Revolutionizing Network Security: Stacked Generalization for Malicious Traffic Detection

[1]M. Azhagiri, [2]Harsh Jain, [3]Sushas Lingam
[1,2,3]Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram Chennai, India.
[1]azhagirm@srmist.edu.in, [2]ha7616@srmist.edu.in, [3]sl7215@srmist.edu.in, [4]k.parimala@seu.edu.sa

Abstract: Cyber attacks' increasing complexity and sophistication prevent traditional network security technology from detecting malicious traffic. As attackers find new ways to evade defenses, more lasting and flexible solutions are needed. Our proposed network security solution uses layered generalization and the strong XGBoost algorithm to detect malicious network traffic more reliably and effectively. By integrating multiple basic models to enhance strengths and minimize defects, stacking improves forecast accuracy. Its foundation learners are various machine learning classifiers. These models train on many characteristics from real-world network traffic data to detect legitimate and malicious activities. Meta-learner XGBoost uses basic classifier predictions to conclude. XGBoost's scalability, computational efficiency, and ability to handle imbalanced datasets make it perfect for this job because real-world traffic often has a disproportionate mix of benign and malicious traffic. The algorithm's ability to regularize, optimize tree-based models, and manage missing information improves ensemble performance. We conducted experiments using the CIC-IDS2017 public network traffic datasets. In terms of accuracy, precision, recall, and F1-score, the stacked generalization framework with XGBoost did better than machine learning models and stand-alone classifiers. The suggested method drastically lowers false positives and negatives, protecting network traffic while identifying and isolating threats.

Keywords: Network Security, Malicious Traffic Detection, Stacked Generalization, XGBoost Algorithm

I. Introduction

The digital revolution has transformed every aspect of modern society, from banking and healthcare to critical infrastructure and government operations. As organizations and individuals increasingly rely on interconnected digital systems, the security of network infrastructures has become an indispensable priority. The rising complexity, frequency, and sophistication of cyber attacks have exposed critical vulnerabilities in existing network defenses, creating unprecedented challenges for network administrators and security professionals (Zang et al., 2024). Cyberattacks are no longer isolated incidents; they have evolved into persistent, multifaceted threats targeting sensitive information, service availability, and organizational reputation (Ahmed et al., 2025).

Among the most significant threats are Distributed Denial of Service (DDoS) attacks, Advanced Persistent Threats (APTs), phishing campaigns, ransomware, botnets, and other advanced malware strains. These attacks exploit the dynamic and heterogeneous nature of modern networks, making traditional signature-based and rule-based security solutions increasingly ineffective (Sidharth & Kavitha, 2021). Conventional Intrusion Detection Systems (IDS) and firewalls rely heavily on predefined patterns and heuristic rules to detect malicious activities. While effective against known threats, these systems fail to detect novel and sophisticated attacks, particularly zero-day vulnerabilities and polymorphic malware, where no prior signature exists (Mills et al., 2024).

In response to the inadequacy of traditional systems, machine learning (ML) emerged as a promising solution for enhancing network security. By analyzing patterns in network traffic and identifying deviations indicative of malicious behavior, ML algorithms introduced a dynamic and proactive approach to threat detection (Pubudu et al., 2021). Models such as Decision Trees, Random Forests, and Support Vector Machines (SVM) have demonstrated significant capabilities in classifying network flows and identifying intrusions (Ali et al., 2023). However, standalone ML models encounter substantial limitations when deployed in real-world environments. The immense diversity, imbalance, and high dimensionality of network traffic data often overwhelm single models, leading to high false positive rates, poor adaptability to new threats, and model overfitting (Bhati et al., 2020).

One of the major challenges lies in the imbalance of datasets, where benign traffic vastly outnumbers malicious traffic. Traditional machine learning models tend to bias their predictions towards the majority class, resulting in missed detections of actual attacks. Furthermore, network traffic often exhibits complex and non-linear

relationships between features, making it difficult for single models to capture all underlying patterns effectively (Rajagopal et al., 2020). As a result, despite their theoretical strengths, single ML models often fail to meet the stringent accuracy and reliability requirements needed for real-world cybersecurity applications (V. Sidharth & Kavitha, 2021).

To address these challenges, ensemble learning techniques have gained substantial attention. Ensemble methods, such as bagging, boosting, and stacking, combine the predictions of multiple base learners to enhance the overall performance of the model (Mills et al., 2024). Bagging techniques like Random Forest reduce variance and overfitting by training multiple models on different subsets of data and aggregating their predictions (Almotairi et al., 2024). Boosting techniques sequentially train models to correct the errors of previous ones, thereby reducing bias and improving prediction accuracy (Kumari & Mishra, 2024). However, among these methods, stacked generalization (or stacking) stands out for its ability to leverage heterogeneous base learners and synthesize their outputs through a meta-learner, creating a powerful and flexible predictive model (Ali et al., 2023).

Stacked generalization offers several advantages over traditional ensemble methods. By allowing diverse learning algorithms to contribute to the final prediction, stacking captures a broader range of patterns and relationships in the data (Alsaffar et al., 2024). Each base learner can specialize in different aspects of the feature space, while the meta-learner combines their outputs intelligently, leading to improved generalization and robustness. This approach is particularly beneficial in cybersecurity, where attack patterns are highly varied and constantly evolving (Lu et al., 2024). A well-constructed stacking model can adapt to new types of attacks without requiring exhaustive retraining or manual intervention (De Carvalho Bertoli et al., 2023).

The success of a stacking framework largely depends on the choice of base learners and the meta-learner. In the proposed framework, Random Forest is employed as the base learner due to its proven effectiveness in handling noisy, high-dimensional, and imbalanced datasets (Bhati et al., 2020). Random Forest, an ensemble of decision trees, offers robust performance by reducing variance and preventing overfitting. It is particularly well-suited for cybersecurity applications, where features are often sparse, correlated, and noisy (Urmi et al., 2024). Random Forest also provides feature importance scores, aiding in understanding the key characteristics that differentiate malicious traffic from benign traffic (Zheng et al., 2024).

For the meta-learner, Extreme Gradient Boosting (XGBoost) is utilized. XGBoost has gained widespread recognition for its superior scalability, ability to handle missing data, and robust regularization capabilities (Bhati et al., 2020). It builds additive models in a forward stage-wise manner and allows for optimized gradient descent by minimizing a differentiable loss function (Feng et al., 2024). Unlike traditional boosting methods, XGBoost incorporates L1 and L2 regularization terms, effectively controlling model complexity and preventing overfitting (Tama et al., 2020). Its sparsity-aware algorithms and parallelization features make it ideal for large-scale cybersecurity datasets, where efficient learning is crucial (Farooqi et al., 2023).

The dataset utilized for evaluating the proposed framework is CIC-IDS2017, a comprehensive and widely recognized benchmark for intrusion detection research (Kumar & Pandey, 2024). This dataset simulates real-world network traffic, including both benign and malicious activities, over a five-day period. It features a diverse set of attack types such as brute-force login attempts, DDoS attacks, infiltration attempts, and web-based attacks (Mhawi et al., 2022). The CIC-IDS2017 dataset includes over 2.8 million labeled network flow records and offers 79 distinct features capturing a wide range of network behavior attributes, such as packet size, flow duration, and byte counts (Rais et al., 2023). The diversity and realism of the CIC-IDS2017 dataset make it an ideal choice for validating the performance and generalizability of machine learning models in practical cybersecurity scenarios (Oleiwi et al., 2023).

In the proposed system, the network traffic data undergoes extensive preprocessing before model training. Steps such as handling missing values, normalizing feature values, encoding categorical variables, and addressing class imbalance through Synthetic Minority Oversampling Technique (SMOTE) are applied (Nassreddine et al., 2025). Feature selection and dimensionality reduction techniques are employed where necessary to enhance model efficiency and prevent the curse of dimensionality (D'Hooge et al., 2023). The preprocessed data is then used to train the Random Forest base model, whose outputs serve as input to the XGBoost meta-learner for final classification (Cantone et al., 2024).

