A DEEP LEARNING-BASED SMART SCALABLE, AND ADAPTIVE DDOS DEFENCE SYSTEM

Mr. Mohnish Saxena[1], Prof Sudhir Goswami[2]

[1]M. TECH (Computer Science &Engineering), [2]Assistant Professor,

Computer Science & Engineering, School of Research & Technology, Peoples University

Bhopal, India

Abstract

(DDoS) attacks a significant risk to CAI traits of online services. To address these threats, organizations can use machine learning (ML) and deep learning (DL) approaches. By implementing this sophisticated technique, they can effectively identify, categorize, and forecast DDoS attacks, enabling initiative-taking measures to be taken before an attack occurs. Investigating (RF)s for DDoS Protection

(RF)s have become a particularly useful tool for analysis and thwarting DDoS assaults among the variety of ML methods. This study explores the use of (RF)s for DDoS defense, outlining both their advantages and disadvantages. (RF)s were effectively used to identify and stop a DDoS assault, as demonstrated in a real-world case study.

The analysis suggests that (RF)s are advantageous over their resilience to noise and outliers, with their demonstrated effectiveness in myriad studies. Nevertheless, further research is essential to innovate and enhance (RF)-based methods for more effective DDoS prevention.

Keywords: DDoS attacks, Machine Learning (ML), Random Forest (RF), DDoS prevention, Intrusion detection

1. Introduction

Distributed Denial-of-Service (DDoS) attacks are intentional efforts to disrupt the normal operations of websites or online services, making them inaccessible to legitimate users. The two main categories of these assaults are volumetric attacks, which use technique like SYN or UDP floods to overload the victim with traffic, and application-layer attacks, which focus on exhausting specific applications through numerous requests, such as HTTP floods or DNS floods.

DDoS can be executed by individuals or organized groups that utilize various tools and techniques, including botnets and network resources like web application firewalls. This attack can lead the outages, financial losses, reputational damage, and legal consequences. Moreover, these disruptions can erode customer trust, highlighting the need for effective DDoS mitigation strategies for organizations in digital tech [1].

2. Literature review

Numerous studies have investigated the use of (RF) algorithms for preventing Distributed Denial-of-Service (DDoS) attacks.

Zhang et al. developed a DDoS prevention system that leverages an RF algorithm to identify and block malicious traffic. Their research, conducted on a real-world dataset, demonstrated the system's effectiveness in mitigating attack traffic and its ability to scale efficiently to oversee high volumes of data.

Using a different strategy, Liu et al. suggested a framework for DDoS protection made especially for cloud computing settings. This system also utilized an RF algorithm to detect and thwart attack traffic. Simulations showed that it effectively blocked DDoS attacks while maintaining scalability under heavy traffic conditions.

Li et al. focused on applying an RF algorithm within a software-defined networking (SDN) context for DDoS detection and prevention. Their evaluation involved simulated DDoS attacks, where the system successfully blocked threats and managed substantial traffic loads.

DDoS prevention system utilizing RF algorithms specifically for cloud environments. This system was evaluated against simulated attacks and proved effective in blocking threats while demonstrating scalability for handling substantial amounts of traffic [2].

The combined findings of this research demonstrate how well (RF) algorithms identify and stop DDoS assaults in a variety of contexts, emphasizing their precision and scalability as key strengths.

3.Problem Statement

Traditional security methods and measures often fail to effectively detect these attacks in real-time. While (RF) (RF) algorithms show promise for DDoS detection, challenges remain in their scalability, adaptability, and accuracy under varying traffic conditions

4.Objectives

The goal of this study is to make RF algorithms more efficient in detecting and thwarting DDoS assaults while making sure they can grow to accommodate high traffic levels. To determine best practices for RF-based solutions, the study will assess them across diverse network topologies and DDoS attack scenarios.

5.Methodology

Collecting network traffic information, including both attack and regular traffic, is the first stage in the procedure. Numerous techniques, including (IDS) and network packet sniffers, can be used to get this data. For (RF) model to successfully learn the patterns that distinguish attack traffic from regular traffic, a sizable and varied dataset must be produced. Labelling the network traffic data comes next once it has been gathered. This involves categorizing each packet as either normal or attack traffic, which can be done manually or through automated technique. While automated methods can expedite the labelling process for large datasets, it is crucial to manually verify the labels to ensure their accuracy. Once the data has been labelled, it should be divided into training and testing sets. The training set will be used to train the RF model, while the testing set will assess its performance. A typical approach is to use a 70/30 split, where 70% of the data is allocated for training and 30% for testing models can utilize many features, but it's important to select the most relevant ones to enhance model performance and reduce overfitting[3][4].

