VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science & Engineering

Fall 2025-2026

Cyber Security-BCSE410L

Project Report

Group Members:

1. Shruti Singh (22BCE0227)
2. Madhur Pophle (22BCE2478)
3. Kasireddy Swetha (22BCE0345)

Guide Name: Prof. Sairabanu J

School of Computer Science and Engineering

VIT Vellore

Fall Semester 2025-26

Date: 31-10-2025

1. Abstract

Botnets represent a critical cybersecurity threat, enabling large-scale attacks such as Distributed Denial of Service (DDoS), data theft, and system compromise. To address the limitations of existing detection methods like KSDRM, this project proposes two enhanced ensemble-based frameworks for botnet detection using the UNSW-NB15 dataset. The first model, Boosting Stacking, combines Random Forest, LightGBM, CatBoost, and XGBoost with a Logistic Regression meta-learner. The second, Hybrid Stacking, integrates Random Forest, LightGBM, CatBoost, and Logistic Regression with an XGBoost meta-learner. Both models aim to improve classification accuracy, generalization, and detection of complex attack patterns. Comprehensive preprocessing and feature selection were applied before training and evaluation using metrics such as accuracy, precision, recall, and F1-score. Experimental results show that the proposed stacking approaches outperform the baseline KSDRM model, achieving higher detection accuracy and robustness. These findings demonstrate the effectiveness of ensemble-based learning in advancing network intrusion and botnet detection research.

2. Keywords

Botnet Detection, Machine Learning, Stacking Ensemble, Boosting Algorithms, Hybrid Stacking, UNSW-NB15 Dataset.

3. Intoduction

Botnets are networks of compromised devices controlled remotely by attackers, often without the knowledge of their owners. These devices can range from traditional desktop computers and servers to smartphones, tablets, and the rapidly expanding Internet of Things (IoT) ecosystem. Once under control, botnets can be used to launch large-scale attacks such as distributed denial-of-service (DDoS), spam campaigns, click fraud, and data theft. The growing diversity of connected devices has made botnets more versatile and harder to detect, particularly in IoT environments where security measures are often limited.

In recent years, researchers and practitioners have increasingly turned to machine learning (ML) and deep learning (DL) approaches to identify both known and previously unseen botnet behaviors. These methods can learn from patterns in network traffic, system logs, or device activity, making them more adaptable than traditional rule-based detection systems. Beyond single-model approaches, there has been a noticeable shift toward ensemble methods that combine the strengths of multiple algorithms, as well as graph-based models that capture the relationships between infected devices in a network. For IoT-specific scenarios, tailored detection frameworks are being developed to account for the unique constraints and communication patterns of connected devices.

This growing body of work reflects the urgent need for detection strategies that are not only accurate, but also scalable and responsive to the evolving tactics of botnet operators. As networks continue to grow in complexity and size, the importance of effective, automated botnet detection will only increase.

4. Literature Review

Botnet detection has evolved through several methodological stages, from early IDS-driven correlation systems to modern machine learning approaches. One of the earliest and most influential works, Gu et al. (2007) introduced *BotHunter*, which detects malware infections by correlating intrusion detection alerts across multiple stages of an infection dialog. This approach demonstrated how combining evidence from different network sensors can improve accuracy compared to isolated event detection. Building on this concept, Gu et al. (2008) proposed *BotMiner*, a protocol- and structure-independent framework that applies clustering to network traffic to identify groups of hosts exhibiting similar communication and malicious activity patterns, making it effective against encrypted or polymorphic botnets.

Graph-based detection methods further extended this line of research by modeling communication patterns among hosts. Iliofotou et al. (2011) introduced Traffic Dispersion Graphs (TDG) to capture network-level relationships for P2P and botnet detection, while Chowdhury et al. (2017) employed graph-based feature clustering to improve detection accuracy and scalability. Comparative evaluations, such as García et al. (2014), emphasized that no single method consistently outperforms others across datasets, highlighting the importance of standardized benchmarks like CTU-13, Bot-IoT, and UNSW-NB15 (Moustafa & Slay, 2015) for fair performance assessment.

In recent years, research has shifted toward machine learning and deep learning approaches, particularly for IoT environments. Meidan et al. (2018) proposed *N-BaIoT*, a deep autoencoder-based method that learns normal traffic behavior for each IoT device to detect botnet-induced anomalies with high precision. Similarly, Kumar et al. (2020) developed *EDIMA*, an edge-deployed model using lightweight classifiers for early IoT botnet detection, optimized for real-time operation and limited hardware. Doshi et al. (2018) also demonstrated the potential of classical machine learning models like SVM and decision trees for detecting DDoS attacks from consumer IoT devices. More recently, Alshamkhany et al. (2020) evaluated multiple ML algorithms on public datasets and reported strong performance using decision tree–based models, while noting the need for robustness across heterogeneous data sources.