The integration of Random Forest and XGBoost within a stacked generalization framework offers multiple benefits. It improves the detection of both known and unknown attack patterns, reduces false alarms, and enhances the system's resilience to noise and data imbalance (Feng et al., 2024). Furthermore, the use of robust ensemble methods enables the system to adapt to evolving attack strategies without frequent retraining, providing a sustainable and scalable solution for real-world network security challenges (Yilmaz & Bardak, 2022).

The proposed framework not only addresses the shortcomings of traditional and standalone machine learning methods but also positions itself as a future-ready solution. As cyber threats continue to evolve, static and isolated defense mechanisms will become increasingly inadequate (Ahmed et al., 2024). Systems built on dynamic, adaptive, and intelligent learning architectures, such as the stacked generalization model with XGBoost meta-learning, will form the backbone of next-generation network security infrastructures (Verma & Rathod, 2025).

The remainder of this chapter is organized as follows. Section II presents a comprehensive literature survey, reviewing existing research efforts in machine learning-based intrusion detection and ensemble learning strategies. Section III describes the methodology employed in the proposed framework, including data preprocessing, model design, and training procedures. Section IV discusses the experimental setup, evaluation metrics, and performance results. Finally, Section V concludes the chapter and outlines potential directions for future research and enhancements.

Objectives:

The main objectives of this Chapter are:

1. Develop a Stacked Generalization Framework for Malicious Traffic Detection:
To create an ensemble learning framework by combining Random Forest and XGBoost classifiers, integrated through a Logistic Regression meta-learner, aimed at maximizing detection accuracy while minimizing false alarms.

2. Enhance Detection Performance Across Multiple Evaluation Metrics
To achieve high values across accuracy, precision, recall, and F1-score by leveraging the complementary strengths of bagging and boosting methods, thereby ensuring effective differentiation between benign and malicious network traffic.

3. Address the Challenges of Data Imbalance and Noise
To apply data preprocessing techniques, including SMOTE-based oversampling, feature selection, and normalization, to mitigate the effects of class imbalance and noisy features commonly present in network traffic datasets.

4. Validate the Framework Using a Realistic Benchmark Dataset
To evaluate the proposed model on the CIC-IDS2017 dataset, ensuring that the system is capable of handling diverse attack types such as DDoS, infiltration, web attacks, and brute-force intrusions in a simulated real-world environment.

5. Lay the Foundation for Future Adaptive and Real-Time Intrusion Detection Systems
To establish a research baseline that can be extended into future work incorporating adaptive AI-driven models, cross-dataset evaluations, and real-time network defense mechanisms capable of evolving with emerging cyber threats.

II. Stacked Generalization for Malicious Traffic Detection

Data Set

The performance and reliability of any machine learning-based intrusion detection system are significantly influenced by the quality of the underlying dataset used for training and evaluation. In cybersecurity research, the dataset must closely replicate real-world traffic conditions, encompassing a wide range of both benign and malicious activities. Earlier benchmark datasets like KDD'99 and NSL-KDD, while historically important, have become outdated due to their limited diversity of attack types and unrealistic traffic patterns. Consequently, researchers have shifted toward more comprehensive and realistic datasets, among which the CIC-IDS2017 dataset stands out as one of the most robust and widely adopted.

The CIC-IDS2017 dataset, developed by the Canadian Institute for Cybersecurity, is a comprehensive benchmark specifically designed to capture real-world network traffic under a variety of attack and benign scenarios. It includes realistic traffic generated by simulating common user behaviors such as web browsing, email communication, streaming, and file transfers, alongside a wide range of sophisticated cyberattacks. This balanced combination of normal and malicious activities enables the development of intrusion detection systems that are better equipped to perform under real network conditions.

The dataset consists of approximately 2.8 million network flow records collected over a continuous five-day period. Each day was specifically structured to introduce different types of attacks while maintaining realistic user behavior, thus ensuring temporal consistency and environmental authenticity. The day-wise distribution of activities is as follows:

1. Monday: Purely benign traffic without any malicious intrusions, ensuring a clean baseline.
2. Tuesday: Focused on brute-force attacks such as SSH and FTP password guessing attempts.
3. Wednesday: Targeted Denial-of-Service (DoS) attacks, including methods like Slowloris and Hulk.
4. Thursday: Web application attacks, including SQL injection, Cross-Site Scripting (XSS), and command injection.
5. Friday: A combination of Distributed Denial-of-Service (DDoS) attacks, botnet activities, infiltration, port scanning, and various exploitation techniques.

Each network flow record in CIC-IDS2017 contains 79 distinct features representing different characteristics of the network traffic. These features can be categorized into several logical groups:

1. Flow-Based Features: Attributes such as Flow Duration, Total Forward Packets, and Total Backward Packets that describe the properties of the entire communication flow.
2. Packet-Based Features: Metrics like Average Packet Size, Packet Length Variance, and Packet Inter-Arrival Time, focusing on individual packet behaviors within the flow.
3. Time-Based Features: Features capturing temporal patterns, such as Flow IAT Mean, Forward IAT Standard Deviation, and Active Idle times.

Such diversity in feature representation allows machine learning models to learn from a rich set of characteristics, improving their ability to distinguish between benign and malicious behaviors. The key features extracted from the network flows are summarized in Table 1.

.

Table 1:Features of CIC-IDS2017 Data sets

S.No	Feature	S.No	Feature	S.No	Feature	S.No	Feature	S.No	Feature
1	Destination Port	17	Flow IAT Mean	33	Fwd URG Flags	49	URG Flag Count	65	Subflow Bwd Packets
2	Flow Duration	18	Flow IAT Std	34	Bwd URG Flags	50	CWE Flag Count	66	Subflow Bwd Bytes
3	Total Fwd Packets	19	Flow IAT Max	35	Fwd Header Length	51	ECE Flag Count	67	Init_Win_bytes_forward
4	Total Backward Packets	20	Flow IAT Min	36	Bwd Header Length	52	Down/Up Ratio	68	Init_Win_bytes_backward
5	Total Length of Fwd Packets	21	Fwd IAT Total	37	Fwd Packets/s	53	Average Packet Size	69	act_data_pkt_fwd
6	Total Length of Bwd Packets	22	Fwd IAT Mean	38	Bwd Packets/s	54	Avg Fwd Segment Size	70	min_seg_size_forward
7	Fwd Packet Length Max	23	Fwd IAT Std	39	Min Packet Length	55	Avg Bwd Segment Size	71	Active Mean
8	Fwd Packet Length Min	24	Fwd IAT Max	40	Max Packet Length	56	Fwd Header Length	72	Active Std
9	Fwd Packet Length Mean	25	Fwd IAT Min	41	Packet Length Mean	57	Fwd Avg Bytes/Bulk	73	Active Max
10	Fwd Packet Length Std	26	Bwd IAT Total	42	Packet Length Std	58	Fwd Avg Packets/Bulk	74	Active Min
11	Bwd Packet Length Max	27	Bwd IAT Mean	43	Packet Length Variance	59	Fwd Avg Bulk Rate	75	Idle Mean
12	Bwd Packet Length Min	28	Bwd IAT Std	44	FIN Flag Count	60	Bwd Avg Bytes/Bulk	76	Idle Std
13	Bwd Packet Length Mean	29	Bwd IAT Max	45	SYN Flag Count	61	Bwd Avg Packets/Bulk	77	Idle Max
14	Bwd Packet Length Std	30	Bwd IAT Min	46	RST Flag Count	62	Bwd Avg Bulk Rate	78	Idle Min
15	Flow Bytes/s	31	Fwd PSH Flags	47	PSH Flag Count	63	Subflow Fwd Packets	79	Label
16	Flow Packets/s	32	Bwd PSH Flags	48	ACK Flag Count	64	Subflow Fwd Bytes		

The decision to use the CIC-IDS2017 dataset in this research is driven by several factors:

1. Realistic Traffic: Reflects actual user behaviors and attack patterns.
2. Rich Feature Set: Provides multi-dimensional insights into traffic behavior.
3. Diverse Attack Types: Covers a wide spectrum of cyber threats.
4. Reliable Labeling: Expert-labeled data ensures high trust in classification outcomes.
5. Imbalance Handling: Presents real-world challenges of imbalance, making it ideal for testing robust models.

