Applications of Machine Learning to DDoS Attack Detection and Prevention

1st Laela Olsen (1220948755)
School of Computing and Augmented Intelligence
Arizona State University
Tempe, United States
lolsen7@asu.edu

2nd Anchala Balaraj (1233434009)
School of Computing and Augmented Intelligence
Arizona State University
Tempe, United States
abalaraj@asu.edu

3rd Jin Heo (1212846750)
School of Computing and Augmented Intelligence
Arizona State University
Tempe, United States
jhuh3@asu.edu

4th Ahmad Karimi (1233868989)
School of Computing and Augmented Intelligence
Arizona State University
Tempe, United States
aikarimi@asu.edu

5th Yesha Modi (1232679307)
School of Computing and Augmented Intelligence
Arizona State University
Tempe, United States
ymodi6@asu.edu

6th Vignesh Mohana Velu (1233212372)
School of Computing and Augmented Intelligence
Arizona State University
Tempe, United States
vmohanav@asu.edu

7th Nikhil Swami (1233379331)
School of Computing and Augmented Intelligence
Arizona State University
Tempe, United States
nswami1@asu.edu

8th Advaith Venkatsubramanian (1233367072)
School of Computing and Augmented Intelligence
Arizona State University
Tempe, United States
avenk138@asu.edu

***Abstract*—This project aims to examine the efficacy of various machine learning techniques in identifying DDoS attacks. Techniques examined include Naive Bayes, Support Vector Machine, Random Forest, Decision Tree, Logistic Regression, Artificial and Convolution Neural Networks, and K-nearest Neighbors. These techniques have been evaluated on a portion of the CICIDS2017 dataset, which contains realistic simulated network traffic containing Distributed Denial-Of-Service attacks, by measuring the performance of different machine learning techniques on detecting these attacks. Also discussed is how machine learning techniques can be implemented into supply chains to prevent the potential damage that can be inflicted by cyberattacks. The findings of our team were that any machine learning technique can be used to effectively identify DDoS attacks within network traffic, as all models were able to achieve accuracies over 80%. The Decision Tree and Random Forest models achieved the highest accuracies at 99.98%, while the Naive Bayes model had the lowest accuracy at 86.11%.**

***Index Terms*—Network Intrusion Detection, Machine Learning, DDoS Attacks, Network Traffic Analysis, Cybersecurity**

I. Introduction

Distributed Denial of Service attacks or DDoS attacks pose a substantial threat to the integrity and security of online services and networks. These kinds of attacks keep evolving over time and have become even more sophisticated. Research in the techniques of leveraging Artificial Intelligence (AI) and Machine Learning (ML) have rapidly picked up and have proven to be very promising thus far. DDoS attacks can result in significant financial losses, reduction in productivity, and sometimes irreparable damage to reputation.

The main objectives of this research domain include to implement and modify existing Machine Learning and Artificial Intelligence algorithms for DDoS threat detection, to discuss potential response mechanisms which could mitigate or compensate for DDoS attacks with minimal loss, to compare the efficacy of different algorithms across different scenarios and to gauge how well unsupervised learning can perform against previously unknown attack patterns.

Using our research findings, we aim to address and answer questions regarding the accuracy of using ML and AI algorithms for threat detection and the speed of the detection process. Additionally, we look at various aspects like the data, the relevant features which can assist in building a predictive model, the performance trade-offs and computational resources required by each of the various techniques,

Research and development in this domain can greatly contribute to the field of cybersecurity by providing novel solutions to fight the danger of DDoS attacks in this modern world.

II. Literature Review

The authors [4] propose a framework for detecting DDoS attacks that leverages the NSL-KDD and KDD Cup 99 datasets, known benchmarks in intrusion detection research, which contain 41 distinct network traffic features. Their approach begins with data pre-processing, including data refinement and cleaning, such as removing missing values, followed by feature selection using correlation analysis. This analysis reduces the feature set to eight highly correlated attributes relevant to DDoS detection, which are then used to train machine learning models, specifically Naïve Bayes and K-Nearest Neighbor (KNN). The model training process distinguishes between "normal" network behavior, which includes every day user activities like file transfers and email exchanges, and "attack" behavior indicative of a DDoS attack. The dataset is split 70:30 for training and testing, respectively. Performance metrics, including accuracy, F-measure, recall, and precision, were employed to evaluate model effectiveness, with KNN achieving a higher true positive rate and Receiver Operating Characteristic (ROC) score (0.99) compared to Naïve Bayes (ROC score 0.97). This framework demonstrates the importance of feature selection in reducing noise and optimizing model accuracy in DDoS attack detection.

In the work "Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning" [5], the authors investigate several machine learning techniques, finding that the most accurate to the least accurate are random forest, decision tree, perceptron, standard gradient descent, linear regression, and AdaBoost. Although random forest was the most accurate, the finding was that all techniques were able to achieve an accuracy of over 95% on preprocessed data, with each technique besides AdaBoost achieving an accuracy of over 98%. This research used the signatures extracted from network traffic as features for analysis.

In "Web Server Protection against Application Layer DDoS Attacks Using Machine Learning and Traffic Authentication" [6], researchers investigate the uses of machine learning and traffic authentication in mitigating application layer DDoS attacks. The machine learning technique chosen for this research was a random tree classifier. This random tree classifier was used in conjunction with existing intrusion detection and prevention systems (IDS/IPS) to both prevent more malicious traffic and to allow false positives to be forwarded to their intended destinations. The study found that this technique along with the use of bait servers was able to speed up server response times during an attack when a network intrusion protection system (NIPS) is enabled.

The paper titled "Detection of DDoS Attack on Smart Home Infrastructure Using Artificial Intelligence Models" is primarily focused on comparing the K-Nearest neighbors algorithm with the Artificial Neural Network algorithm in detecting DDoS attacks on smart home infrastructure appliances [8]. The researchers have followed the cross industry standard process for Data Mining (CRISP-DM) Methodology for data mining and data analysis. This methodology follows a six step cycle to evaluate model performance. The paper concluded that the KNN model is a more suitable model for detecting DDoS attacks in smart homes.

The paper "A Machine Learning-Based Classification and Prediction Technique for DDoS Attacks" [2] focuses on the need for a strong defense mechanism for the identification of DDoS attacks and working against them which can be possible by implementing Machine Learning models that can classify and predict those attacks along with providing high accuracy. This research made use of XGBoost and Random Forest Classifier to differentiate between normal traffic patterns and DDoS patterns on the basis of features like protocol type and traffic volume. It was seen that using these two classifiers, the model provided good accuracy which outperformed the previous methods of two classifiers: convolutional neural network and recurrent neural network used by the authors of a different paper. The model demonstrates particular effectiveness in handling complex DDoS patterns.