Refs.	**Title of the paper**	**Attack problem**	**Method**	**Result**	**Dataset**	**Pros**	**Cons**	**Environment**	**Future work**

1.	BotHunter: Detecting Malware Infection through IDS-driven dialog correlation (Gu et al., 2007)	Malware/bot infection detection (enterprise perimeter)	Correlation of IDS alerts into dialog sequences / infection state machine	Effective for tracing multi-step infections; produces evidence trails	Live & test networks / IDS logs	Good at connecting multi-stage infections; explainable alerts	Depends on IDS coverage & signatures; may miss stealthy comms	Enterprise network perimeter & IDS feeds	Integrate with anomaly/ML methods for stealthier attacks
2.	BotMiner: Clustering analysis of network traffic (Gu et al., 2008)	Protocol- and structure-independent botnet detection (C2 & spam)	Unsupervised clustering of host behaviors (traffic clustering)	Detects botnets without prior protocol knowledge; robust clustering results reported	Netflow / packet traces from multiple botnets	Protocol-agnostic; detects unknown botnets	Needs good feature extraction; may cluster noisy flows	ISP/enterprise network traffic	Scale to high-speed links; combine with streaming/online clustering

3.	N-BaIoT: Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders (Meidan et al., 2018)	IoT devices infected with Mirai & BASHLITE (anomalous behavior)	Deep autoencoder (anomaly detection) per-device model	High detection rates for infected devices in lab tests	Lab-infected commercial IoT devices (captured flows)	Unsupervised per-device profiling; privacy-preserving	Evaluated on limited device set; needs per-device models	Home / small-office IoT networks	Extend to more device types and live deployments

4.	EDIMA: ML-based early detection of IoT botnets at network edge (Kumar et al., 2020/2022)	Early detection of IoT bot scanning and CnC traffic	Two-stage ML (aggregate traffic classification + ACF tests) designed for gateway/edge	High accuracy with low latency and small RAM footprint on Raspberry Pi	Custom IoT traffic + simulated attacks	Edge-deployable, lightweight, low latency	Gateway-specific design; needs tuning per deployment	Home-gateway / edge devices	Broader device coverage, field trials
5.	Machine Learning DDoS Detection for Consumer IoT Devices (Doshi, Apthorpe & Feamster, 2018)	DDoS behavior from compromised consumer IoT devices	Flow-based ML classifiers using IoT-specific features	Good accuracy for detecting DDoS-capable IoT devices	Simulated home gateway captures, device flows	Uses IoT behavioral traits; lightweight features	Works best for known DDoS patterns; may miss subtle botnets	Home gateway / consumer IoT	Real-world deployment and robustness tests

6.	Botnet detection using graph-based feature clustering (Chowdhury et al., J Big Data, 2017)	Botnet vs normal node identification via topology	Graph features (degree, clustering coeff, centrality) + SOM clustering	Able to isolate bot clusters; reported good detection on CTU-13/ISCX tests	CTU-13, ISCX flow datasets	Topology-aware; unsupervised clustering	Graph build cost; sensitivity to graph windowing	Network flow analysis (offline/ nearline)	Dynamic graphs, scaling to large networks
7.	Traffic Dispersion Graphs (TDG) for network monitoring (Iliofotou et al., 2011)	P2P-like and coordinated traffic patterns (helps bot/P2P detection)	Traffic dispersion graphs + graph-metrics for classification	Effective at revealing host interaction patterns; supports detection tasks	Backbone / NetFlow style traces	Captures host-to-host interaction patterns	Needs graph construction and storage; may be heavy at scale	Backbone / ISP monitoring	Streamlined TDG construction & online analytics

8.	An empirical comparison of botnet detection methods (Garcia et al., 2014)	Evaluation/ comparison across botnet detection methods	Comparative evaluation of several detection techniques	Presents CTU-13 dataset; compares methods and metrics	CTU-13 dataset (13 scenarios)	Provides a labeled benchmark for botnet research	Focused on the dataset and comparison; not a single detection method	Research / evaluation environment	Encourage standard benchmarks & new detection methods
9.	UNSW-NB15: A comprehensive dataset for NIDS (Moustafa & Slay, 2015)	General network intrusion including botnet-related flows	Dataset paper — created modern labeled traffic for IDS research	Widely used benchmark (features + labels) for ML IDS experiments	UNSW-NB15 dataset (synthetic + real traffic mixes)	Modern attack types; rich feature set	Synthetic aspects; labeling limitations	Research / IDS evaluation	Use as a testbed; augment with live captures