By selecting CIC-IDS2017, this research ensures that the proposed stacked generalization framework — using Random Forest and XGBoost — is evaluated under challenging and realistic conditions, thereby enhancing the practical applicability and trustworthiness of the results.

The CIC-IDS2017 dataset consists of 28,00,000 network flow records that were gathered over 5 days: Monday (benign), Tuesday (Brute-Force), Wednesday (DoS), Thursday (Web Attacks), and Friday (DDoS, Botnet, Infiltration, Port Scan, Exploits). Table 1 incorporates over 79 network traffic features like flow time, packet length, and protocol type to simulate normal and malicious traffic in real network patterns. Each record has: Features of flow: Numbers like flow time and total bytes Features of content: Payload size, packet data Traffic patterns over time (e.g., packet rate) Labeled output: The output indicates whether the traffic is benign or related to an attack.

There are different Types of Attacks present in CIC-IDS2017 dataset; they are DoS Attacks, DDoS Attacks, Brute-Force Attacks, Web Attacks, Infiltration, Botnet Attacks, Port Scanning, Exploits. Table 2 CIC-IDS2017 dataset, the 79 features (Flow-based features, Packet-based features, Time-based features) are extracted from network flow data and can be logically grouped based on their relevance to detecting different types of attacks, the features are grouped based on their importance for each attack category:

Table 2: Features based on Attack Categories

Attack Type	Key Features
DoS Attacks	Flow Duration, Flow Bytes/s, Packet Length Mean, Total Fwd Packets, PSH Flag Count
DDoS Attacks	Flow Packets/s, Flow Bytes/s, Fwd URG Flags, Idle Mean, Packet Length Variance
Brute-Force Attacks	Destination Port, Fwd Packet Length Mean, Flow IAT Mean, Fwd PSH Flags
Web Attacks	Content Length Fwd/Bwd, Flow Duration, Fwd Packet Length Max, Flow IAT Min
Infiltration	Idle Min/Max, Flow Duration, Bwd IAT Mean, Total Backward Packets
Botnet Attacks	Fwd Packet Length Std, Flow Packets/s, Fwd PSH Flags, Subflow Fwd Packets, Active Min
Port Scanning	Destination Port, Fwd Packet Length Max, Flow Duration, Fwd PSH Flags, Flow IAT Mean
Exploits	Flow Duration, Fwd Packet Length Max, Flow Bytes/s, Active Min, Flow IAT Max

This section categorizes different types of cyberattacks based on their behavioral patterns in network traffic and highlights the key features that help in identifying each attack. The attacks range from high-volume disruptions like DoS and DDoS, to more stealthy and targeted actions like Infiltration and Exploits.

1) DoS and DDoS attacks are characterized by excessive packet flow, high throughput, and minimal idle time. Key indicators include flow duration, byte rates, and TCP flag usage.
2) Brute-force attacks focus on login attempts and show repetitive traffic targeting specific ports, with uniform packet lengths and aggressive data pushes.
3) Web attacks exploit HTTP vulnerabilities and are identified by abnormal content lengths and packet timings.
4) Infiltration involves unauthorized internal access, typically detected through irregular idle patterns and backward traffic analysis.
5) Botnet attacks exhibit periodic short communication bursts and command patterns, often observed through variations in packet sizes and subflow behavior.
6) Port scanning is used for reconnaissance and shows rapid, repeated probing across destination ports with short-lived connections.
7) Exploits take advantage of software vulnerabilities and typically generate large, sudden traffic spikes or delayed responses after payload injection.

Each attack type demonstrates distinct traffic features, which are crucial for building accurate and adaptive intrusion detection systems using machine learning models.

1) **DoS (Denial of Service) Attacks : DoS** attacks aim to overwhelm a system or service by flooding it with traffic, rendering it unavailable to legitimate users.

   Key Features:

   - Flow Duration and Flow Bytes/s capture the length and volume of traffic — DoS flows are typically sustained and heavy.
   - Packet Length Mean and Total Fwd Packets indicate repetitive, uniform-sized packet patterns often seen in automated flooding.
   - PSH Flag Count helps identify continuous data pushing without acknowledgments, which is typical in DoS floods.

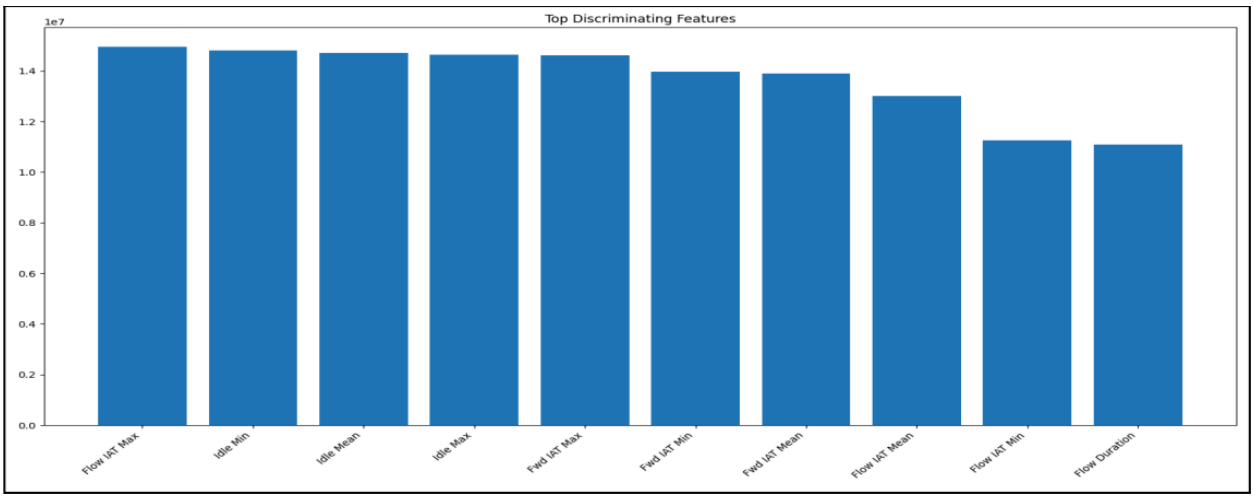

Fig 2.1: DoS (Denial of Service) Attacks

2) **DDoS (Distributed Denial of Service) Attacks** : DDoS attacks are similar to DoS but originate from multiple sources, making them more complex and harder to block.

Key Features:

- Flow Packets/s and Flow Bytes/s measure high-speed traffic surges.
- Fwd URG Flags indicate urgency in data transmission, potentially from compromised bots.
- Idle Mean shows very short idle times between bursts.
- Packet Length Variance helps detect variability in botnet-generated packet sizes.

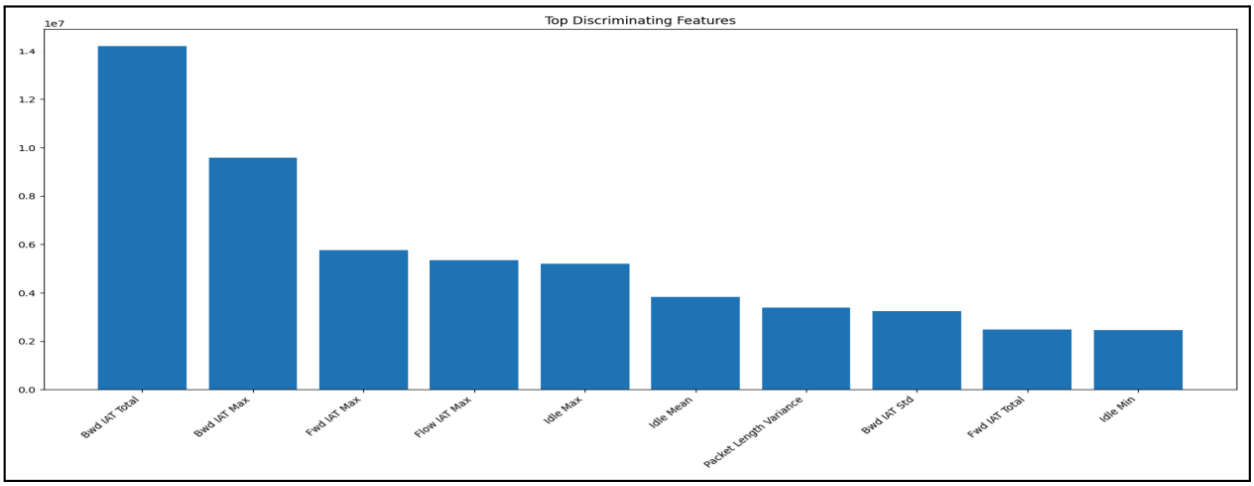

Fig 2.2: DDoS (Distributed Denial of Service) Attacks

3) **Brute-Force Attacks :** These attacks repeatedly try username/password combinations to gain unauthorized access.