This research "Identifying DDoS Attack using Split-Machine Learning System in 5G and Beyond Networks" tackles the issue of safeguarding 5G and future networks against Distributed Denial-of-Service (DDoS) attacks by introducing a Split-Machine Learning (Split-ML) methodology [9]. The Split-ML framework integrates machine learning with human expertise to enhance detection accuracy and response times. By employing a Classification and Regression Tree (CART) model, it detects DDoS patterns based on characteristics such as packet timing and protocol type, effectively differentiating between malicious and legitimate traffic. The model exhibited high accuracy across various protocols, showing particular strength in countering TCP and HTTP-based attacks.

The aim of the paper "Detection and Mitigation of DDOS-based Attacks using Machine Learning Algorithm" is to examine the notable danger presented by Distributed Denial of Service (DDoS) attacks and to investigate sophisticated methods for detection and prevention, particularly in Software-Defined Networks (SDNs) [10]. The research offers an in-depth examination of machine learning methods, such as supervised, unsupervised, and hybrid techniques, for identifying DDoS attacks, with an emphasis on achieving high accuracy in detection while keeping false alarms to a minimum. It suggests a new mixed defense method, blending statistical entropy detection with machine learning tactics, to improve safety in SDN settings. Another paper titled "Enhancing Resilience against DDoS Attacks in SDN-based Supply Chain Networks Using Machine Learning" [11] is also based on data collected from SDNs.

Interestingly, the paper proposed by [1] concludes that KNN outperforms Logistic Regression, CNN, XGBoost [2], and AdaBoostClassifier when applied to a small-scale smart home dataset, providing superior detection accuracy. It also infers that Convolutional Neural Networks (CNNs) and Deep Learning models [13], though computationally intensive, have shown remarkable success in recognizing complex traffic patterns in high-dimensional data, making them suitable for large-scale DDoS detection in complex network environments.

The paper "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," [12] discusses the creation of the CICIDS2017 dataset, which was created to meet real-world criteria and to be available to the public. This is the dataset that was used to test all eight of the models built by our team. The paper compares the newly created dataset against previous cyberattack datasets and finds that their dataset allows machine learning techniques to perform well and meets a number of criteria that no other previously accessible dataset meets all of, those being complete network configuration, complete traffic, the presence of a labeled dataset, complete interaction and capture, having all common protocols available, and more. Additionally, the paper discusses all of the types of cyberattacks present in the dataset, including DoS and DDoS attacks. All of these above-listed factors influenced the choice that our team made to use this dataset rather than another cyberattack dataset to test the models that our team built.

The uniqueness of this current research comes from the sheer quantity of models that our team is testing, in combination with our use of the well-regarded CICIDS2017 dataset. Our team chose to build and test a total of eight different machine learning techniques, more than any of the papers cited above, on the same segments of the CICIDS2017 dataset, those being the segments which contain the DoS and DDoS attacks and the benign traffic, in order to be able to directly compare the metrics garnered from each of the models. Additionally, our research is unique in that our research discusses how the tested techniques could be implemented into real life supply chains in order to reduce the damages caused by cyberattacks such as DDoS attacks in real time.

The paper "Defeating Denial-of-Service Attacks in a Self-Managing N-Variant System" [3] discusses the challenges of securing N-variant systems against DoS attacks, which is a common vulnerability in systems designed to self-heal by restarting or even rolling back after an anomalous behavior is detected. N-variant systems typically operate by running multiple variants of a single program in parallel to ensure that benign inputs yield identical behaviors across these variants, where malicious inputs cause divergences. These systems are however prone to such attacks when the attacker forces the system into repeated divergence cycles with sheer volume, preventing proper recovery.

The paper introduces an automated defense mechanism named 'Crispy', which takes inspiration from CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) and how the adaptive immune system responds to bacterial CRISPR/Cas system. Crispy, much like the biological model does with genetic signatures of past infections to recognize future pathogens, detects divergence-inducing inputs and adds them to a blacklist to filter repeated malicious inputs out. This allows the system to recover and continue to operate without being disrupted by recurring attacks. Crispy has proven to effectively mitigate up to 87.5% of DoS attacks without false positives in a controlled environment of two web servers. As a self-managing and automated system, Crispy offers a promising new pathway for defending critical infrastructure from these attacks and suggests how it may be implemented alongside machine learning models to create a hybrid model to more effectively defend against them.

III. Methodology

The paper focuses on predicting DDoS attacks using various machine-learning techniques. Given that the current dataset's labels' instances take limited values, it would be an excellent approach to implement classification ML algorithms. However, there is a vast array of available classifier models, and the paper aimed to consider a few like Naive Bayes, K-Nearest Neighbors (KNN), Decision Tree Classification, Random Forest, Support Vector Machine (SVM), and Logistic Regression. Additionally, the paper considers linear regression for its simplicity and ultimately employs an Artificial Neural Network (ANN) model. The full code of each model created for this research can be found in the Google Drive folder linked in the references [7].

The CICIDS2017 is a complete and more current intrusion detection dataset developed to minimize the shortcomings of previously developed datasets in this area. According to the researchers at the Canadian Institute for Cybersecurity, it features both benign and malicious network traffic that closely mimics real-world data. Generated in five days, the first day had normal activity; other days had different cyberattacks, including Brute Force, DoS, DDoS, Web attacks, and Infiltration. It contains the complete network configuration, different traffic patterns, and many protocols. Every record is labeled, and more than 80 features of network flow are extracted by CICFlowMeter. It contains raw packet captures in PCAP format along with the processed CSV format so that it can be directly used for ML and deep learning applications. The CICIDS2017 dataset is generated with real background traffic using the B-Profile system, which emulates the human activities making it more natural and hence complex. This makes it an invaluable source for scholars and practitioners in the field of network security and intrusion detection.