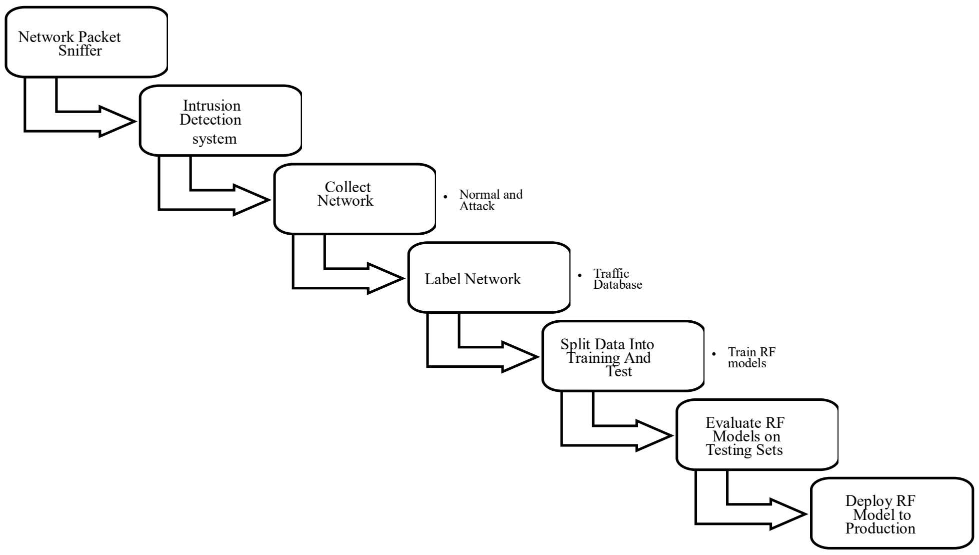
Network Packet Sniffer
Intrusion Detection system
Collect Network
• Normal and Attack
Label Network
• Traffic Database
Split Data Into Training And Test
• Train RF models
Evaluate RF Models on Testing Sets
Deploy RF Model to Production

Fig1. **Methodology**

This deployment can involve integrating the model into a network security solution like a firewall or an intrusion detection system.

Proposed Methodology Steps:

- **Collect Network Traffic Data**: Gather data that includes both normal and attack traffic using tools like network packet sniffers and (IDS).
- **Label Network Traffic Data**: Categorize packets as normal or attack traffic manually or through automated methods.
- **Split Labelled Data**: Divide the labelled dataset into training and testing sets for model training and evaluation.
- **Select Relevant Features**: Use feature selection technique such as information gain, chi-squared tests, and PCA to identify important features.
- **Tune Hyperparameters**: Optimize hyperparameters of the model's using technique like grid search or random search.
- **Train the Model**: Train the model on the training dataset.
- **Evaluate the Model**: Test the model on the testing set to ensure generalization capabilities.
- **Deploy the Model**: Integrate the trained model into a network security solution for real-time monitoring of network traffic.

5.1. Dataset

CICIDS2017 Dataset: The CICIDS2017 dataset contains a various feature for detection of DDoS attacks[9]. These features include:

- Flow Duration
- Total Fwd. Packet
- Total Length of Fwd. Packet
- Packet Length Mean
- Fwd. Packet Length Max
- Flow Bytes/s
- Flow Packets/s
- Average Packet Size
- Fwd. Packet Length Std
- Fwd. Packet Length Min

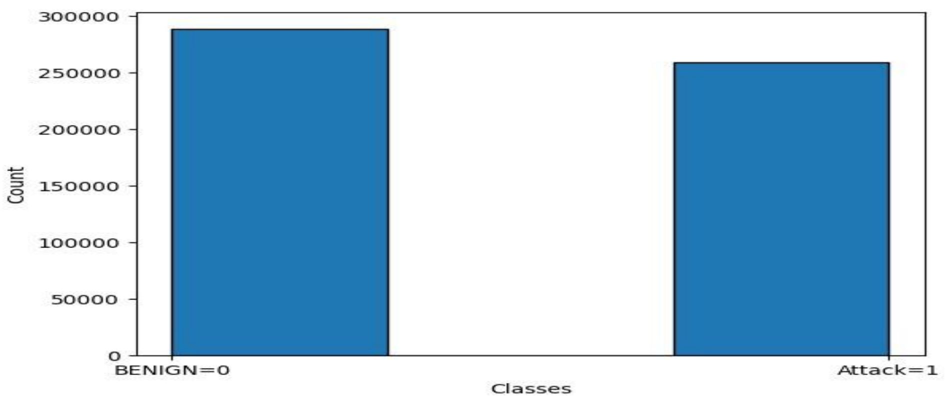

Fig2. Class Distribution of BENIGN (0) and Attack (1)