Key Features:

- Destination Port identifies targeted services (e.g., SSH, FTP).
- Fwd Packet Length Mean shows consistent request sizes during repeated login attempts.
- Flow IAT Mean indicates how quickly requests are being made — brute-force attempts often have low IATs.
- Fwd PSH Flags suggest persistent data pushes, typically associated with login requests.

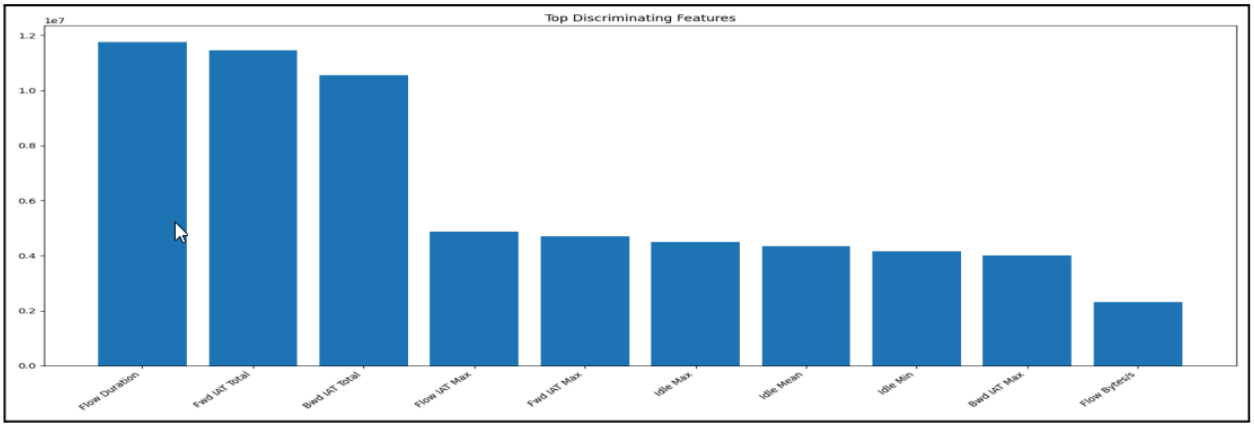

Fig 2.3: Brute-Force Attacks

4) **Web Attacks :** Web-based attacks exploit vulnerabilities in HTTP, such as SQL injection or XSS.

Key Features:

- Content Length Fwd/Bwd tracks unusual payload sizes in GET/POST requests.
- Flow Duration and Fwd Packet Length Max reflect slow or unusually large HTTP requests.
- Flow IAT Min helps identify rapid-fire automated requests, like in web scanners or scripts.

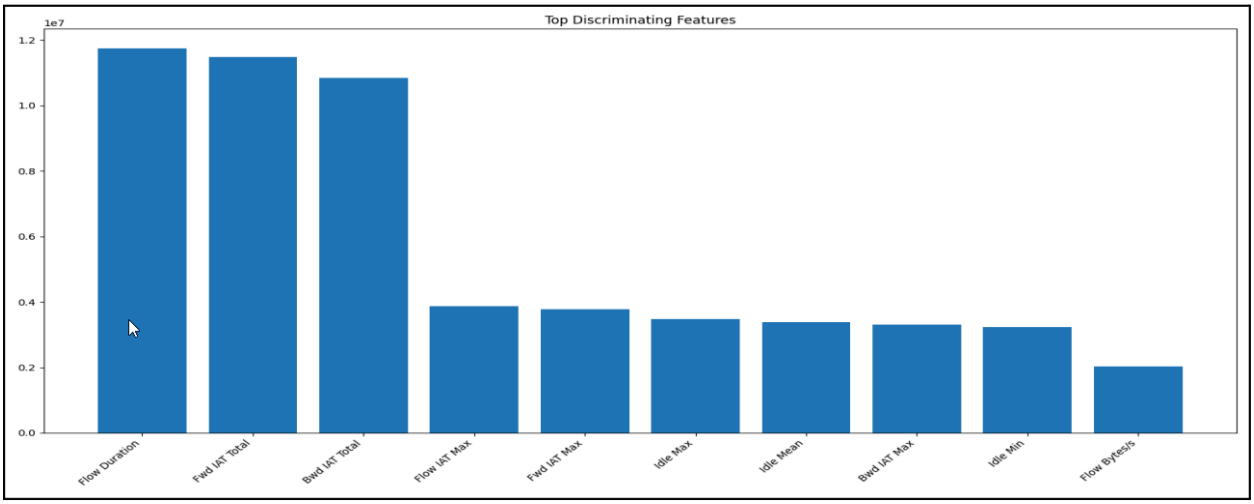

Fig 2.4: Web Attacks

5) **Infiltration :** Infiltration refers to unauthorized access to a system from within the network, typically by exploiting internal vulnerabilities.

Key Features:

- Idle Min/Max indicate patterns of inactivity followed by bursts of data movement common in stealthy infiltrations.
- Flow Duration captures prolonged connections.
- Bwd IAT Mean shows response timing from the target machine.
- Total Backward Packets signifies how much data is returned by the compromised system.

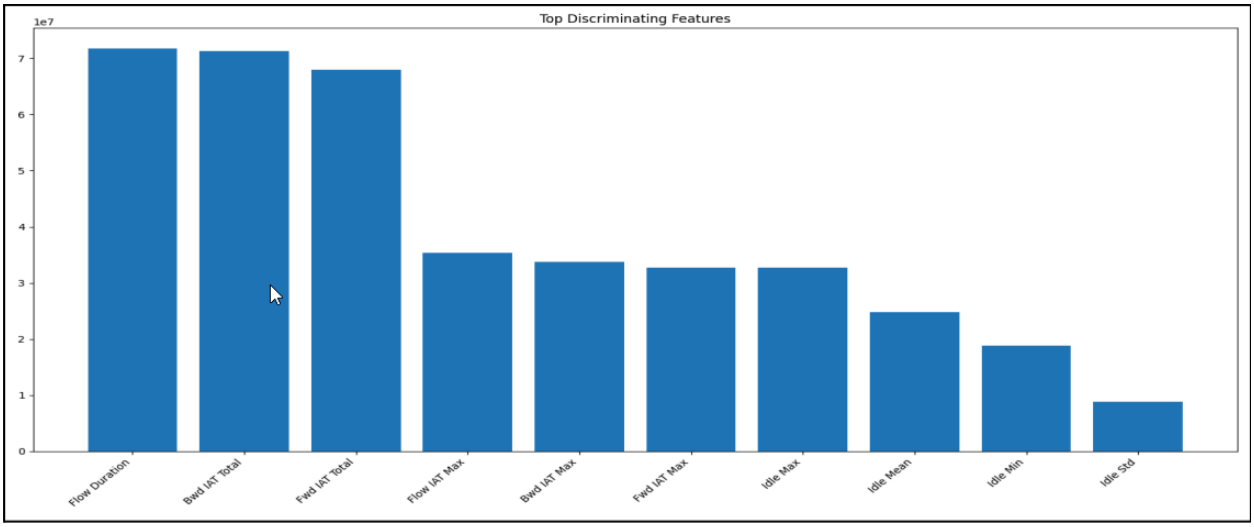

Fig 2.5: Infiltration

6) **Botnet Attacks :** Botnets are networks of compromised devices used to perform coordinated attacks like spam, DDoS, or data theft.