The methodology flow diagram [Fig. 1] explains how the ML algorithms were implemented on the CICIDS2017 dataset. The CICIDS2017 dataset was selected over NSL-KDD, UNSW-NB15, and CTU-13 due to its inclusion of diverse, modern network threats and detailed flow-based features essential for effective DDoS detection. Its realistic traffic simulation and comprehensive labeling make it well-suited to building robust ML models for this purpose, aligning closely with the study's objectives. Firstly, preprocessing is done on the dataset. A general preprocessing is done for all ML models. However, some ML models may require different preprocessing to increase the accuracy and precision of the prediction set generated in the end. The CICIDS2017 dataset was uploaded, and only files containing DDoS attacks were selected. Getting the preprocessing right is the most crucial stage as the training of ML models depends entirely on the preprocessed dataset. The selected files were merged so entries

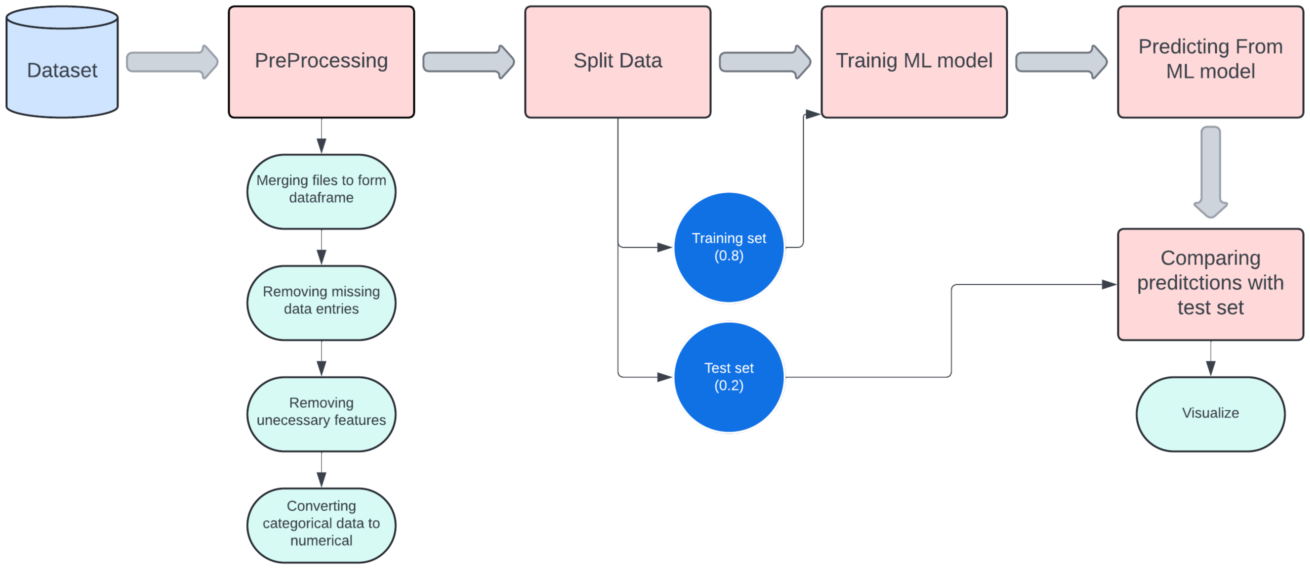
Dataset
PreProcessing
Split Data
Trainig ML model
Predicting From ML model
Merging files to form dataframe
Removing missing data entries
Removing unecessary features
Converting categorical data to numerical
Training set (0.8)
Test set (0.2)
Comparing preditctions with test set
Visualize

Fig. 1. Methodology Flow Diagram: Implementation of Machine Learning Algorithms

could be consistent, creating a data frame. This data frame had many empty entries and values; handling these cases are a must to avoid inconsistent training of the ML models. Removing these entries is the most suitable approach. However, the data frame still had many unwanted values that did not contribute to identifying whether the attack was DDoS or not. After carefully observing the data, the 24 most relevant features were identified and used for further processing. Furthermore, the dataset had many categorical data that needed to be converted into numerical data. For example, the feature 'Label' can take seven values; thus, values from 0 to 6 were assigned to each value as this 'BENIGN': 0, 'DDoS': 1, 'DoS GoldenEye': 2, 'DoS Hulk': 3, 'DoS Slowhttptest': 4, 'DoS slowloris': 5, 'Heartbleed': 6. This was the final step of preprocessing, although some algorithms may demand further processing on the dataset to be used for training.

The second major step is to split the data. Later, the training and test sets were divided into 0.8 and 0.2 parts of the entire dataset. The training set is kept large for better performance of the ML model. More training will improve the model's accuracy and enable it to predict values more precisely. The training set trains the ML model whereas the test set is kept for comparison. After the model is generated and trained, the values of the test set are injected into the model so that the ML model can make predictions. The test set is used to evaluate the model's performance against the predicted values. Finally, visualization is done to summarize the model's performance. For example, creating a confusion matrix, learning curve, etc.

A. *Logistic Regression*

Logistic regression (LR) was selected to be the baseline model due to its simplicity relative to other models used in this study. LR assumes a linear relationship between the input features and the log-odds of variables, and has the advantage of being more computationally efficient and thereby more suitable for datasets with limited feature complexity [Fig. 2]. It is often selected to quickly evaluate the predictive power of basic features before advancements to more complex models.

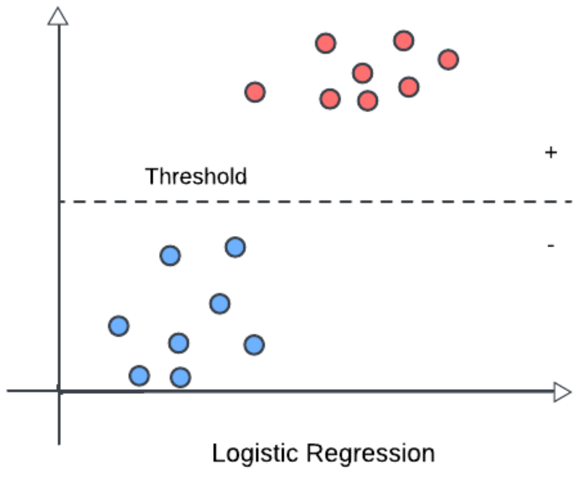

Fig. 2. Logistic Regression

Surprisingly, LR has proven to be quite effective at identifying benign traffic and DDoS attacks with an accuracy and F1 score of over 99.88%, suggesting that it may actually prove viable in application, as it is often less computationally intensive and the trade-off of accuracy (in this instance) is low.

Standardized transformation was applied to the feature set, and the dataset was split into training (80%) and testing (20%) sets to ensure sufficient data was allocated for validation. While this specific dataset has proven high levels of accuracy, it is important to note that DDoS attacks and malicious traffic are evolving over time and may become more complex and harder to detect with less sophisticated models as LR is limited in capturing non-linear relationships.

B. *Naive Bayes*

Naive Bayes is a probabilistic classification model. This classification algorithm uses the Bayes theorem, assuming that strong independence exists amongst features. Naive Bayes classifier is used mainly for sentiment analysis, spam filtering (generally in emails), and recommendation algorithms. The model is easy to use because of its simplicity and speed of implementation. As mentioned, NB is relevant for spam filtering, which aligns well with the problem addressed in this paper, making it a suitable choice for the dataset provided. Since the dataset comprises 79 features, which can work in favor or not depending upon the level of dependency among them, Naive Bayes has a prerequisite that all features are independent (hence the term 'Naive'), the model may yield low prediction accuracy if dependency exists amongst features. Although there are many Naive Bayes classifiers as well such as Gaussian Naive Bayes and Bernoulli Naive Bayes, Gaussian Naive Bayes is used for the purpose of this study. The way this algorithm was implemented is that the preprocessing focused on filtering the files that were related to DDoS attacks which were *Wednesday-workingHours.pcap_ISCX.csv* and *Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv*.

