic generated by internal users, external clients, servers, and network devices such as routers and switches. The traffic consists of both normal communication and potential malicious activities such as scanning, flooding, or unauthorized access attempts[3].

The architecture supports both real-time traffic capture and offline dataset-based analysis, making it suitable for experimental evaluation as well as live deployment[1].

7. Traffic Capture and Data Collection Module

The data collection module continuously captures incoming and outgoing network traffic. It gathers packet-level or flow-level information like the addresses of the source and destination, protocol types, session states, and traffic volumes[5]. Captured traffic is converted into structured records and temporarily stored for further processing. This module ensures minimal packet loss and does not interfere with normal network operations, enabling uninterrupted monitoring[1].

8. Preprocessing and Feature Engineering Module

The preprocessing layer prepares raw traffic data for machine learning analysis. It removes redundant records, handles missing or inconsistent values, and normalizes numerical attributes to a common scale[2]. Feature engineering is performed in this layer to extract meaningful behavioral attributes such as source bytes, destination bytes, session duration, protocol type, and connection state. These features form a compact and informative representation of network behavior, improving detection accuracy and reducing computational complexity.

9. Machine Learning Detection Engine

This is the core analytical component of the architecture. The processed feature vectors are fed into the trained Light Gradient Boosting Machine (LightGBM) model[3].

The detection engine performs:

- Traffic classification as normal or malicious
- Multi-class attack identification for different intrusion categories ,LightGBM efficiently analyzes high-

dimensional traffic data and generates prediction scores with low latency, making it suitable for near real-time intrusion detection.

10. Explainability and Decision Support Module

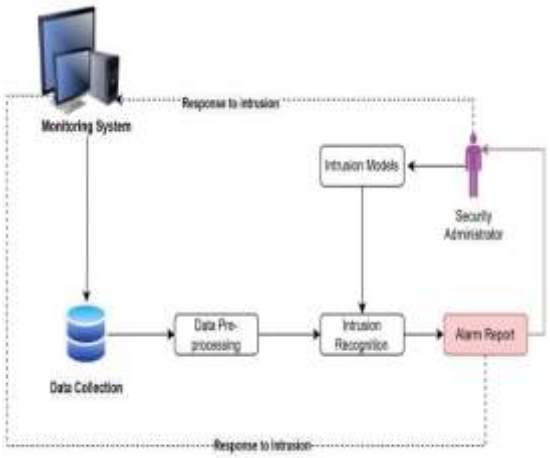
To enhance transparency and trust, the architecture integrates an explainability layer using SHAP (SHapley Additive exPlanations)[2]. This module evaluates the machine learning model's predictions by assigning importance values to individual features. Security analysts can understand which traffic characteristics contributed most to a detection decision. This improves confidence in automated alerts and supports forensic investigation.

1. Alerting, Logging, and Response Module

Once malicious activity is detected, this module generates alerts containing details such as attack type, time of occurrence, and affected resources. Alerts can be forwarded to administrators through dashboards or notification systems[4]. The module also maintains detailed logs of network activities, predictions, and responses. Based on predefined policies, response actions such as blocking IP addresses, terminating sessions, or updating firewall rules can be triggered automatically or manually.

2. User Interface and Monitoring Dashboard

The user interface provides a centralized platform for administrators and security analysts to monitor system status, view alerts, analyze logs, and evaluate detection performance[3]. The dashboard presents summarized statistics, real-time alerts, and historical reports in a user-friendly manner, allowing both technical and non-technical users to communicate with the system effectively.



Intrusion Detection System Architecture

SYSTEM IMPLEMENTATION

ALGORITHMIC STRATEGIES USED FOR DETECTION OF NETWORK TRAFFIC INTRUSION

Machine Learning algorithms form the analytical core of modern Intrusion Detection Systems (IDS)[1]. They are used to learn patterns from historical network traffic and classify incoming flow as either normal or attack. In this project, Light Gradient Boosting Machine (LightGBM) is adopted as the primary algorithm because of its high performance, quick computation, and capacity to handle complex network-level features[3]. Each algorithmic component contributes to improving threat detection accuracy, owering false positives and guaranteeing real-time intrusion monitoring. The following section explains the

LightGBM algorithm used in this system.