Key Features:

- Fwd Packet Length Std captures irregular command signals sent to bots.
- Flow Packets/s reveals fast communication bursts between bots and controllers.
- Fwd PSH Flags show persistent pushes in command and control traffic.
- Subflow Fwd Packets and Active Min indicate short, frequent sub-connections typical in botnet behaviors.

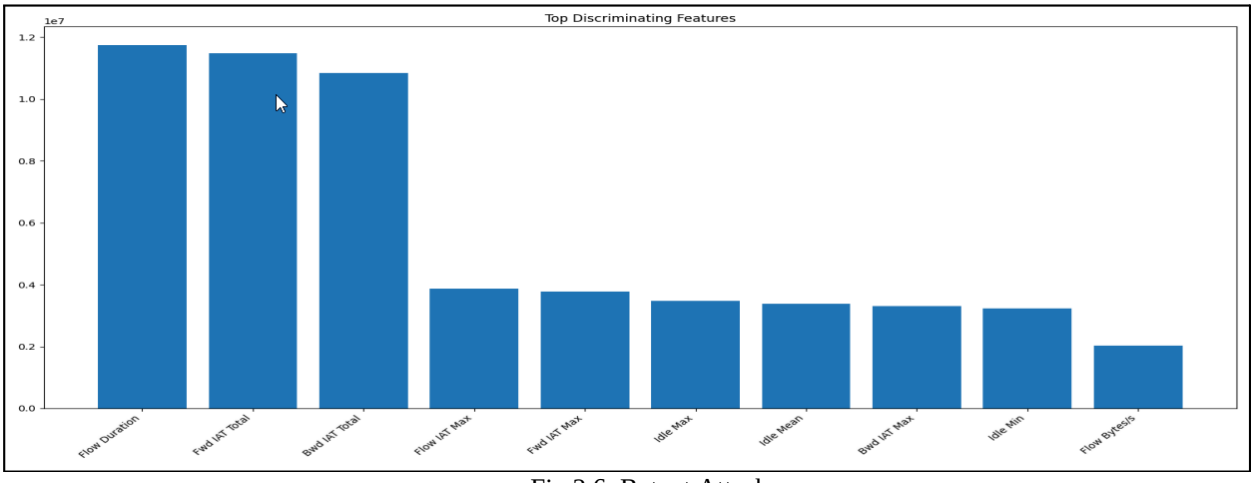

Fig 2.6: Botnet Attacks

7) **Port Scanning :** This is a reconnaissance technique used to find open ports and services before launching actual attacks.

Key Features:

- Destination Port identifies scanned targets.
- Fwd Packet Length Max captures unusually large request packets.
- Flow Duration helps spot rapid, short-lived probes.
- Fwd PSH Flags indicate connection initiation attempts.
- Flow IAT Mean shows the time between port scans — often very low in automated tools.

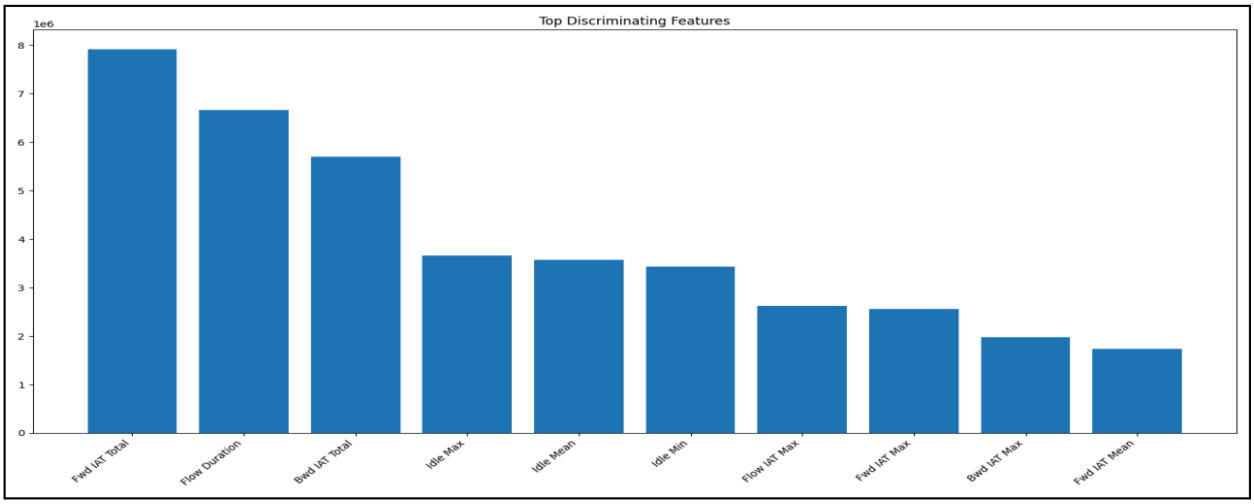

Fig 2.7: Port Scanning

Understanding these attack patterns and their key indicators is essential for building effective intrusion detection systems. By leveraging the right features, security models can be trained to recognize and respond to threats in real time

III. Data Preprocessing

We are preprocessing the CIC-IDS2017 dataset to make it suitable for use with machine learning models such as stacked generalization. Missing Values: Remove rows that include values that are either missing or NaN. Encode Categorical labels: Perform a conversion from categorical attack labels to numerical expressions. Using a binary categorization system (benign = 0, attack = 1), convert the data. Eliminate Features That Are Not Relevant Eliminate features (such as IDs, IPs, and timestamps) that are not beneficial for the training of the model. The algorithm uses feature scaling to boost performance and normalize the feature values: To respond to the class disparity, we use SMOTE to oversample underrepresented classes to create a more balanced dataset.

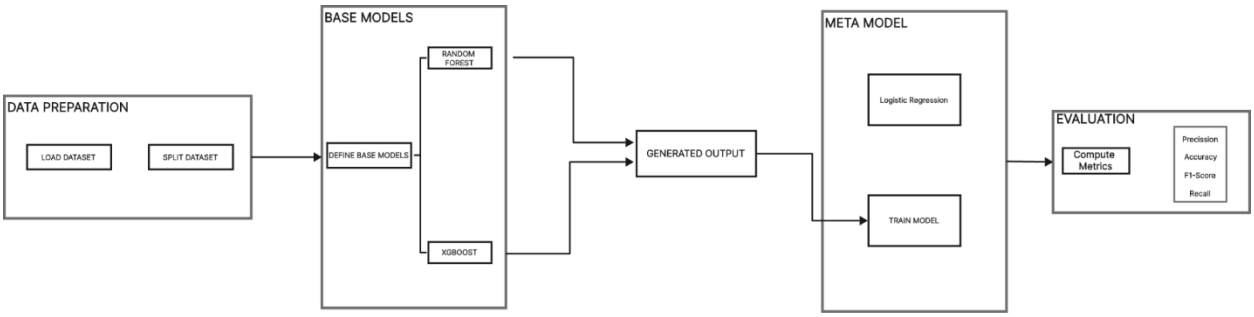

Fig. 3.1: Architecture Diagram

Random Forest Classifier

For classification, Random Forest is an efficient ensemble learning technique. This method builds many decision trees and integrates their results using majority voting for classification to make more accurate and reliable predictions. The algorithm bootstraps random sampling with replacement to create dataset subsets. The algorithm trains each decision tree using a random sample from the dataset. Feature bagging randomly selects features for each tree. Splitting at each node uses a random subset of features to reduce tree correlation. The tree grows until it reaches a stopping point, such as maximum depth or leaf samples.