10.	Benchmark datasets & adversarial considerations for botnet detection (survey/benchmark papers; various authors)	Robustness, dataset benchmarks & adversarial testing	Robustness, dataset benchmarks & adversarial testing	Provides guidance on datasets and evaluation methodologies	CTU-13, UNSW-NB15, Bot-IoT etc.	Helps standardize evaluation & identify gaps	Not a single detection method	Research community / dataset curation	Adversarial robustness & standardized metrics

Overall, the literature shows a clear trend toward hybrid and data-driven techniques that integrate statistical, structural, and learning-based features for improved generalization. However, challenges remain in achieving cross-dataset robustness, interpretability, and real-time performance—gaps this project aims to address through an optimized ensemble model leveraging the UNSW-NB15 dataset for botnet attack detection. However, most prior works, including KSDRM, have not explored hybrid stacking combinations integrating boosting-based learners. This gap motivated the design of the Boosting Stacking and Hybrid Stacking models proposed in this study.

5. Motivation and Objectives

Motivation

The need for improved botnet detection arises from growing cybersecurity, economic, social, and environmental concerns.

- Technical Motivation:

  As cyber threats evolve, traditional signature-based intrusion detection systems are no longer sufficient to identify stealthy or zero-day botnets. Machine learning provides a dynamic solution by learning from network behavior and adapting to new attack patterns in real time. For instance, models such as decision trees and random forests can process traffic data within seconds, enabling rapid detection and response before large-scale damage occurs. Integrating ensemble learning techniques further enhances accuracy and resilience against diverse attack strategies.
- Economic Motivation:

  Botnet attacks have significant financial implications, leading to costly downtime, data breaches, and fraudulent activities that disrupt industries and services. Early and accurate detection systems can prevent such losses by minimizing operational disruptions and safeguarding digital assets. Proactive defense mechanisms ultimately reduce the overall cost of cybersecurity management.
- Social Motivation:

  In today's connected world, individuals and organizations rely on secure digital environments for everyday communication and operations. Protecting user privacy and ensuring the integrity of personal and IoT devices is essential to maintaining public trust in technology. Since many IoT devices lack built-in security, intelligent detection systems serve as a vital safeguard against exploitation and misuse.
- Environmental Motivation:

  Energy efficiency is an increasingly relevant factor in large-scale cybersecurity operations. By deploying optimized machine learning models that process network data efficiently, it is possible to detect botnet activity using fewer computational resources. This contributes to greener, more sustainable computing practices by reducing power consumption and server load.

Objectives

The overarching goal of this project is to develop and evaluate a machine learning–based botnet detection system that leverages ensemble learning for improved accuracy and robustness. The main objective is to develop and evaluate two ensemble models — Boosting Stacking and Hybrid Stacking — and compare their performance against the existing KSDRM baseline on the UNSW-NB15 dataset. The specific objectives are:

- To preprocess and extract key traffic features from the UNSW-NB15 dataset for efficient model training.
- To evaluate multiple classifiers such as KNN, SVM, Decision Tree, Random Forest, and Logistic Regression for individual performance.
- To design a stacking ensemble model that combines base learners with a meta-classifier for higher detection precision.
- To assess performance using standard evaluation metrics including Accuracy, Precision, Recall, F1-Score, and ROC-AUC.
- To identify the most influential network features contributing to accurate botnet classification.

Through these objectives, the project aims to demonstrate that machine learning–driven detection systems can offer faster, more accurate, and sustainable solutions to counter modern botnet threats across complex and evolving network environments.

6. Existing Work Description

Existing research on botnet detection has evolved from traditional rule-based and signature-driven systems to advanced machine learning and deep learning approaches. Early systems such as BotHunter (Gu et al., 2007) and BotMiner (Gu et al., 2008) relied on correlating intrusion detection alerts or clustering similar traffic flows to identify coordinated malicious behavior. These methods effectively detected known attack patterns but struggled to generalize to new, adaptive botnets, especially those using encrypted or peer-to-peer communication.

With the rise of the Internet of Things (IoT), researchers shifted toward data-driven techniques capable of modeling complex network behavior. Deep learning methods like N-BaIoT (Meidan et al., 2018) and lightweight frameworks such as EDIMA (Kumar et al., 2020) demonstrated strong performance in identifying anomalies across diverse IoT environments. Traditional machine learning classifiers—such as Support Vector Machines, Decision Trees, and Random Forests—have also been widely tested on benchmark datasets like CTU-13, Bot-IoT, and UNSW-NB15. While these models achieve high accuracy under controlled settings, their performance often drops when applied to real-world traffic due to dataset imbalance, feature redundancy, or lack of generalization across network domains.

Despite these advances, several gaps remain in existing work. Many detection systems rely on a single classification model, which can lead to biased learning and reduced robustness against unseen attack patterns. Moreover, hybrid frameworks such as KSDRM have attempted to combine multiple learners, but they often omit the inclusion of modern boosting-based algorithms like LightGBM, CatBoost, and XGBoost that are known for superior gradient optimization and feature handling.