General preprocessing techniques were applied to the data set to filter the data so the model could be trained and give accurate results. The two files were combined together, forming a data frame, followed by removing empty data entries, discarding unnecessary features, and converting categorical data values into numerical values. Furthermore, the dataset is divided into training and test sets with the test size equal to 0.2 fraction of the entire dataset and the rest being the training set. After that, the Naive Bayes model was trained on the training dataset. The predictions were made, and the test set was compared with the prediction results from the model to test for accuracy.

C. *Artificial Neural Networks (ANN)*

Artificial Neural Networks (ANNs) are models consisting of layers of interconnected nodes or neurons, where each neuron processes inputs by applying a weighted sum followed by an activation function. The network "learns" by adjusting these weights in response to errors in predictions, enabling it to detect intricate relationships within large datasets. In binary classification tasks, such as DDoS attack detection, the ANN is particularly useful due to its ability to capture nonlinear patterns that distinguish benign from malicious traffic.

For this study, we applied an ANN model to a subset of the CICIDS2017 dataset, reducing it to 30% of the original size and selecting the 24 most relevant features from an initial set. After handling missing values and replacing infinite values, the dataset was split into features (X) and labels (y), with 'Label' as the binary target column (where "0" represents benign traffic and "1" represents DDoS attacks). Stratified sampling was employed to divide the dataset into training (80%) and testing (20%) sets, preserving the class distribution across both sets. Feature scaling, performed using standardization, enhanced learning efficiency—a critical step given the sensitivity of ANNs to input variations.

Our ANN model architecture included two hidden layers in addition to the input and output layers. The input layer comprised 64 neurons with ReLU activation, followed by a second hidden layer of 32 neurons, also using ReLU activation. A sigmoid function in the output layer facilitated binary classification. We compiled the model with binary cross-entropy as the loss function and employed the Adam optimizer, monitoring validation loss with early stopping to prevent overfitting. The model was trained with a 10% validation split, up to a maximum of 20 epochs, and a batch size of 64.

D. *Convolutional Neural Network (CNN)*

The convolutional neural network built by our team, like the other models, uses the Wednesday and Friday Afternoon data from the CICIDS2017 dataset, as these are the portions of the dataset which contain DDoS attack network traffic, along with benign network traffic. The CNN is built upon a TensorFlow Keras sequential model. After the layers of the sequential models, a convolutional layer is added, the data is flattened, and then a ReLU activation layer, a dropout layer, and a softmax layer are added. Six output classes are used, to differentiate between benign traffic and the five different types of DDoS attacks found in the network traffic. The model is trained over ten epochs with early stopping on eighty percent of the data, making up the training set, and tested on the remaining twenty percent of the data. Finally, the metrics are outputted and a confusion matrix is built.

E. *Decision Tree*

In this implementation, the Decision Tree classifier is used to analyze network traffic data to detect DDoS attacks. The dataset consists of multiple network features from CICIDS2017 datasets where implementation was focused on Wednesday and Friday Afternoon data and this subset includes both Benign and DDoS attack traffic.

Feature Label-DDoS was taken as a target, which identified if each instance of traffic was Benign or associated with the DDoS attack. Further, the dataset was divided into 80% training data and 20% testing data. Accuracy, precision, recall, and F-1 score were estimated after training the data, which gave insight into the predictive capability of the model.

Finally, a confusion matrix was created that presented the number of true positives, true negatives, false positives, and false negatives within the classification results which shows the capability of the model in distinguishing between Benign and DDoS traffic instances.

F. *Random Forest*

Random Forest is an ensemble learning algorithm which means it uses multiple models to boost its performance. It uses multiple decision trees to reach its final decision. A decision tree is a tree-like model that makes predictions by following a series of decisions based on the input features. Each internal node in the tree represents a decision or a test on an attribute, and the leaf nodes represent the final decision. When building each decision tree, Random Forest randomly selects a subset of the available features to consider at each split point. This process of random feature selection helps to introduce diversity among the decision trees and reduces the correlation between them [Fig. 3]. It leverages Bootstrap Aggregating or bagging for the sampling process. To create each decision tree in the Random Forest, the algorithm uses a technique called bootstrap sampling, which involves randomly selecting a subset of the training data with replacement. This algorithm aggregates predictions by following a system of majority voting. For each bootstrap sample, a decision tree is grown to its maximum depth without any pruning. The decision trees are grown using the selected subset of features, which helps to create diverse and unbiased trees. When making a prediction on a new data point, each decision tree in the Random Forest makes a prediction. For classification tasks, the final prediction is the majority vote among the individual tree predictions, whereas, for regression tasks, the final prediction is the mean of the predictions of each individual decision tree. The Random Forest algorithm is robust to outliers and is less prone to overfitting since it is an ensemble-based learning algorithm. It is also able to handle large feature spaces efficiently.

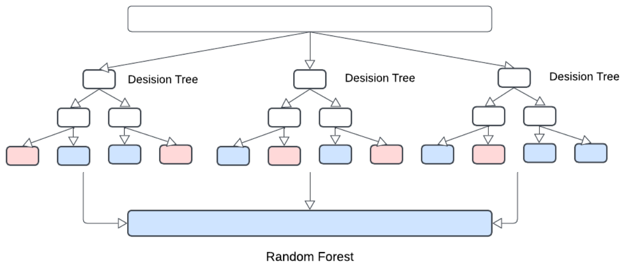

Fig. 3. Random Forest

The team applied this algorithm to a sample of data from the CICIDS2017 dataset which had traces of DDoS attacks. In the data preprocessing step, label encoding was done on the target variable where different types of attacks were grouped into a single category which represented a DDoS attack. Unnecessary columns were dropped from the data. The model was trained by making use of 100 decision trees and the criterion for calculating the quality of the split was the Gini Impurity index. The data was split into two parts where 80% of the data was used for training and the remaining 20% was used for testing.

G. *Support Vector Machines*

Support Vector Machines (SVM) are powerful supervised learning algorithms that excel in binary classification problems. They function by creating a hyperplane or multiple hyperplanes in a high-dimensional space to effectively distinguish between different classes [Fig. 4]. Additionally, SVM is capable of managing non-linearly separable data by transforming it into higher-dimensional feature spaces, where linear separation becomes possible. This adaptability makes SVM a valuable asset for tackling intricate classification challenges.