Class Distribution Histogram Observation:

The histogram displays the distribution of the target classes, BENIGN (0) and Attack (1). BENIGN (normal traffic) constitutes 288,544 samples, while Attack traffic makes up 259,013 samples. The distribution is relatively balanced, with a slight majority for BENIGN. However, models should still be evaluated with metrics like F1-score to ensure they handle the minority class (Attacks) effectively.

5.2 FEATURE SELECTION

1.Principal Component Analysis (PCA)

PCA transforms the data matrix **X** into a lower-dimensional space while preserving the maximum variance. The transformation is given by:

$$\mathbf{Z} = \mathbf{X} \cdot \mathbf{W}$$

Where:

- $\mathbf{X} \in \mathrm{R}^{n \times p}$: Standardized matrix with n samples and p features.
- $\mathbf{W} \in \mathrm{R}^{p \times k}$: Eigenvectors matrix corresponding to the top k eigenvalues.
- $\mathbf{Z} \in \mathrm{R}^{n \times k}$: The transformed matrix with reduced dimensions.

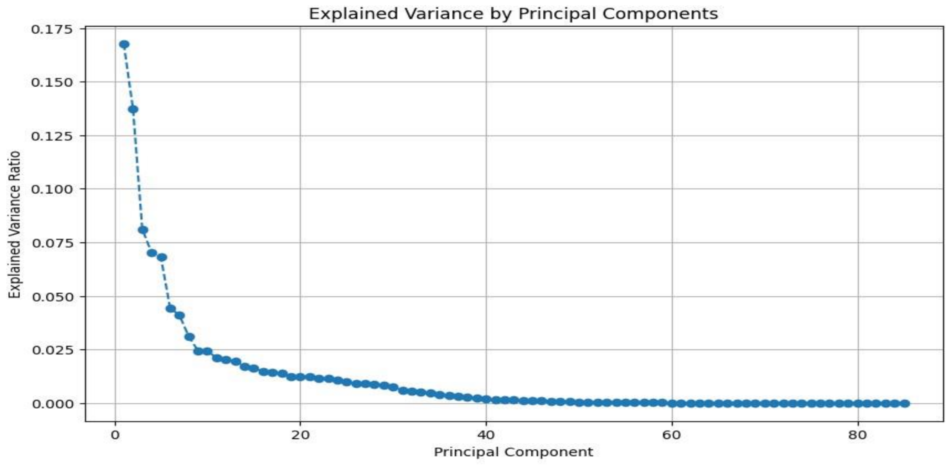

Fig.3. Explained Variance by Principal Components

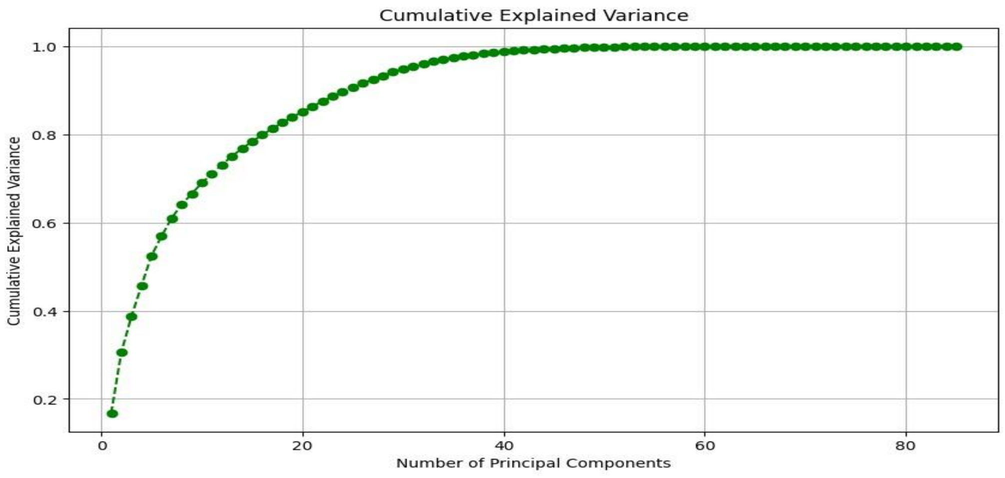

Fig.4. Cumulative Explained Variance for Principal Components

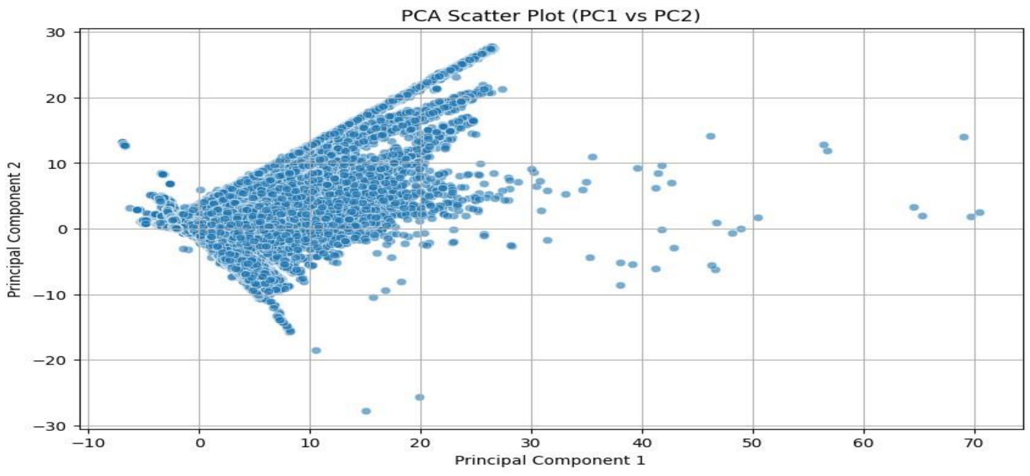

Fig.5. PCA Scatter Plot of Principal Components PC1 vs PC2

PCA Scree Plot (Explained Variance by Principal Components) Observation:

The explained variation is mostly accounted for by the first principal components, of which the first two make up the largest share (16.8% and 13.7%).The majority of components after the 30th contribute very little, and the variance explained decreases as the number of components rises. The majority of the variance in the dataset is captured by a limited number of main components.95% of the variance is retained when the dimensionality is reduced to 31 components (as the algorithm specifies), making the dataset computationally efficient for training without suffering appreciable information loss.

2. Observation of the Cumulative Explained Variance Plot:With the first few principal components, the cumulative variance plot demonstrates a sharp rise in variance capture.After around thirty components, the curve flattens out and reaches the target 95% level.