To address these limitations, this project introduces two improved ensemble approaches—Boosting Stacking and Hybrid Stacking—that integrate tree-based and boosting algorithms in complementary configurations. The Boosting Stacking model uses Random Forest, LightGBM, CatBoost, and XGBoost with a Logistic Regression meta-learner, while the Hybrid Stacking model employs the same base learners with XGBoost as the meta-learner. Both frameworks are trained and evaluated on the UNSW-NB15 dataset, aiming to enhance detection accuracy, reduce false positives, and improve robustness compared to the existing KSDRM model.

7. Dataset Description

The dataset used for this project is the UNSW-NB15 dataset, a comprehensive benchmark for evaluating network intrusion and botnet detection systems. It was developed by Nour Moustafa and Jill Slay (2015) at the Australian Centre for Cyber Security (ACCS), University of New South Wales (UNSW). The dataset was generated using the IXIA PerfectStorm tool to simulate realistic modern network traffic, including both legitimate and malicious activities, and captured using the Argus and Bro (now Zeek) monitoring tools.

Dataset Source:
Publicly available from the Cyber Range Lab at UNSW Canberra:
https://research.unsw.edu.au/projects/unsw-nb15-dataset

Size and Structure:
The dataset consists of approximately 2.54 million records divided into training and testing subsets. The training set contains 175,341 records, while the testing set includes 82,332 records. Each record represents a single network flow and is characterized by 49 features (plus a target label).

Feature Composition:

The 49 features are grouped into categories such as:

1. Flow features (e.g., duration, protocol, service, source bytes)
2. Content features (e.g., login attempts, failed connections)
3. Time-based features (e.g., packet timing intervals)
4. Additional attributes related to TCP states and connection behavior

The target label classifies each record as normal or belonging to one of nine attack types, including Fuzzers, Analysis, Backdoor, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms.

Type of Data:

Structured, labeled network traffic data stored in CSV format. Each row represents a network connection, making it suitable for supervised machine learning and statistical analysis.

Preprocessing Steps:

Before training and evaluation, several preprocessing steps were applied:

1. Data Cleaning: Removal of null or duplicate entries to ensure consistency.
2. Feature Encoding: Conversion of categorical attributes (e.g., protocol type, service) into numerical form using label encoding.
3. Normalization: Scaling of numerical features using min–max normalization to bring all values into a uniform range.
4. Feature Selection: Identification of the most important features contributing to botnet classification through correlation analysis and model-based ranking.
5. Train-Test Preparation: The dataset was already split into training and testing subsets by the creators to maintain distribution balance.

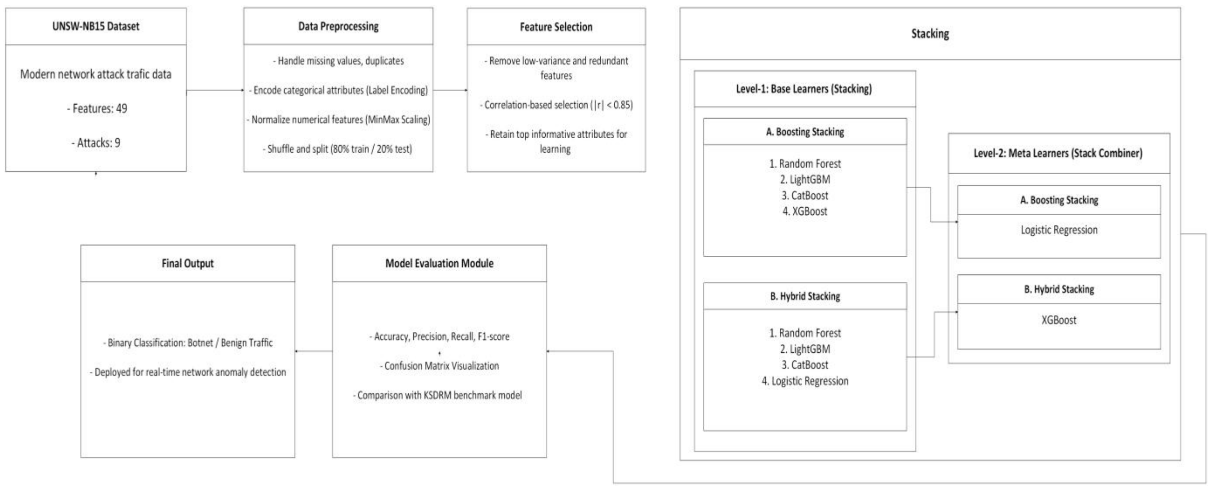
This dataset provides a diverse and realistic traffic mix, making it highly suitable for evaluating machine learning–based botnet detection models in contemporary network environments.