LIGHTGBM CLASSIFIER (GRADIENT BOOSTING–BASED MODEL)

LightGBM is a highly efficient Gradient Boosting-based machine learning algorithm Decision Trees (GBDT)**.** It trains multiple weak decision trees sequentially, where each new tree corrects fixes the mistakes of the prior ones. This makes LightGBM powerful for detecting complex intrusion patterns that are not linearly separable[1].

Unlike traditional models, LightGBM uses two key innovations:

1. **Gradient-based One-Side Sampling (GOSS)** GOSS selects the most important samples with higher gradient values and keeps a subset of low-gradient samples. This reduces training time while maintaining accuracy[5].
2. **Exclusive Feature Bundling (EFB)**
   EFB bundles mutually exclusive features (features that rarely take non-zero values together), reducing dimensionality and improving memory efficiency.
   These techniques make LightGBM extremely fast and suitable for analyzing large datasets such as UNSW-NB15[4]. LightGBM creates an ensemble of decision
   trees where each tree attempts to minimize the prediction errors from the previous trees. The final score is calculated by combining (boosting) the outputs of all trees[5].

Build Raw Tree Ensemble

$$F(x) = \sum_{m=1}^{M} f_m(x)$$

Where:

- **F(x)** = Combined score from all trees for input x
- **Σ** = Summation operator (adds values)
- **m = 1 to M** = Trees indexed from the first tree to the last
- **M** = Total number of trees used by LightGBM
- **fm(x)** = Output produced by the m-th decision tree

Scale Tree contributions using learning rate

$$F'(x) = \sum_{m=1}^{M} \eta \cdot f_m(x)$$

Where:

- **F′(x)** = Scaled prediction score for input x
- **η (eta)** = Learning rate
- **Σ (sigma)** = Summation symbol
- **m = 1 to M** = Index of trees
- **M** = Total number of decision trees
- **fm(x)** = Output of m-th tree for input x

Convert the Score to a Probability

$$F''(x) = \sum_{m=1}^{M} \eta \cdot f_m(x)$$

Where:

- **F′(x)**=The raw prediction score produced by the entire LightGBM model.
- **∑m=1M**= Sum of contributions from all M decision trees**.**
- **η**=The learning rate
- **fm(x)**=The output of the m-th decision tree for input x
- **M**=Total number of trees in the boosting process.

Hardware Requirements

The system requires the following minimum hardware configuration:

Processor

- Intel i5 / AMD Ryzen 5 or higher

Memory

- Minimum 8 GB RAM (16 GB recommended for large datasets)

Storage

- Minimum 256 GB HDD/SSD for datasets, logs, and models

Network Interface

Ethernet or wireless network interface capable of handling real-time traffic

- **Optional Hardware**

GPU (NVIDIA CUDA-enabled) for accelerated machine learning training (optional)

- **Software Requirements**

The system shall be developed and executed using the following software components:

- **Operating System**

Windows 10 / Ubuntu Linux

- **Programming Language**

Python 3.x

- **Development Environment**

Jupyter Notebook / PyCharm / VS Code

- **Machine Learning Libraries**

Scikit-learn

TensorFlow / PyTorch (if deep learning is used)

- **Data Processing Tools**

NumPy

andas

- **Visualization Tools**

Matplotlib

Seaborn

- **Database (Optional)**

MySQL / MongoDB for log and alert storage

IV RESULTS AND DISCUSSIONS

Output

The suggested machine learning-based network intrusion detection system's effectiveness was assessed.

using multiple result visualizations, including a confusion matrix, feature importance graph, and a prediction interface. These outputs provide insight into the accuracy, reliability, and interpretability of the model.