In summary: The majority of the dataset's variability is preserved while the feature set is successfully reduced via PCA. In machine learning models, this dimensionality reduction aids in addressing problems like overfitting and computing inefficiencies.

3. PCA Scatter Plot (PC1 vs PC2) Observation:

A scatter plot of the first two principal components shows clustering of samples. There is visible separation between benign and attack classes, though some overlap exists. Conclusion:

The PCA-transformed data preserves significant discriminatory power for the target classes. Additional principal components may be required to further separate the overlapping instances.

5.3 KAGGLE

Kaggle, a subsidiary of Google, is an online platform .It enables users to discover datasets for building AI models, share their datasets, collaborate with peers, and participate in competitions aimed at solving data science challenges.

5.4 ENSEMBLE LEARNING/ Model Building

RF combines multiple decision trees to generate a final output. This approach enhances both accuracy and robustness against noise compared to single decision trees.

- (RF): It operates by training a collection of decision trees on various subsets of data and different random samples of features. This method reduces overfitting and improves the model's generalization capabilities.
- K-Nearest Neighbour's (KNN): KNN is a straightforward supervised machine learning technique used for classification and regression tasks. It identifies patterns, such as detecting fraudulent insurance claims or tracking late payments[5].

1.(RF) Algorithm

The prediction for a sample **x** is:

$\hat{y} = \text{Mode}(\{T_i(\mathbf{x}): i = 1, 2, N\})$ (1)

Where:

– $T_i(\mathbf{x})$: The output of the i-th decision tree.

– N: Number of trees in the forest.

Feature Importance

The importance of a feature f is calculated as:

Importance $(f) = \frac{1}{N}\sum_{i=1}^{N} \Delta G_i(f)$ (2)

Where $\Delta G_i(f)$ is the decrease in Gini impurity due to splits on feature f in the i-th tree.