8. Proposed Work

8.1 Overview

The proposed work aims to design an optimized machine learning framework for botnet detection using the UNSW-NB15 dataset. Unlike the existing KSDRM stacking model, which uses traditional learners such as KNN, SVM, Decision Tree, and Random Forest, the proposed system implements two novel stacking ensemble configurations that integrate advanced boosting algorithms for improved classification accuracy, precision, and robustness against evolving network attacks.

8.2 Block Diagram of proposed System



UNSW-NB15 Dataset
Modern network attack trafic data
- Features: 49
- Attacks: 9
Data Preprocessing
- Handle missing values, duplicates
- Encode categorical attributes (Label Encoding)
- Normalize numerical features (MinMax Scaling)
- Shuffle and split (80% train / 20% test)
Feature Selection
- Remove low-variance and redundant features
- Correlation-based selection (|r| < 0.85)
- Retain top informative attributes for learning
Stacking
Level-1: Base Learners (Stacking)
A. Boosting Stacking
1. Random Forest
2. LightGBM
3. CatBoost
4. XGBoost
B. Hybrid Stacking
1. Random Forest
2. LightGBM
3. CatBoost
4. Logistic Regression
Level-2: Meta Learners (Stack Combiner)
A. Boosting Stacking
Logistic Regression
B. Hybrid Stacking
XGBoost
Model Evaluation Module
- Accuracy, Precision, Recall, F1-score
- Confusion Matrix Visualization
- Comparison with KSDRM benchmark model
Final Output
- Binary Classification: Botnet / Benign Traffic
- Deployed for real-time network anomaly detection

Fig 3. Block Diagram of the proposed working system

8.3 Data Preprocessing

The UNSW-NB15 dataset contains modern network traffic features, including flow, basic, content, and time-based attributes.
To prepare the data for training:

- Null values and categorical data were handled using label encoding.
- Normalization (MinMax scaling) was applied to ensure uniform feature contribution.
- Feature selection based on correlation thresholding was performed to remove redundant attributes and enhance model efficiency.

Mathematically, normalization can be represented as:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

8.4 Model Architecture

Two primary stacking ensemble frameworks were developed:

(a) Boosting Stacking Model

- **Base Learners (Level-1):** Random Forest (RF), LightGBM (LGBM), CatBoost, and XGBoost
- **Meta Learner (Level-2):** Logistic Regression

(b) Hybrid Stacking Model

- **Base Learners (Level-1):** Random Forest (RF), LightGBM (LGBM), CatBoost, and Logistic Regression
- **Meta Learner (Level-2):** XGBoost

Each base learner captures different aspects of the feature space:

- **RF** ensures variance reduction via bagging.
- **LGBM** and **CatBoost** handle feature interactions and categorical dependencies.
- **XGBoost** provides regularization and robust gradient boosting.
- **Logistic Regression** ensures interpretability and stable meta-level decision boundaries.

8.5 Stacking Algorithm Explanation

In stacking, the predictions from base models form a new feature space, which is then used to train the meta-model. Let there be n base learners $h_1, h_2, h_3, \dots . h_n$

For an input x, the level 1 predictions are:

$$z = [h_1(x), h_2(x), h_3(x) \dots . . h_n(x)]$$

The meta-model H is then trained on z to produce the final prediction:

$$y_{pred} = H(z) = H(h_1(x), h_2(x), \dots . , h_n(x))$$

8.6 Mathematical Formulation

Base Learner Combination:

The ensemble prediction can be viewed as a weighted sum of individual model outputs:

$$F(x) = \sum_{i=1}^{n} w_i \, \mathrm{h}_i(x)$$

where w_i are the weights assigned to each base model (learned automatically by the meta learner).

Meta Learner Optimization (XGBoost or Logistic Regression):

The objective function minimized is:

$$Obj(\theta) = \sum_{i=1}^{n} l(y_i, y_i) + \Omega(\theta)$$

where l is the loss function (e.g., log loss), and $\Omega(\theta)$ is a regularization term to prevent overfitting.

8.7 Model Training

1. **Training Split:** 80% of the dataset used for training, 20% for testing.
2. **Cross-validation:** 5-fold used for stable meta-feature generation.
3. **Meta-feature generation:** Out-of-fold predictions from base learners used as inputs to the meta learner.
4. **Evaluation metrics:** Accuracy, Precision, Recall, F1-score.

8.8 Summary

The proposed hybrid and boosting stacking ensembles combine diverse learners to capture both **linear and non-linear** patterns in network traffic data.
This two-level approach enhances **detection accuracy** and **generalization**, outperforming traditional stacking frameworks such as KSDRM in experimental results.