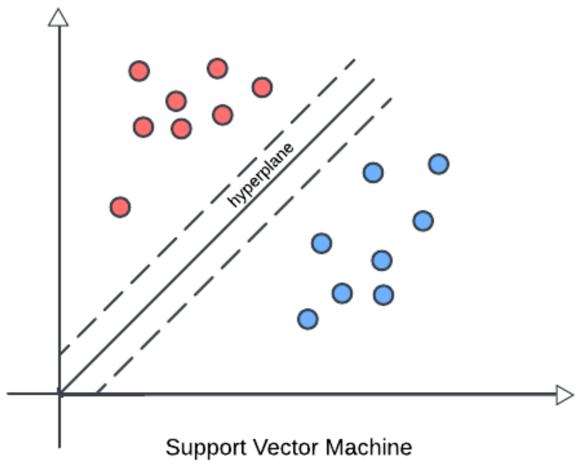

Fig. 4. Support Vector Machines

The initial phase involved importing the dataset and conducting a preliminary examination to ascertain its structural characteristics. Initial data assessments identified the presence of missing values. Furthermore, categorical variables, especially those found in the 'Protocol' column, were transformed into numerical format through the application of LabelEncoder, thereby ensuring compatibility with the SVM algorithm.

The target column, designated as Label, was extracted to indicate whether a data packet signified an attack. The other columns constituted the feature set (X). The dataset was divided into training and testing subsets with an 80-20 ratio, ensuring a balanced methodology that facilitated both the training of the model and the evaluation on previously unseen data.

Feature scaling was implemented using StandardScaler, which adjusted each feature to possess a mean of zero and a variance of one. This step is essential for Support Vector Machine (SVM) models, as they depend on distance-based computations and are enhanced by features that are uniformly scaled. Subsequently, a linear SVM model was developed using the scaled training data.

H. *k-Nearest Neighbors (kNN)*

The k-nearest neighbors model built by our team is built upon the KNeighborsClassifier module for the scikit-learn Python library. The data is preprocessed using the same methods as for the other machine learning models [Fig. 5]. A range of values for k is tested to determine which value of k would yield the highest accuracy. Using that value of k, the k-nearest neighbors classifier is trained on the training data set and then tested on the remainder of the data. Finally, the metrics of the classification are reported and a confusion matrix is generated.

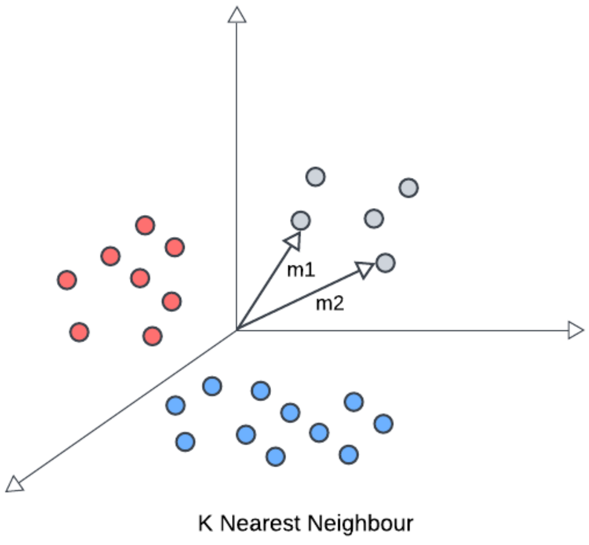

Fig. 5. K-Nearest Neighbors (kNN)

IV. Results

A. *Overview*

To evaluate and compare the effectiveness of various machine learning models in detecting DDoS attacks, we measured their accuracy, precision, recall, F1 score, and processing time on the CICIDS2017 dataset. The table below [Table I] summarizes these metrics for each model in said order.

TABLE I
Performance Metrics for DDoS Detection Models

Model	Acc (%)	Prec (%)	Rec (%)	F1 (%)	Time (s)
LR	99.86	99.87	99.88	99.87	17.919
NB	86.11	85.00	85.00	85.00	0.400
ANN	98.35	98.00	98.00	98.00	100.03
CNN	99.67	99.66	99.67	99.66	218.35
DT	99.98	100.0	100.0	100.0	7.801
RF	99.98	99.00	99.00	99.00	210.48
SVM	99.10	100.0	100.0	100.0	98.670
kNN	99.10	99.10	99.10	99.10	147.89

The Decision Tree (DT) and Random Forest (RF) models achieved the highest accuracy, while the Naive Bayes (NB) model displayed lower performance, potentially due to its assumption of feature independence. Because real-time performance directly impacts both the speed and accuracy of DDoS detection, as well as the volume of traffic it can handle, computational time is critical in assessing each model's overall effectiveness. To gauge this 'true' effectiveness, we introduce a new metric that balances model performance (in terms of accuracy, precision, recall, and F1 score) against computational time. This metric, named the 'time-weighted performance score', should provide a more practical perspective on each model's viability in real-world scenarios.

TABLE II
Time-weighted performance scores for each model

Model	Perf. Score	Proc. Time	P. Score (Weighted)
LR	0.9975	17.919s	0.0556
NB	0.8528	0.400s	0.6095
ANN	0.9800	100.03s	0.0098
CNN	0.9965	218.35s	0.0046
DT	1.0000	7.801s	0.1137
RF	1.0000	210.48s	0.0047
SVM	0.9980	98.670s	0.0100
kNN	0.9910	147.89s	0.0067

The time-weighted performance score is computed using the formula

$$\text{Weighted Score} = \frac{\text{Performance Score}}{1 + \text{Processing Time (s)}} \tag{1}$$

where Performance Score is an average of all four performance metrics (accuracy, precision, recall, and F1 score). Higher performance score indicates better accuracy, precision, recall, and F1 metrics in relation to computational time. A lower score does not necessarily indicate poor performance; if resources are sufficient and accuracy is prioritized over computational time, this metric can be modified to favor accuracy further. This metric is highly context-dependent, as the model's effectiveness will vary significantly with factors like traffic volume and machine capabilities. In high-traffic environments with sufficient computational resources, accuracy gains may outweigh moderate increases in processing time, and vice versa.

B. *Results for Logistic Regression*

After thoroughly processing the dataset, the model achieved a high accuracy of 99.86% and an F1 score of 99.87%, indicating a good predictive accuracy and balanced performance between precision and recall. Examining the ROC curve [Fig. 16] shows that the model effectively differentiates between positive and negative classes, making it highly suitable for DDoS classification.