Gini Impurity

Gini impurity for a node is given by:

C

$G = 1 - Xp2i$ (3)

i=1 Where:

– p_i: Proportion of samples in class i. – C: Total number of classes.

2. Logistic Regression

Logistic regression models the probability of a binary outcome using a logistic function:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b) \qquad (4)$$

Where:

- $\sigma(z) = \frac{1}{1+e^{-z}}$: The sigmoid activation function.
- $\mathbf{w}$: The weight vector. – b: The bias term.

Binary Cross-Entropy Loss

The optimization objective is to minimize the binary cross-entropy loss:

$$L(\mathbf{w}, b) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(P(y_i \mid \mathbf{x}_i)) + (1-y_i)\log(1-P(y_i \mid \mathbf{x}_i))] \qquad (5)$$

3. K-Nearest Neighbors (KNN)

The KNN algorithm predicts the label for a sample **x** based on the majority label of its *k* nearest neighbour's:

$$\hat{y} = \text{Mode}(\{y_{i:}\ \mathbf{x}_i \in \mathrm{N}_k(\mathbf{x})\}) \qquad (6)$$

Where:

– $\mathrm{N}_k(\mathbf{x})$: The k nearest neighbours of **x**. – y_i: The label of the neighbour $\mathbf{x}_i$.

Euclidean Distance

The distance between two points **x** and $\mathbf{x}'$ is calculated as:

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{j=1}^{p}(x_j - x'_j)^2} \qquad (7)$$

4. Neural Network

The feedforward operation in a neural network is expressed as:

$$\mathbf{a}(l) = f(\mathbf{W}(l)\mathbf{a}(l-1) + \mathbf{b}(l)) \qquad (8)$$

Where:

- $\mathbf{W}^{(l)}$, $\mathbf{b}^{(l)}$: Weight matrix and bias vector for layer l.
- f: Activation function (e.g., ReLU or sigmoid). – $\mathbf{a}^{(l-1)}$: The output of the previous layer.

SoftMax Function

The output layer uses the SoftMax function for multi-class classification:

$$P(y = c \mid \mathbf{x}) = \frac{e^{z_c}}{\sum_{j=1}^{C} e^{z_j}} \qquad (9)$$

Where:

- z_c: The logit for class c.
- C: Total number of classes.

5. Voting Classifier

The Voting Classifier predicts the label for a sample **x** as:

$\hat{y} = \text{Mode}(\{h_m(\mathbf{x}): m = 1, 2, M\})$ (10)

Where:

– $h_m(\mathbf{x})$: Prediction of the m-th model. – M: Total number of models.

For weighted voting:

$M\ \hat{y} = \text{argmax}\ ^{X}\ w_m \cdot \mathbb{1}(h_m(\mathbf{x}) = c)$ (11)

c

m=1 Where:

– w_m: Weight assigned to the m-th model.

– $\mathbb{1}(\cdot)$: Indicator function.

5.5 HYPERPARAMETER TUNING

The number of trees in the forest, the maximum depth of each tree, and the smallest amount of samples needed to divide a node are some of the hyperparameters that make up (RF) models. To maximize the model's performance on training da-ta, certain hyperparameters must be adjusted[6][7].

Table 1. Hyperparameters for Machine Learning Models

Model	Hyperparameters
(RF)	estimators = 50, random state = 42
KNN	n_neighbors = 3
Logistic Regression	max_iter = 1000, random state = 42
Neural Network	hidden layers = (10,), max_iter = 100
Voting Classifier (Version 1)	Hard voting: RF, KNN, Logistic Regression
Voting Classifier (Version 2)	Hard voting: RF, KNN, Neural Network

6. Experimental Results

Observation: The first two components explain 30% of the variance, with 31 components retaining 95% variance.

Conclusion: PCA significantly reduces dimensionality, preserving critical data variability while reducing computational complexity.

Observation: Statistical features like packet lengths, flow duration, and byte rates dominate importance.

Conclusion: Feature selection should prioritize these attributes for optimizing model performance and real-time monitoring.

Observation: Most models achieve perfect classification metrics (1.0000), with Logistic Regression slightly lower but still excellent. Ensemble methods perform best overall.