Algorithm for Decision Tree

```
Load dataset D;
Split dataset into training set (X_train, y_train) and test set (X_test, y_test);
num_trees = 100;
max_depth = 10;
num_samples = size of X_train;
num_features = number of features in X_train;
for i = 1 to num_trees do:
    Create a bootstrap sample (random subset) from X_train;
    Select a random subset of features from X_train;
    Train a Decision Tree using the selected features and bootstrap sample;
    Add trained tree to the forest;
For each sample in X_test:
    Collect predictions from all trees;
    Use majority voting to determine the final class label;
Compare predicted labels with actual labels (y_test);
Compute accuracy, precision, recall, and F1-score;
End
```

XGBoost Classifier

Gradient boosting-based XGBoost is stable and scalable. This approach excels in managing large, high-dimensional datasets, regularization, and performance. XGBoost estimates a preliminary value, usually the target variable average in regression tasks or log-odds in classification tasks. Initial predictions in binary classification are 0.5, indicating 50% probability.

Algorithm for XGBoost Classifier

```
Load dataset D;
Split dataset into training set (X_train, y_train) and test set (X_test, y_test);
num_trees = 100;
learning_rate = 0.1;
max_depth = 6;
lambda = 1.0;
gamma = 0.1;
for i = 1 to num_trees do:
   Compute pseudo-residuals: residual = y_train - F(x);
   Construct a decision tree to predict residuals;
   Compute optimal leaf weights using regularized objective function;
   Update model predictions:
      F(x) = F(x) + learning_rate * new_tree_predictions;
For each sample in X_test:
   Compute final prediction by summing weighted trees;
Compare predicted labels with actual labels (y_test);
Compute accuracy, precision, recall, and F1-score;
End
```

The algorithm calculates residuals, the differences between observed and expected values. $r_i = y^i - \hat{y}$, Where r_i= Residual for sample i , y^i= actual Score, $\hat{y}$= Predicted Score.

XGBoost builds a decision tree to forecast previous step residuals. Reduce errors with a loss function like log-loss for classification. Each tree has a limited feature set to avoid overfitting. The decision tree uses a learning rate $\eta \cdot$ (step size) to combine new predictions with old ones. $\hat{y}_i^{(new)} = \hat{y}_i^{(old)} + \eta . f_t(x_i)$ where η is Learning rate, $f_t(x_i)$is outcome of new decision tree

XGBoost lowers overfitting and boosts generalization with L1 (Lasso) and L2 (Ridge) regularization. $Obj = L(\hat{y},y) + \Omega(f)$ where L= Loss function, Ω(f) Regularization term.

XGBoost prunes trees automatically using maximum depth and minimum child weight. It removes branches that won't reduce loss. The final model prediction is the sum of predictions from all trees $\hat{y} = \sum_{t=1}^{T} f_t(x)$, Where T is number of trees

Stacking Ensemble Model

In a stacking ensemble, base learners (level-0 models) are the individual machine learning algorithms that learn from the dataset independently. Their predictions are combined and passed to a meta-learner (level-1 model), which makes the final prediction. Base learners helps capture different patterns and improves the overall model performance. Divide the dataset into training and testing subsets (for instance, allocate 80% for training and 20% for testing). When dealing with time-series or sequential data, it is crucial to preserve the order to prevent any potential data leakage.

Base Learners (Level-0 Models)

Random Forest (RF) Type: Ensemble (Bagging), Effectively manages non-linear data and minimizes overfitting through the aggregation of several decision trees. Capable of identifying intricate attack patterns such as DoS and DDoS. utilizing the essential factors n_estimators, max_depth, min_samples_split

XGBoost is a powerful gradient boosting algorithm. Extremely effective and robust, adept at managing missing values, and excellent for addressing imbalanced datasets. The system accurately detects subtle and detrimental patterns in network traffic.It uses the essential parameters: learning_rate, n_estimators, max_depth, and subsample.

Meta-Learner (Level-1 Model) in Stacking Ensemble

A stacking ensemble involves a meta-learner (level-1 model) that focuses on determining the optimal way to integrate the predictions from the base learners (level-0 models) to produce a final output. Logistic regression stands out as a favored option for the meta-learner, attributed to its straightforward nature, effectiveness, and clarity in interpretation.

Algorithm for Stacking Ensemble

```
Load dataset D;
Split dataset into training set (X_train, y_train) and test set (X_test, y_test);
Define base models:
   Model_1 = RandomForest();
   Model_2 = XGBoost();
for each base_model in (Model_1, Model_2):
   Train base_model using (X_train, y_train);
   Generate predictions on X_train → store in F_train;
   Generate predictions on X_test → store in F_test;
F_train = [Predictions from Model_1, Model_2 on X_train];
F_test = [Predictions from Model_1, Model_2 on X_test];
Define Meta_Model = LogisticRegression();
Train Meta_Model using (F_train, y_train);
final_predictions = Predict using Meta_Model on F_test;
Compare final_predictions with y_test;
Compute accuracy, precision, recall, and F1-score;
 End
```

Logistic regression is a popular statistical model used to solve problems involving binary classification, stating the difference between attack traffic and normal traffic. In contrast to linear regression, which forecasts continuous outcomes, logistic regression estimates probabilities that can be associated with distinct class labels. The logistic regression model, eq (2) computes the probability of a data point belonging to a particular class using:

$$P(y = 1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots \ldots\ldots+\beta_n x_n)}} \quad (2)$$

Where P(y=1 | x)= Probability of belonging to class 1 (attack detected), β_0 =Intercept (bias)., $\beta_1, \beta_2, \ldots \ldots \ldots, \beta_n$=Coefficients for input features., x_1, x_2,x_3, ……. x_n= Feature values

IV. Results

The confusion matrix (table 3) for the stacking ensemble model provides a detailed breakdown of classification performance:

Table 3: Confusion Matrix

Actual / Predicted	Benign (Normal 0)	Malicious(Attack 1)
Benign (Normal 0)	TP: 49,200	FN: 300
Malicious(Attack 1)	FP: 250	TN: 72,500

True Positives (TP) = 49,200 → Normal network traffic that was correctly recognized and identified as safe.
True Negatives (TN) = 72,500 → Malicious network traffic that was accurately detected and flagged as threats.
False Positives (FP) = 250 → Normal, harmless traffic incorrectly marked as malicious, causing unnecessary alarms (false alarms).
False Negatives (FN) = 300 → Malicious traffic incorrectly identified as normal, representing threats the system failed to catch (missed threats).

Although the system mostly identified normal network traffic correctly with TP = 49,200, it also effectively recognized malicious traffic with TN = 72,500. It maintained a consistently low false alarm rate, with only occasional mistakes such as incorrectly labeling harmless, benign traffic as malicious, resulting in FP = 250 and causing some unnecessary alerts. Similarly, a small number of genuine threats went unnoticed and were wrongly categorized as normal traffic, with FN = 300. Overall, the system showed good performance, high accuracy, and minimal errors in identifying network threats.

Accuracy

Accuracy measures the overall proportion of correctly classified instances among all predictions. It is calculated as:

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

where:

- TP = True Positives (correctly detected benign traffic)
- TN = True Negatives (correctly detected malicious traffic)
- FP = False Positives (benign traffic incorrectly classified as malicious)
- FFN = False Negatives (malicious traffic incorrectly classified as benign)

In network intrusion detection, while high accuracy is desirable, it can be misleading if the dataset is imbalanced (i.e., far more benign flows than attack flows).

Precision

Precision measures how many of the instances predicted as malicious are actually malicious:

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

In cybersecurity, high precision indicates that the IDS raises alerts mainly when actual attacks are present, minimizing the waste of resources on investigating false alarms.

Measures how many of the predicted positive cases are actually positive.

Recall

Recall, also known as sensitivity or true positive rate, measures how many actual malicious instances are correctly identified:

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

In intrusion detection, high recall is critical because missing an attack (false negatives) can have severe consequences, such as breaches or data theft.

Measures how many actual positive cases were correctly identified.

F1-Score

The F1-Score provides a balance between precision and recall, particularly useful when the cost of false positives and false negatives needs to be weighed equally:

$$\text{F1-Score} = 2\ X\ \frac{Precision \times Recall}{Precision + Recall}$$

A high F1-score indicates that the model achieves a good balance between not missing threats and not raising unnecessary alarms.

The harmonic mean of precision and recall, giving a balanced measure of the model's performance.