C. *Results for Naive Bayes*

The accuracy for the Naive Bayes classifier came out to be 86.11%, which, compared to other algorithms used in the study, is relatively low. One reason could be dependencies between the features, as discussed in the methodology section. One other reason for this could be imbalanced classes in

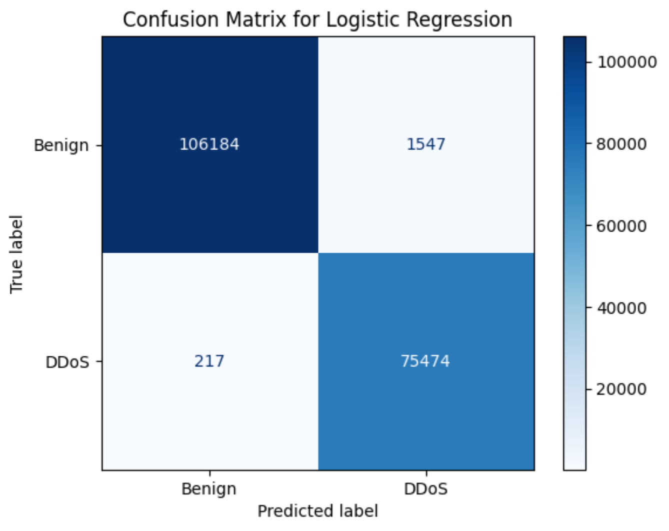

Fig. 6. Confusion Matrix for Logistic Regression.

the dataset since some entries were less compared to others, preprocessing the dataset by balancing the classes can help, however, that reduced the size of the dataset and did not help much in improving the accuracy. While the accuracy was 86.11%, precision, recall and F1-score came out to be 85%.

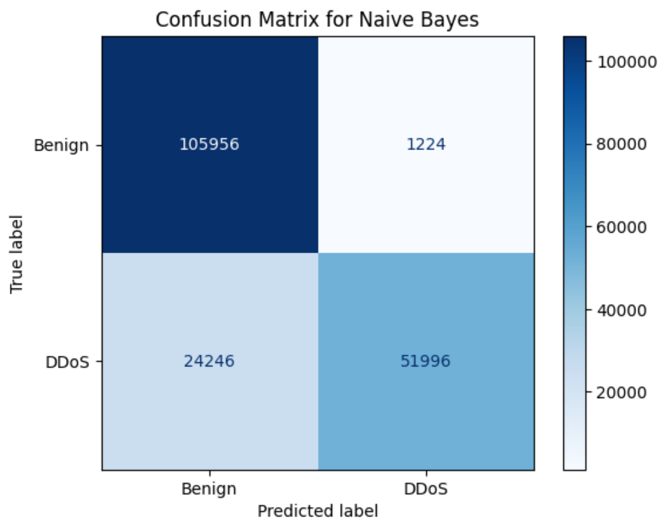

Fig. 7. Confusion Matrix for Naive Bayes.

D. Results for Artificial Neural Network

The trained ANN achieved a high test accuracy of 98.35%, demonstrating strong performance in distinguishing between benign and DDoS traffic using the selected features. Predictions on the test data were made by thresholding output probabilities at 50% to classify each instance as either a DDoS attack or benign traffic. These results indicate the ANN's effectiveness in capturing key patterns within network data, reinforcing its suitability for DDoS attack detection tasks. The precision, recall and F1-score came out to be 98%.

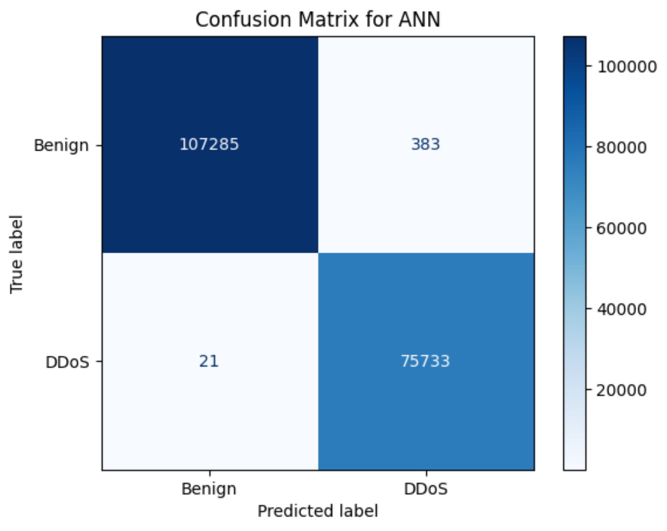

Fig. 8. Confusion Matrix for ANN.

E. Results for Convolutional Neural Network

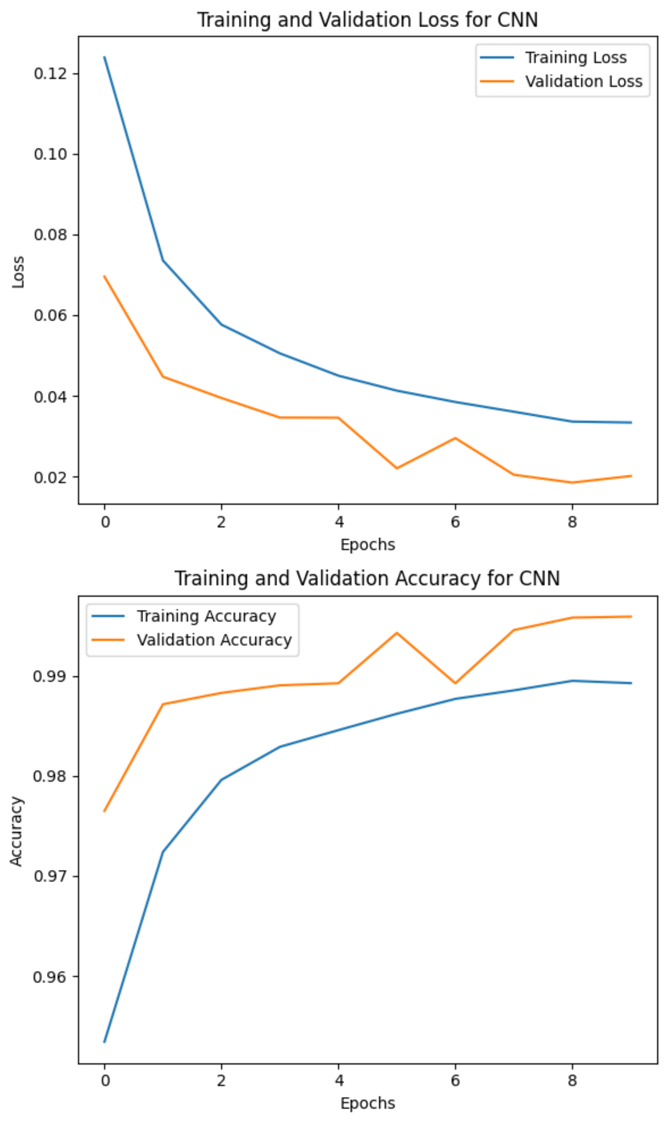

Fig. 10. Learning curve for CNN.