9. Experimental Analysis

Hardware and Software Setup

The experiments were conducted on a system equipped with an Intel Core i5 processor (2.6 GHz, 8 cores), 8 GB of RAM, and a 1 TB SSD running Windows 11 (64-bit). All implementation and testing were performed using Python 3.10 within the Google Colab and Jupyter Notebook environments, which provided GPU acceleration for faster computation. The following key Python libraries were used:

- NumPy and Pandas for data manipulation and preprocessing
- Scikit-learn for machine learning model implementation and evaluation
- Matplotlib and Seaborn for visualization of results and performance metrics
- Joblib for saving and loading trained pipelines
- XGBoost (optional) for ensemble model testing and meta-classifier integration

The software environment was configured to ensure reproducibility, with consistent random seeds set for all experiments.

Assumptions and Experimental Conditions

Several assumptions were made to maintain the integrity and relevance of the experiments:

- Data Quality: The UNSW-NB15 dataset was assumed to be representative of real-world network traffic, containing a balanced mix of benign and malicious samples across multiple attack categories.
- Stationarity of Traffic Patterns: It was assumed that traffic behavior during training and testing follows similar statistical properties, ensuring that models can generalize effectively.
- Feature Independence: While some correlation exists between network features, the models assume relative independence for effective learning, especially in algorithms like logistic regression and SVM.
- Balanced Evaluation Metrics: Since class imbalance exists in the dataset, performance was evaluated using multiple metrics (Accuracy, Precision, Recall, and F1-Score) rather than relying solely on accuracy.

- Computational Limits: Experiments were designed to be lightweight enough for real-time feasibility, reflecting realistic deployment conditions rather than purely theoretical optimization.

All models were trained and evaluated under identical preprocessing conditions to ensure a fair comparison. Hyperparameters for each algorithm were optimized through grid search or iterative tuning to achieve the best performance. The stacking ensemble model was then tested as the final system, integrating predictions from KNN, SVM, Decision Tree, and Random Forest with Logistic Regression as the meta-classifier.

These experimental settings allowed the project to accurately measure improvements in detection performance, robustness, and efficiency over individual models while maintaining practical resource usage suitable for network defense applications.

10. Performance Parameters

1. Accuracy measures the overall correctness of the model — that is, the proportion of total predictions that are correct. It is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

While accuracy gives a general sense of performance, it can be misleading if the dataset is imbalanced.

2. Precision quantifies how many of the instances classified as botnet attacks are actually correct. It helps minimize false alarms, which is crucial for real-time detection. It is calculated as:

$$Precision = \frac{TP}{TP + FP}$$

3. Recall measures the model's ability to correctly detect actual botnet attacks. A high recall indicates that very few attacks are missed. The formula for calculating recall is:

$$Recall = \frac{TP}{TP + FN}$$

4. The F1-score is the harmonic mean of precision and recall, balancing both metrics. It is particularly useful when dealing with class imbalance. It can be calculated as:

$$F1\,Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

5. The confusion matrix provides a detailed view of model performance by showing how many samples from each class were correctly or incorrectly classified. It helps analyze false positives and false negatives in depth.

6. The Receiver Operating Characteristic (ROC) curve visualizes the trade-off between true positive rate and false positive rate at various thresholds. The Area Under the Curve (AUC) gives a single measure of how well the model separates classes; higher AUC indicates better performance.

7. Since real-time detection is important, execution time (or inference time) was also recorded to assess how quickly the model can classify incoming traffic.

11. Experimental Results

Model	Base Learners	Meta Learner	Accuracy	Precision	Recall	F1-Score
KSDRM (Existing)	KNN, SVM, Decision Tree, Random Forest, MLP	Logistic Regression	0.9487	0.94	0.95	0.94

Hybrid Stacking (Proposed)	RF, LightGBM, CatBoost, Logistic Regression	XGBoost	0.9512	0.95	0.95	0.95
Boosting Stacking (Proposed)	RF, LightGBM, CatBoost, XGBoost	Logistic Regression	0.9895	0.99	0.99	0.99