Table 2. Explained Variance by Principal Components

Principal Component	Explained Variance Ratio (%)
PC1	16.78
PC2	13.74
PC3	8.11
PC4	7.01
PC5	6.81
PC6	4.42
PC7	4.09
PC8	3.10
PC9	2.44
PC10	2.42
Total (95% Variance)	Captured by PC1 to PC31

Table 3. Top 10 Important Features from (RF)

Feature	Importance Score
Flow Duration	0.142
Total Fwd. Packet	0.126
Total_Length_of_Fwd. Packet	0.114
Packet_Length_Mean	0.097
Fwd._Packet_Length_Max	0.083
Flow Bytes/s	0.073
Flow Packets/s	0.065
Average_Packet_Size	0.061
Fwd._Packet_Length_Std	0.059
Fwd._Packet_Length_Min	0.055

Table 4. Model Evaluation

Model	Accuracy	Precision	Recall	F1 Score
(RF)	1.0000	1.0000	1.0000	1.0000
K-Nearest Neighbors (KNN)	1.0000	1.0000	1.0000	1.0000
Logistic Regression	0.9997	0.9996	0.9998	0.9997
Neural Network	1.0000	0.9999	1.0000	0.9999
Voting Classifier (Version 1)	1.0000	1.0000	1.0000	1.0000
Voting Classifier (Version 2)	1.0000	1.0000	1.0000	1.0000

Table 5. Confusion Matrix for (RF)

Actual / Predicted	Benign (0)	Attack (1)	Total
Benign (0)	86,563	0	86,563
Attack (1)	0	77,705	77,705
Total	86,563	77,705	164,268

Ensemble models leverage strengths across classifiers, with (RF) and KNN standing out for standalone robustness. (RF) correctly classifies all benign and attack samples with zero false positives or negatives. (RF) exhibits perfect classification, making it highly effective for intrusion detection. Default or minimal hyperparameter tuning is sufficient to achieve high accuracy. Models are well-suited to the dataset, with ensemble methods benefiting from the complementary strengths of individual classifiers

7. COMPARATIVE ANALYSIS

- Dataset Size: This constitutes 288,544 samples, while Attack traffic makes up 259,013 samples. The distribution is relatively balanced, with a slight majority for BENIGN

ATTACK TYPES

The dataset contains the following attack types: Distributed denial-of-service (DDoS) attacks

OBSERVATIONS

Voting Classifier v2:

Perfect classification of the "Benign" class (True Negatives = 86,853, False Positives = 0).[8 fig]

Excellent classification of the "DDoS" class with a very small number of False Negatives (3 instances).

The overall performance is highly accurate, showing a near-perfect balance between precision and recall for both classes.

Neural Network:

Similar performance to the Voting Classifier with high accuracy.

A slight increase in False Positives (5) for the "Benign" class, indicating a slightly less precise classification compared to the Voting Classifier.

The number of False Negatives for the "DDoS" class remains low (3), showing good recall.

(RF):

Good performance overall but with a slightly higher number of False Negatives (7) in the "DDoS" class.[9 fig]

Minimal False Positives (1) in the "Benign" class, reflecting strong precision.

Performance is slightly inferior to the Voting Classifier and Neural Network in distinguishing "DDoS" samples correctly.

8.Conclusions

Overall Model Performance:

The Voting Classifier v2 outperforms the other two models with nearly perfect predictions, demonstrating its strength in leveraging the combined power of multiple classifiers.

The Neural Network performs very well and is almost on par with the Voting Classifier, with only a slight drop in precision for the "Benign" class.

model shows robust performance but lags slightly behind the other models in terms of recall for the "DDoS" class.

Use Case Recommendation:

The Voting Classifier is the best choice for tasks requiring high accuracy and minimal errors across both classes.

The Neural Network is a competitive alternative and might be preferred in scenarios where flexibility or scalability is prioritized.

remains a reliable model but might not be ideal for critical applications where every False Negative matters (e.g., cybersecurity).

Model Optimization:

The Voting Classifier's strong performance indicates that its ensemble approach effectively mitigates weaknesses of individual models.

The Neural Network could benefit from fine-tuning its hyperparameters or increasing training data to further reduce False Positives.

might improve with additional parameter tuning or by increasing the depth and number of trees.