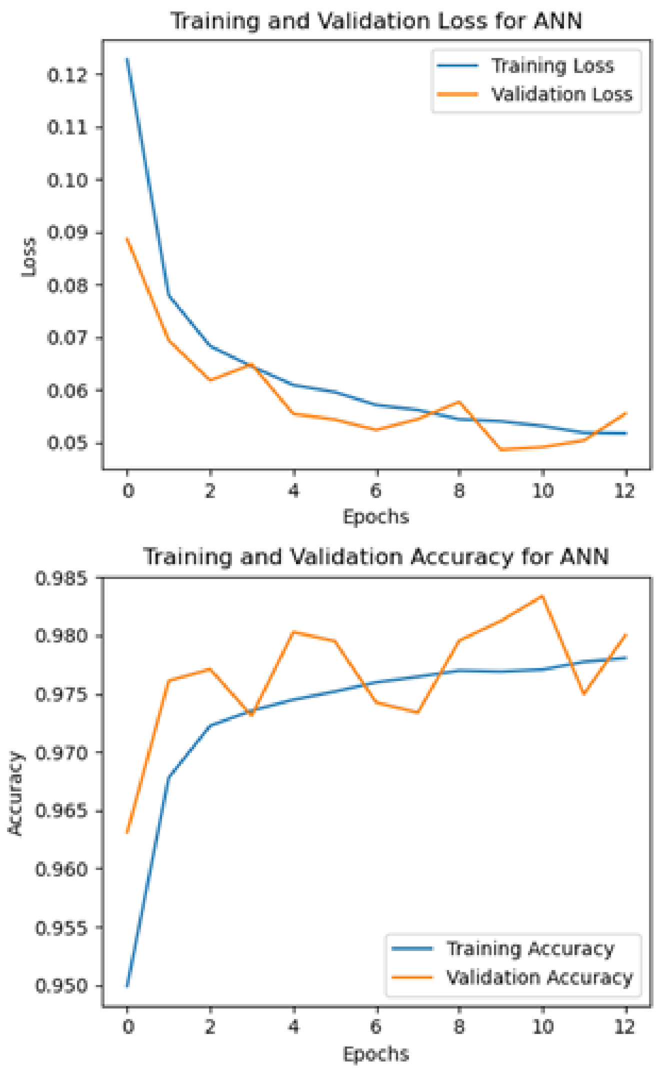

Fig. 9. Learning curve for ANN.

The convolutional neural network model achieved very high metrics on the dataset. On the test data, the accuracy achieved was 99.665%, the precision achieved was 99.664%, the recall achieved was 99.665%, and the F1 score achieved was 99.657%. The model trained for 10 epochs using early stopping and training and validation accuracy increased overall at a decreasing rate over the course of those epochs. Similarly, the training and validation loss decreased at a decreasing rate, as can be seen in [Fig. 10]. The confusion matrix, as shown in [Fig. 11] shows that the most common misclassification by the model was misclassifying DoS Hulk traffic as DoS slowloris traffic.

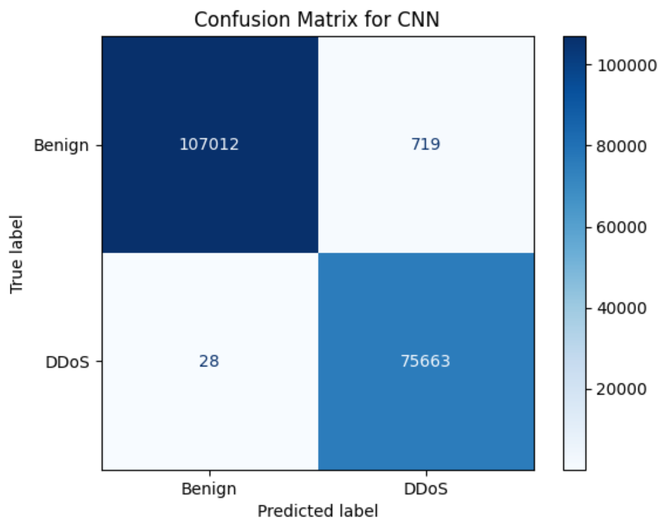

Fig. 11. Confusion Matrix for CNN.

F. Results for Random Forest

The model which was trained using the Random Forest algorithm had a testing accuracy of 99.98%. This shows that this particular algorithm is suitable for detecting suspicious activity within the dataset. The high accuracy signifies that it performs well on unseen data and is able to generalize well on the dataset. The model internally used 100 decision trees which individually give their predictions and the final output is the majority label selected by the trees. This particular model can be used to counter the issue of overfitting. The F-1 score, precision, and recall also turned out to be 99%.

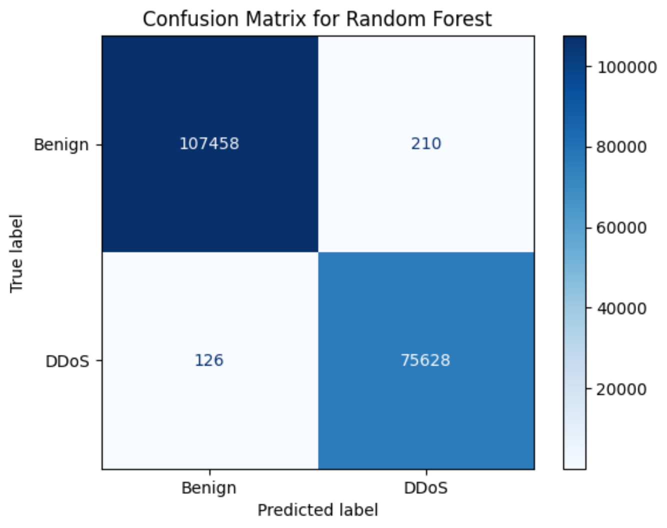

Fig. 12. Confusion Matrix for Random Forest.

G. Results for Decision Tree

For the Decision Tree model, the accuracy is 99.98%, showing that the given features in this dataset are enough for the model to classify if the traffic is benign or DDoS. Based on the confusion matrix [Fig. 13], 107696 instances of Benign and 75668 instances of DDoS were classified correctly while 35 benign were misclassified to DDoS and 23 DDoS to benign. The model also depicts the precision, recall, and F-1 score of the classification report as 100% for both the classes, namely Benign and DDoS. This has established the model as quite reliable for the detection of DDoS attacks.