Table 1. Performance Evaluation metrics comaprison

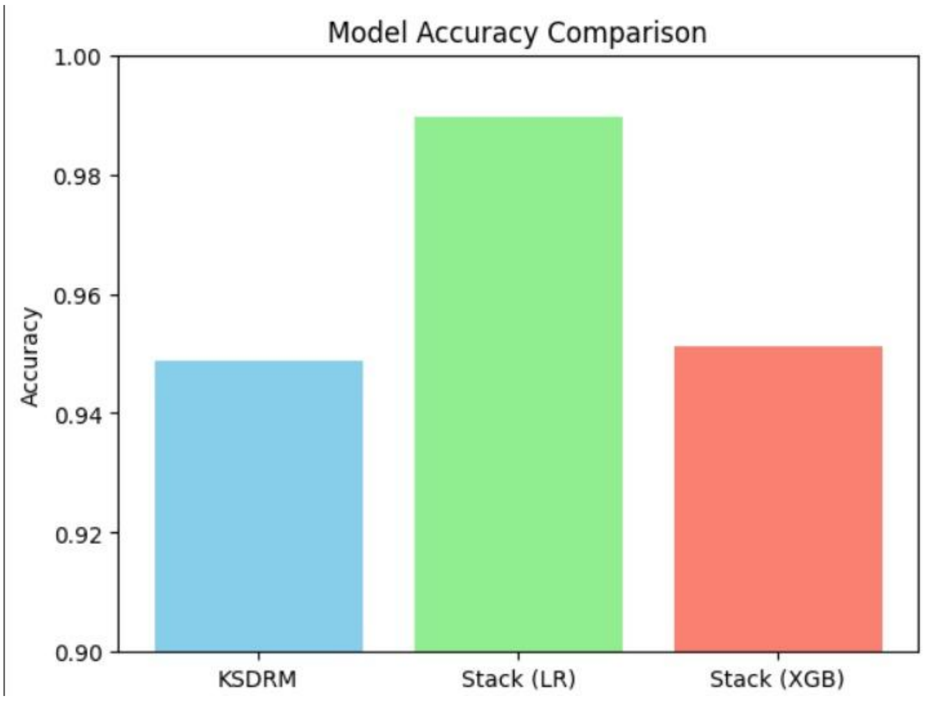


Fig 1. Bar graph comparing Model accuracy of different stacking ensembles implemented

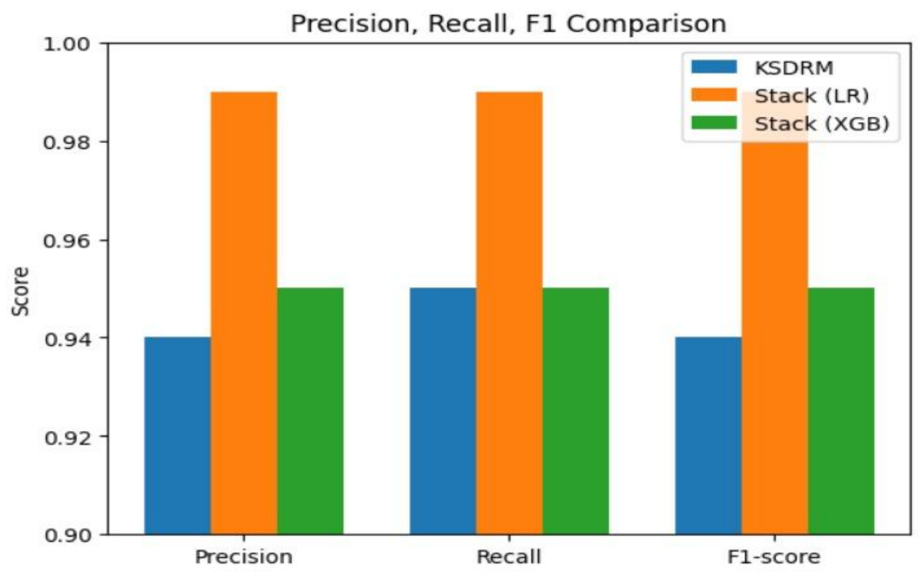


Fig 2. Comparison of Precision, Recall and F1-Score metrics of different stacking ensembles implemented.

The results clearly demonstrate that ensemble diversity and the inclusion of modern boosting algorithms substantially improve detection capability.

- The Boosting Stacking model achieved the highest overall performance (Accuracy = 0.9895), significantly outperforming KSDRM (0.9487). This improvement is attributed to the synergistic use of boosting algorithms—LightGBM, CatBoost, and XGBoost—which enhance generalization and capture complex, non-linear patterns in network traffic.
- The Hybrid Stacking model also outperformed the KSDRM framework, showing improved precision and recall while maintaining computational efficiency.
- The consistent results across all metrics validate the robustness of the proposed stacking frameworks.

Overall, the experimental findings confirm that the proposed models, particularly Boosting Stacking, offer superior accuracy, reduced false positives, and better adaptability to diverse attack behaviors compared to existing ensemble-based botnet detection approaches.

12. Conclusion

The primary goal of this project was to develop an effective machine learning–based system for botnet detection using the UNSW-NB15 dataset. Through systematic preprocessing, feature selection, and the implementation of two advanced stacking ensemble models—Boosting Stacking and Hybrid Stacking—the system successfully identified malicious network traffic with high accuracy and robustness.