Confusion Matrix Analysis

The confusion matrix presented in Fig. 01 illustrates the classification efficacy of the suggested model in distinguishing between normal and malicious network traffic. The model effectively classified 17,556 instances of normal traffic and 29,994 instances of attack traffic, indicating a high level of correct predictions.

However, a limited number of misclassifications were observed. Approximately 1,044 normal connections

were incorrectly identified as attacks, resulting in false alarms. Additionally, 2,941 attack instances were misclassified as normal traffic, representing missed intrusions. Although the overall classification performance is strong, reducing the number of missed attacks remains a critical requirement, as undetected intrusions pose a greater security risk. These findings imply that the model performs reliably while still offering scope for improvement in attack recall.

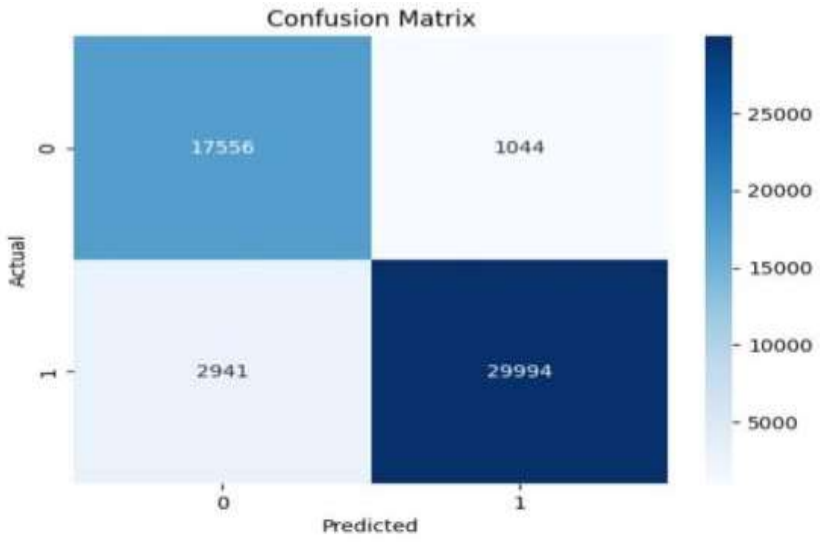
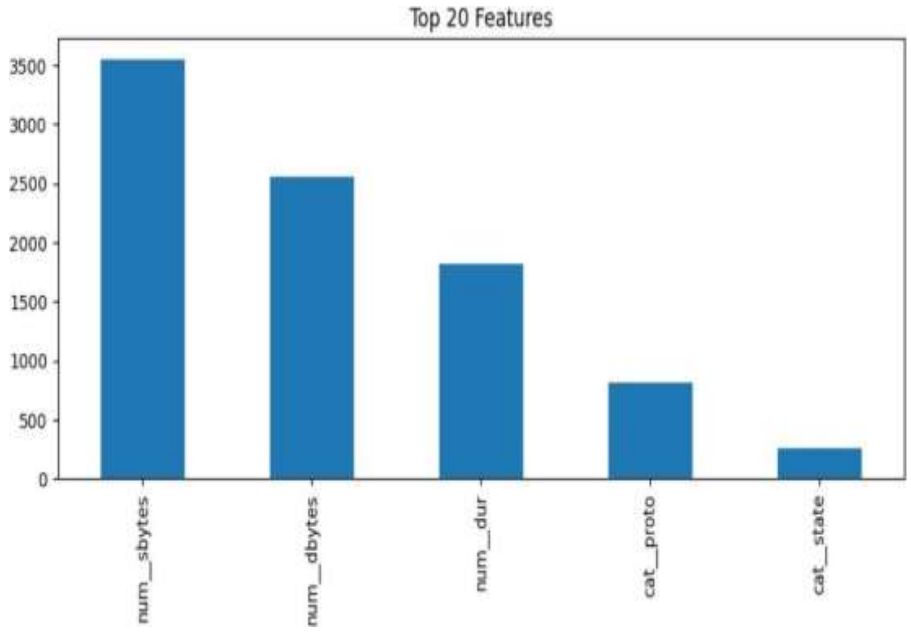