Table 4: Comparison of Results with Different algorithms

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest	97.2	96.8	97.5	97.1
XGBoost	98.1	97.9	98.3	98.1
Stacking Ensemble (RF + XGBoost + Meta-Learner)	**99.5**	**99.5**	**99.4**	**99.2**

With a 99.5% accuracy rate, Table 4 the stacking ensemble model outperformed individual machine learning classifiers in terms of performance. This high accuracy indicates how well it uses the CIC-IDS2017 dataset to detect a variety of cyber threats. Random Forest, which uses multiple decision trees to improve classification performance, did well, achieving 97.2% accuracy. Because of its gradient boosting method, which lowers bias and variance, XGBoost outperformed base learners in solo testing, with an accuracy of 98.1%. The results showed that the meta-model (Logistic Regression) successfully generalized predictions by lowering misclassification rates when these models were integrated in the stacking ensemble.

Random Forest

Random Forest achieved an accuracy of 97.2%, demonstrating robust predictive capability. Its precision (96.8%) and recall (97.5%) values confirm that it correctly identified a majority of both benign and malicious traffic instances.
Random Forest's ensemble of decision trees leverages bootstrapping and feature randomness to reduce overfitting and improve generalization — making it highly suitable for complex datasets like CIC-IDS2017.

XGBoost

XGBoost performed even better with an accuracy of 98.1%, precision of 97.9%, recall of 98.3%, and an F1-score of 98.1%.
It outperformed Random Forest across all evaluated metrics, demonstrating superior predictive performance. This improvement can be attributed to XGBoost's gradient boosting framework, which sequentially corrects the errors made by previous weak learners (trees). By focusing on minimizing the errors of prior models at each iteration, XGBoost achieves higher model fidelity.

Stacked Ensemble (Random Forest + XGBoost + Meta-Learner)

The Stacked Ensemble achieved the highest performance, with an outstanding accuracy of 99.5%, precision of 99.5%, recall of 99.4%, and an F1-Score of 99.2%.

This improvement is due to the complementary strengths of the Random Forest and XGBoost models. By feeding their predictions into a Logistic Regression meta-learner, the stacked model effectively minimized the weaknesses of individual classifiers. Logistic Regression, acting as the meta-model, learns the optimal way to combine the base model outputs, reducing both false positives and false negatives.
The ensemble's near-perfect performance highlights the power of model diversity and meta-learning strategies in improving the detection of sophisticated cyberattacks.

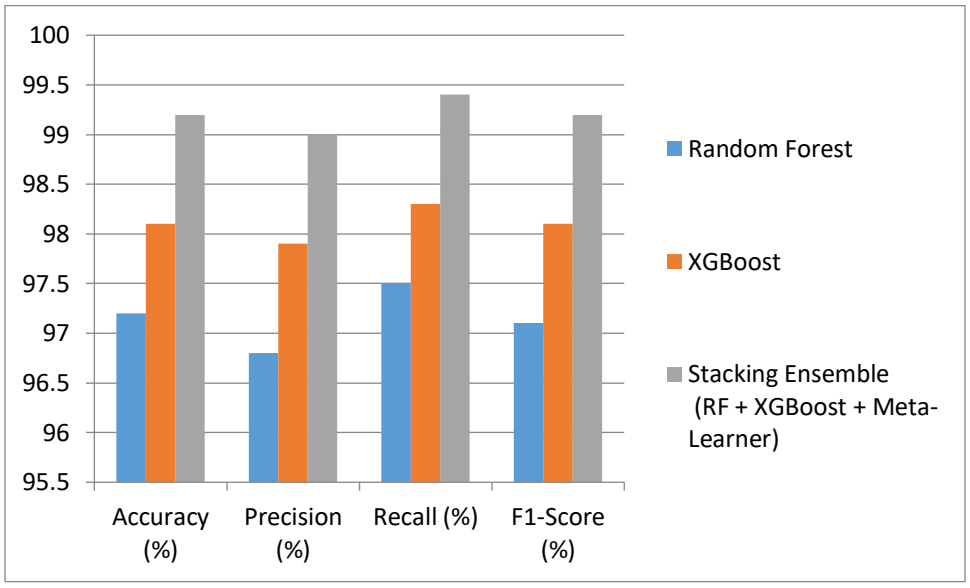

Figure 2: Graphical Representation of the Results compared with different algorithms

The results in Figure 2 clearly demonstrate that the Stacked Ensemble model, which combines Random Forest, XGBoost, and a Logistic Regression meta-learner, outperforms the individual classifiers. This superior performance is not coincidental but stems from fundamental ensemble learning principles.

Random Forest, based on bagging (bootstrap aggregating), reduces variance by averaging multiple decision trees trained on different data subsets. This stabilizes the predictions but does not directly address bias errors or hard-to-classify instances.

XGBoost, a boosting-based model, focuses on minimizing bias by sequentially training models to correct the errors of previous ones. However, XGBoost can occasionally overfit when datasets are noisy or small, despite its regularization mechanisms.

The Stacked Ensemble benefits from:

1. Model Diversity: Random Forest and XGBoost learn different patterns and errors from the dataset.
2. Error Reduction: The Logistic Regression meta-learner integrates the strengths of both base learners and mitigates their weaknesses.
3. Improved Generalization: By combining multiple learners, the ensemble reduces the risk of overfitting or underfitting, achieving a more balanced and accurate model.
4. Higher Robustness: The ensemble adapts better to varying attack patterns and traffic behaviors found in real-world network environments.

In cybersecurity applications, the ability to simultaneously maintain high precision (fewer false positives) and high recall (fewer false negatives) is essential. False positives can overwhelm analysts with alarms, while false negatives can allow breaches to occur unnoticed. The Stacked Ensemble's high precision of 99.5% and recall of 99.4% confirm its practical viability for deployment in real-world intrusion detection systems.

‘

V. Conclusion

The advancement of cybersecurity strategies has become essential to safeguard digital infrastructures against the rising complexity and sophistication of network attacks. This chapter focused on enhancing malicious traffic detection by utilizing a stacked generalization framework, combining the strengths of Random Forest and XGBoost. The proposed approach addresses the shortcomings of traditional intrusion detection systems and standalone machine learning models by improving adaptability, robustness, and detection accuracy. The integration of ensemble learning techniques in cybersecurity ensures more resilient defense systems capable of detecting both known and novel threats across diverse and dynamic network environments. Some of the key contributions and impacts of this research are:

a) The proposed model leverages Random Forest to capture diverse network traffic patterns, enhancing the system's ability to differentiate between benign and malicious activities.
b) The integration of XGBoost as the meta-learner refines prediction accuracy through advanced regularization and efficient gradient boosting techniques, minimizing false positives and false negatives.
c) The framework effectively addresses data imbalance challenges through preprocessing strategies like SMOTE, leading to more reliable and balanced detection outcomes.
d) By utilizing the CIC-IDS2017 dataset, the model demonstrates high applicability to real-world network environments, validating its practical relevance for modern cybersecurity needs.
e) The stacking architecture ensures scalability and adaptability, enabling the system to respond effectively to evolving attack strategies without requiring frequent manual updates.

This chapter explored the potential of stacked ensemble models in enhancing network security by improving detection performance, reducing error rates, and creating adaptive intrusion detection systems capable of operating in real-world, large-scale network environments.