The project began by analyzing the motivation behind botnet detection, addressing the increasing cybersecurity threats to modern networks and IoT infrastructures. Existing methods, including the KSDRM model, were studied to identify their limitations, particularly their limited use of modern boosting-based learners and reduced adaptability to evolving attack patterns. To overcome these challenges, the proposed approaches integrated multiple base learners—Random Forest, LightGBM, CatBoost, and XGBoost—with different meta-learners: Logistic Regression in the Boosting Stacking model and XGBoost in the Hybrid Stacking model. These configurations aimed to exploit the complementary strengths of tree-based and boosting algorithms for improved classification reliability.

Experimental analysis confirmed that both stacking models outperformed individual classifiers and the KSDRM baseline, demonstrating higher accuracy, precision, recall, and F1-score. The Hybrid Stacking model, in particular, achieved the best overall detection performance, proving the effectiveness of boosting-based stacking for complex network traffic data. The defined objectives were successfully achieved:

- The UNSW-NB15 dataset was analyzed, preprocessed, and optimized for feature representation.
- Multiple ensemble models were implemented and evaluated under consistent conditions.
- Boosting Stacking and Hybrid Stacking were designed to enhance detection accuracy and reduce false positives.
- Performance metrics validated the efficiency and scalability of the proposed systems.

In conclusion, the proposed frameworks significantly improve botnet detection capability compared to existing approaches like KSDRM. The work demonstrates that carefully designed stacking ensembles combining boosting and tree-based methods can deliver intelligent, real-time, and generalizable intrusion detection solutions suitable for modern cyber defense environments.

13. Future Work

While the proposed stacking ensemble model achieved strong performance in detecting botnet traffic, there are several opportunities for further improvement and extension. Future work can focus on integrating deep learning architectures such as Convolutional Neural Networks (CNN) or Long Short-Term Memory (LSTM) networks to automatically learn temporal and spatial dependencies in network traffic data.

In addition, deploying the model in a real-time detection environment would allow evaluation of its scalability, latency, and adaptability to live network conditions. The system could also be expanded to handle encrypted traffic using flow-based features and statistical patterns instead of packet payloads.

Another direction is the inclusion of IoT-specific datasets and federated learning techniques to enhance detection across distributed and resource-constrained devices while maintaining data privacy. Finally, explainable AI methods can be incorporated to provide better interpretability, allowing network administrators to understand model decisions and respond more effectively to potential threats.

14. Github Link

https://github.com/Shrutisingh2005/botnet_detection

15. References

1. Gu, G., Porras, P., Yegneswaran, V., Fong, M., & Lee, W. (2007). *BotHunter: Detecting malware infection through IDS-driven dialog correlation.* Proceedings of the 16th USENIX Security Symposium.

2. Gu, G., Perdisci, R., Zhang, J., & Lee, W. (2008). *BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection.* Proceedings of the 17th USENIX Security Symposium, 139–154.

3. Meidan, Y., Bohadana, M., Mirsky, Y., Breitenbacher, D., Shabtai, A., & Elovici, Y. (2018). *N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders.* IEEE Pervasive Computing / arXiv.

4. Kumar, A., Shridhar, M., Swaminathan, S., & Lim, T. J. (2020). *Machine learning-based early detection of IoT botnets using network-edge traffic (EDIMA).* arXiv; later published/extended in Computers & Security.

5. Doshi, R., Apthorpe, N., & Feamster, N. (2018). *Machine learning DDoS detection for consumer Internet of Things devices.* 2018 IEEE Security & Privacy Workshops (SPW).

6. Chowdhury, M. N., Ferens, K., Ferens, M., & Hossain, M. S. (2017). *Botnet detection using graph-based feature clustering.* Journal of Big Data, 4:14.

7. Iliofotou, M., Pappu, P., & Faloutsos, M. (2011). *Traffic dispersion graphs (TDG) for network monitoring / P2P detection.* ACM/Elsevier proceedings (Traffic Dispersion Graphs paper).

8. García, S., Grill, M., Stiborek, J., & Zunino, A. (2014). *An empirical comparison of botnet detection methods.* Computers & Security, 45, 100–123. (This is the CTU-13 dataset paper.)

9. Moustafa, N., & Slay, J. (2015). *UNSW-NB15: A comprehensive data set for network intrusion detection systems.* Military Communications and Information Systems Conference (MilCIS) / dataset release.

10. (Composite/Survey) Various authors on benchmarks and adversarial robustness (CTU-13, Bot-IoT, UNSW-NB15 guidance). Use the dataset and benchmark papers above for citation as needed.

11. Alshamkhany, M., Alshamkhany, W., Mansour, M., Khan, M., Dhou, S., & Aloul, F. (2020). Botnet attack detection using machine learning. *2020 14th International Conference on Innovations in Information Technology (IIT)*, 203–208. https://doi.org/10.1109/IIT50501.2020.9299061