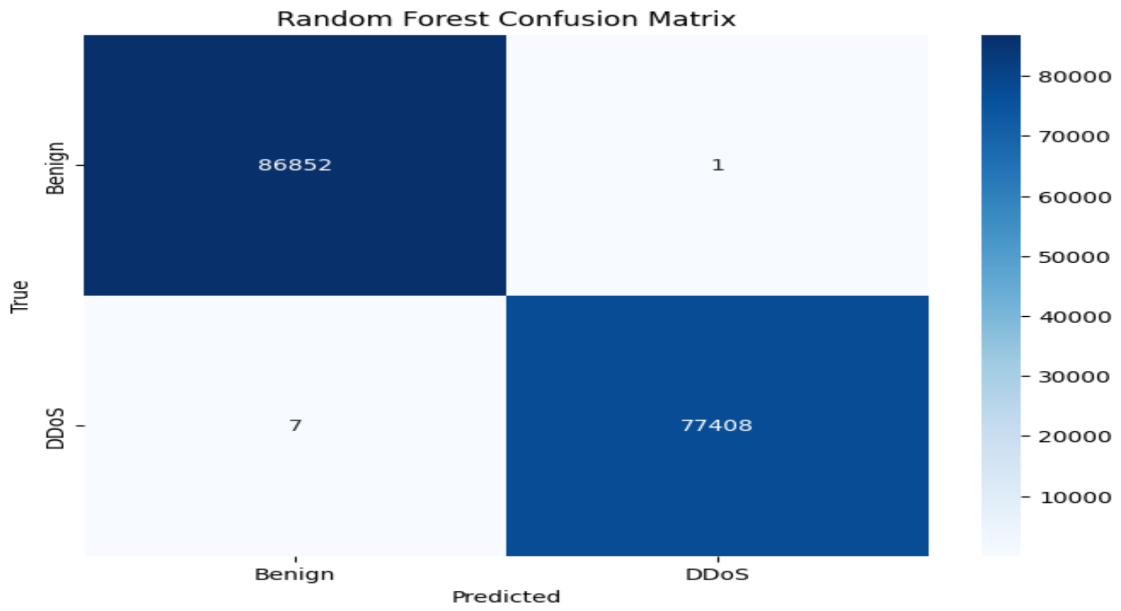

Figure 7 (RF)