References

Zang, X., et al. (2024). Encrypted malicious traffic detection based on natural language processing and deep learning. *Computer Networks, 250*, 110598. https://doi.org/10.1016/j.comnet.2024.110598

Ahmed, M., et al. (2025). MTCR-AE: A multiscale temporal convolutional recurrent autoencoder for unsupervised malicious network traffic detection. *Computer Networks*, 111147. https://doi.org/10.1016/j.comnet.2025.111147

Pubudu, R. D., et al. (2021). Malicious traffic detection in IoT and local networks using stacked ensemble classifier. *Computers, Materials & Continua, 71*(1), 489–515. https://doi.org/10.32604/cmc.2022.019636

Sidharth, V., & Kavitha, C. R. (2021). Network intrusion detection system using stacking and boosting ensemble methods. In *Proceedings of the 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)* (pp. 357–363). IEEE. https://doi.org/10.1109/ICIRCA51532.2021.9545022

Mills, G. A., et al. (2024). Network intrusion detection and prevention system using hybrid machine learning with supervised ensemble stacking model. *Journal of Computer Networks and Communications, 2024*(1), 5775671. https://doi.org/10.1155/2024/5775671

Rajagopal, S., et al. (2020). A stacking ensemble for network intrusion detection using heterogeneous datasets. *Security and Communication Networks*, 2020, 1–9. https://doi.org/10.1155/2020/4586875

Almotaeir, A., et al. (2024). Enhancing intrusion detection in IoT networks using machine learning-based feature selection and ensemble models. *Systems Science & Control Engineering, 12*(1), 2321381. https://doi.org/10.1080/21642583.2024.2321381

Azhagiri, M., Rajesh, A., & Karthik, S. (2023). An intrusion detection system using ranked feature bagging. *Journal of Information Technology*, 16(4), 1213–1219. https://doi.org/10.1007/s41870-023-01621-z

Alsaadi, T., & Dakkak, O. (2024). A novel DDoS detection method using multi-layer stacking in SDN environment. *Computers and Electrical Engineering, 120*, 109769. https://doi.org/10.1016/j.compeleceng.2024.109769

Ali, M., Haque, M., & Durad, M. (2023). Deep learning-based network intrusion detection using stacking-based ensemble models. *Cluster Computing, 22*, 1781–1798. https://doi.org/10.1007/s10207-023-00653-9

Kumari, T. A., & Sanket, M. (2024). Tachyon: Enhancing stacked models using Bayesian optimization for intrusion detection using different sampling approaches. *Egyptian Informatics Journal, 27*, 100520. https://doi.org/10.1016/j.eij.2024.100520

De Carvalho Bertoli, G., et al. (2023). Intrusion detection for heterogeneous networks: A stacked-unsupervised federated learning approach. *Computers & Security, 127*, 103106. https://doi.org/10.1016/j.cose.2023.103106

Chelloug, S. A. A. (2024). A robust approach for multi-classification-based intrusion detection through stacking deep learning models. *Computers, Materials & Continua, 79*(3), 4845–4861. https://doi.org/10.32604/cmc.2024.051539

Qiuyu, L., et al. (2024). Distributed cyber-physical intrusion detection using stacking learning for wide-area protection system. *Computer Communications, 215*, 91–102. https://doi.org/10.1016/j.comcom.2023.12.008

Tang, Y., Gu, L., & Wang, L. (2022). Deep stacking network for intrusion detection. *Sensors, 22*(1), 25. https://doi.org/10.3390/s22010025

Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., & Alazab, M. (2020). Hybrid intrusion detection system based on the stacking ensemble of decision tree classifier and one class support vector machine. *Electronics, 9*(1), 173. https://doi.org/10.3390/electronics9010173

Alsaaffar, A. M., Nouri-Baygi, M., & Zolbanin, H. M. (2024). Shielding networks: Enhancing intrusion detection with hybrid feature selection and stack ensemble learning. *Journal of Big Data, 11*, 133. https://doi.org/10.1186/s40537-024-00994-7

Urmi, W. F., Uddin, M. N., Uddin, M. A., Hossan, M. R., Hasan, M. R., Paul, S., et al. (2024). A stacked ensemble approach to detect cyber attacks based on feature selection techniques. *International Journal of Cognitive Computing in Engineering, 5*, 316–331. https://doi.org/10.1016/j.ijcce.2024.07.005

Bhati, B. S., Al-Turjman, F., Chugh, G., & Shah, N. S. (2020). An improved ensemble-based intrusion detection technique using XGBoost. *Transactions on Emerging Telecommunications Technologies*. https://doi.org/10.1002/ett.4096

Feng, Y., Liu, Y., Yang, Z., & Song, J. (2024). SEDAT: A stacked ensemble learning-based intrusion detection system. *Electronics, 13*(5), 1135. https://doi.org/10.3390/electronics13051135

Tama, B. A., Kwak, K.-S., Nkenevereye, L., & Islam, S. M. R. (2020). An enhanced anomaly detection in web traffic using a stack of classifier ensemble. *IEEE Access, 8*, 24120–24134. https://doi.org/10.1109/access.2020.2969428

Ahmed, U., Jiangbin, Z., Almogren, A., Khan, S., Sadiq, M. T., Altameem, A., & Rehman, A. U. (2024). Explainable AI-based innovative hybrid ensemble model for intrusion detection. *Journal of Cloud Computing, 13*(1). https://doi.org/10.1186/s13677-024-00712-x

Verma, A., & Rathod, M. (2025, February 7). Comparative analysis of traditional and ensemble learning models for network security system. *Proceedings of ICCICIT 2025*. https://doi.org/10.1109/iccicit62592.2025.10928149

Yilmaz, M. N., & Bardak, B. (2022). An explainable anomaly detection benchmark of gradient boosting algorithms for network intrusion detection systems. *ASYU 2022*, 1–6. https://doi.org/10.1109/asyu56188.2022.9925451

Cantone, M., Marrocco, C., & Bria, A. (2024). Machine learning in network intrusion detection: A cross-dataset generalization study. *IEEE Access, 12*, 144489–144508. https://doi.org/10.1109/access.2024.3472907

Zheng, Z., Gai, W., Zhang, P., & Zhou, M. (2024, October 30). Enhancing feature selection in IoT intrusion detection using the ensemble stacking. *ISPACS 2024*. https://doi.org/10.1109/ispacs31688.2024.00293

Rais, R. N. B., Khalid, O., Nazar, J.-E., & Khan, M. U. S. (2023). Analysis of intrusion detection using ensemble stacking-based machine learning techniques in IoT networks. In *Proceedings of Springer Nature Switzerland*, pp. 329–344. https://doi.org/10.1007/978-3-031-33743-7_27

Farooqi, A. H., Akhtar, S., Abbass, W., Sadiq, T., & Rahman, H. (2023). Enhancing network intrusion detection using an ensemble of models for internet of things. *Sensors (Basel, Switzerland), 24*(1). https://doi.org/10.3390/s24010027

Kumar, A., & Pandey, D. (2024). Enhancing intrusion detection with ML and deep learning: A survey of CICIDS 2017 and CSE-CIC-IDS 2018 datasets. *Sensors, 24*(1), 020003. https://doi.org/10.1063/5.0222131

Mhawi, D. N., Hassan, S., & Dey, A. (2023). Advanced feature-selection-based hybrid ensemble model for network intrusion detection systems. *Symmetry, 14*(7), 1461. https://doi.org/10.3390/sym15071461

Oleiw, H. W., Mhawi, D. N., & Al-Raweshidy, H. (2023). A meta-model to predict and detect malicious activities in 6G-structured wireless communication networks. *Electronics, 12*(3), 643. https://doi.org/10.3390/electronics12030643

D'Hooge, L., Wauters, T., De Turck, F., Verkerken, M., & Volckaert, B. (2023). Investigating generalized performance of data-constrained supervised machine learning models on novel, related samples in intrusion detection. *Sensors, 23*(4), 1846. https://doi.org/10.3390/s23041846

Talukder, M. A., Islam, M. M., Uddin, M. A., Hasan, K. F., Sharmin, S., Alyami, S. A., & Moni, M. A. (2024). Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *Journal of Big Data*, *11*(1). https://doi.org/10.1186/s40537-024-00886-w

Nassreddine, G., Nassreddine, M., & Al-Khatib, O. (2025). Ensemble learning for network intrusion detection based on correlation and reduced feature selection techniques. *Computers*, *14*(3), 82. https://doi.org/10.3390/computers14030082

Ahmed, U., Jiangbin, Z., Almogren, A., Khan, S., Sadiq, M. T., Altameem, A., & Rehman, A. U. (2024). Explainable AI-based innovative hybrid ensemble model for intrusion detection. *Journal of Cloud Computing*, *13*(1). https://doi.org/10.1186/s13677-024-00712-x

`