Fig 01

Feature Importance Analysis

The feature num_sbytes (source bytes) emerges as the most dominant factor, indicating that the volume of data transmitted by the source plays a major role in identifying abnormal behavior. This is followed by num_dbytes (destination bytes) and num_dur (connection duration), both of which significantly influence classification decisions. These features reflect abnormal traffic volume and session timing, which are common indicators of cyber-attacks.

Other attributes such as protocol type and connection state show comparatively lower importance but still contribute to detection accuracy. This analysis enhances model interpretability by clearly identifying which traffic characteristics are most relevant, thereby increasing trust in the system's decision-making process.



Prediction Interface Analysis

The interface allows users to input key network parameters such as connection duration, protocol type, connection state, source bytes, and destination bytes. Upon submitting, the trained model analyzes the data and produces a prediction. In the illustrated example, the model classifies the network traffic as Normal, with an attack probability of 0.11, indicating only an 11% likelihood of malicious behavior. This probabilistic output provides additional clarity to users by not only indicating the predicted class but also showing the confidence level of the prediction. The interface simplifies interaction with the system and enables non-expert users to quickly assess network security status.

The experimental results demonstrate that the proposed intrusion detection system achieves reliable

classification performance with high accuracy and interpretability. The confusion matrix confirms effective detection of both normal and attack traffic, while the feature importance analysis highlights the key attributes influencing detection decisions. Furthermore, the prediction interface enhances system usability by providing real-time, user-friendly intrusion assessment[2]. Overall, the results validate the effectiveness of the proposed approach in detecting network intrusions while maintaining transparency and operational practicality. With further optimization to reduce missed attacks, the system can be strengthened for real-world deployment in dynamic network environments[1].

CONCLUSION AND FUTURE WORK

This paper presented a machine learning–based Network Intrusion Detection System leveraging the Light Gradient Boosting Machine (LightGBM) algorithm to accurately distinguish between normal and malicious network traffic. The proposed model achieved high detection accuracy with well-balanced precision, recall, and F1-score, indicating effective reduction of false alarms while reliably identifying genuine attack instances [1]. Feature
importance analysis revealed that traffic-related attributes such as source bytes, destination bytes, session duration, and protocol type play a vital part in intrusion detection performance, consistent with findings reported in prior studies [2]. The outcomes of the experiment show that LightGBM offers a scalable, efficient, and robust alternative to conventional rule-based security mechanisms Future studies will concentrate on these inputs in real time, system deployment, incorporating deep learning techniques for detecting sophisticated attack patterns, extending multi-class classification for fine-grained attack identification, and enhancing model explainability and computational efficiency for deployment in cloud, IoT, and edge computing environments

REFERENCES

1. Faizal, M.A, Mohd Zaki M, Shabrin Sahib, Robaih Y, “Time based Intrusion Detection on Fast Attack for Network Traffic Intrusion Detection” IEEE , Vol. 2, no. 1, 2010, pp. 148-152.
2. Govind P Gupta, Manish Kulariya , “ A Framework for Fast and Efficient Cyber Security Network Intrusion Detection using Apache Spark ”Elsevier, vol .93, pp. 824-831, 2016.
3. Dr.Nikhil Raj , Dr Kavitha, Dr. Ganapathi sridhar , “Network Traffic monitoring using Intrusion Detection System”,§vol .3, pp. 147–153,2017.
4. Alex Shenfield, Alex Shenfield, “Intelligent intrusion detection systems using artificial neural networks,” Elsevier, vol .4, pp. 95-99., April 2018.
5. Lirim Ashiku, Cihan Dagli, “Network intrusion detection system using deep learning,” Elsevier, vol. 9, pp. 240-247, Jan. 2021.