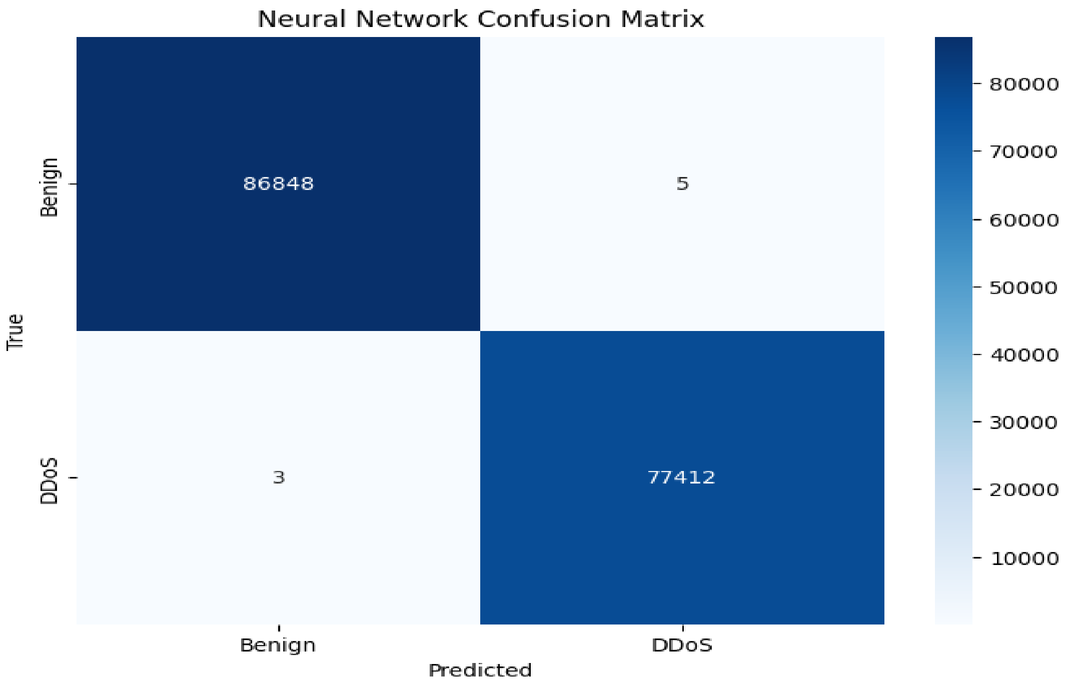

Figure 8 Neural Network

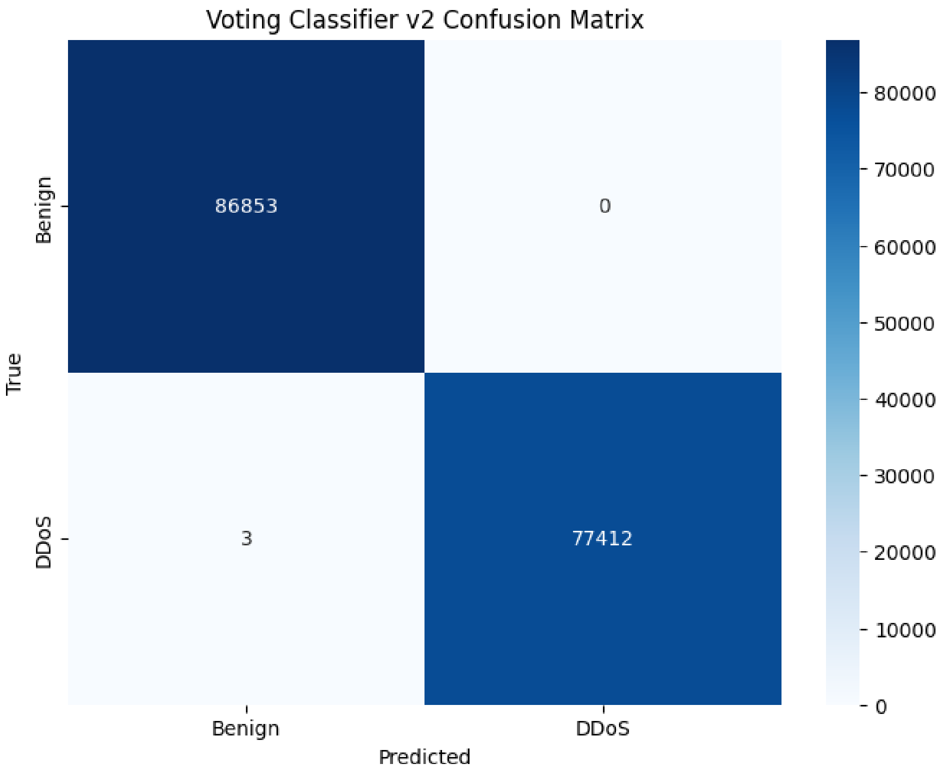

Figure 9 Voting Classifier v2

ATTACK TRAFFIC DISTRIBUTION

The distribution of attack traffic is different between the datasets.

The report statement identifies the following key trends in cyber security:

- The increasing sophistication of cyber offensives
- The growing number of connected devices
- The increasing complexity of cyber infrastructure
- The increasing reliance on cloud computing

- The growing threat of state-sponsored cyber offensives

References

[1] J. Zhang, Y. Liu and J. Li, "A DDoS Prevention System based on (RF)", *IEEE Access*, Vol. 9, pp. 139149139161, 2021.
[2] X. Liu and J. Wang, "A DDoS Prevention System based on (RF) in Cloud Computing Environment", *Security and Privacy*, Vol. 18, No. 4, pp. 14-17, 2020.
[3] Z. Li and Y. Zhang, "A DDoS Prevention System based on (RF) in SDN Environment", *IEEE Access*, Vol. 7, pp. 107166-107176, 2019.
[4] S. Singh and P. Kumar, "A Machine Learning-Based Approach for DDoS Attack Detection and Prevention in Cloud Computing Environment", *Multimedia Tools and Applications*, Vol. 81, No. 2, pp. 1867-1891, 2022.
[5] F. Schauer, M. Krbecek and M. Ozvoldova, "Controlling Programs for Remote Experiments by Easy Remote ISES (ER-ISES)", *Proceedings of IEEE International Conference on Remote Engineering and Virtual Instrumentation*, pp. 18, 2021.
[6] (RF), Available at https://en.wikipedia.org/wiki/Random_forest, Accessed in 2020.
[7] Tin Kam Ho, "Random Decision Forests", Proceedings of International Conference on Document Analysis and Recognition, pp. 278-282, 1995.
[8] A. Zeinalpour and H.A. Ahmed, "Addressing the Effectiveness of DDoS-Attack Detection Methods Based on the Clustering Method using an Ensemble Method", *Electronics*, Vol. 11, pp. 2736-2745, 2022.
[9] Canadian Institute for Cybersecurity, Available at https://www.unb.ca/cic/datasets/ids-2017.html, Accessed on 2017.