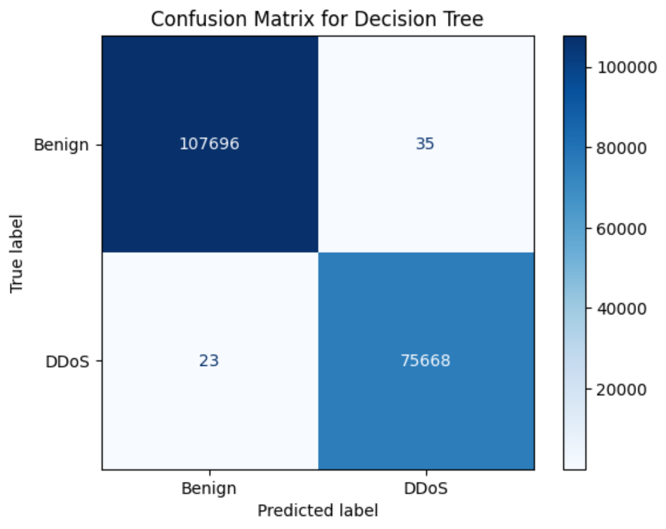

Fig. 13. Confusion Matrix for Decision Tree.

H. *Results for Support Vector Machines*

Nearly every case in the test set was correctly identified by the model, which was able to differentiate between DDoS and innocuous network traffic with an accuracy of 99.1%.

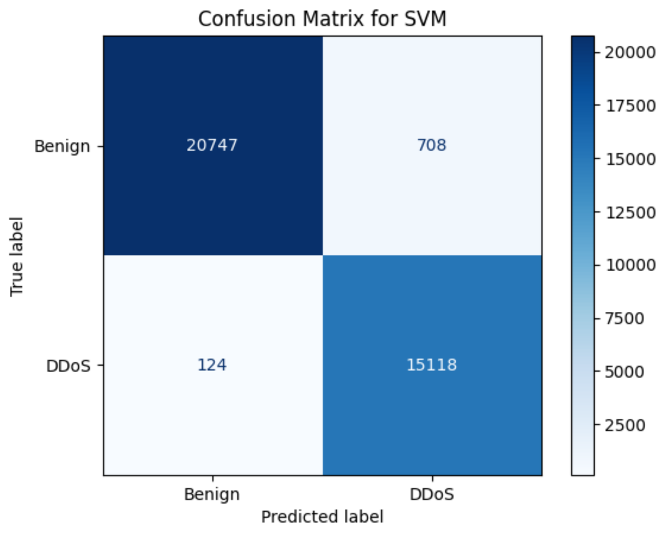

Fig. 14. Confusion Matrix for SVM.

The model successfully detects both classes—BENIGN and DDoS—with no appreciable errors, as evidenced by the precision, recall, and F1-score for both classes being at the maximum score of 100%. The model's high performance across metrics shows that it is dependable, capturing nearly all actual DDoS incidents without false negatives and minimizing false positives, which are when benign traffic is mistakenly labeled as DDoS [Fig. 14].

I. *Results for k-Nearest Neighbors*

Various values of k were tested using five-fold cross-validation in order to determine which value of k would yield the highest metrics, and the result was that a k value of 1 is best, as in, determining the class of a test sample using only the class of the closest sample yields the most accuracy. The k-nearest neighbors model achieved very high metrics on the dataset when tested on the test set. On the test data, the accuracy achieved was 99.101%, the precision achieved was 99.103%, the recall achieved was 99.101%, and the F1 score achieved was 99.101%. The confusion matrix, as shown in [Fig. 15], shows that there was no misclassification that happened at a significant rate compared to the other misclassifications.

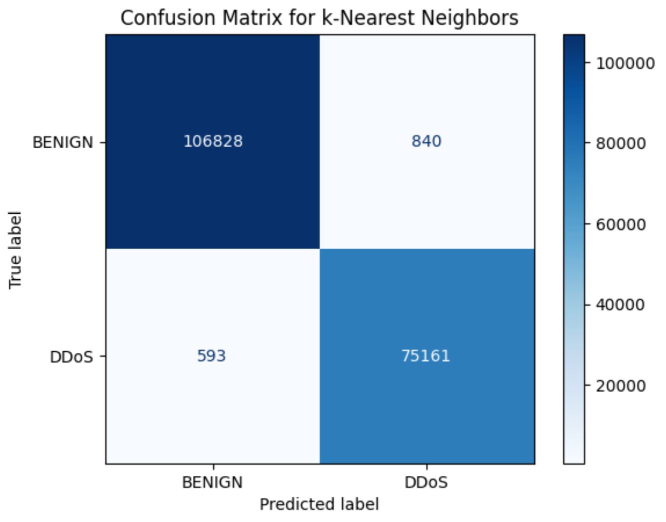

Fig. 15. Confusion Matrix for KNN.

The ROC curve shown in [Fig. 16] compares the performance of six different machine learning models: Random Forest, Logistic Regression, Neural Network, Naive Bayes, K-Nearest Neighbors, and Support Vector Machine. The Area Under the Curve (AUC) values provide a quantitative measure of each model's classification performance. Logistic Regression, K-Nearest Neighbors, and SVM all achieve a perfect AUC of 1.00, indicating that these models are highly effective at distinguishing between the positive and negative classes without making false predictions. The Neural Network and Naive Bayes models also perform very well with AUCs of 0.99. The Random Forest model, however, performs poorly with an AUC of 0.42, suggesting that it is significantly less effective in this context and performs worse than a random classifier (AUC = 0.50). The dashed line represents a random classifier's performance, serving as a baseline for comparison. Overall, Logistic Regression, K-Nearest Neighbors, and SVM

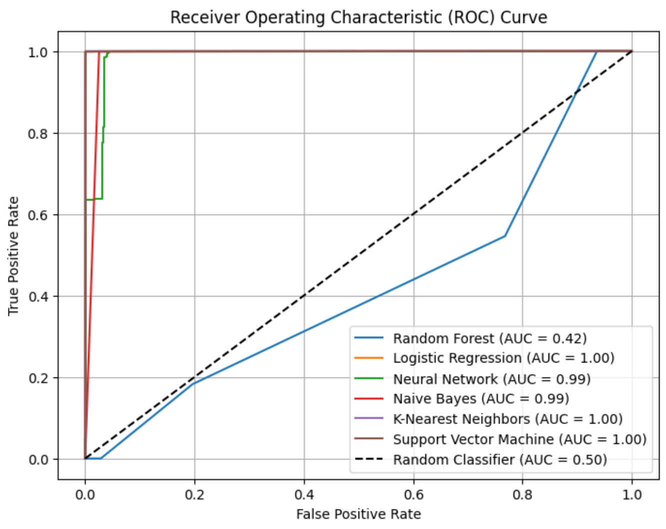

Fig. 16. Receiver Operating Characteristic Curve

stand out as the best classifiers for this dataset based on their ROC curves and AUC scores.

V. Discussion

Each of the eight models was able to achieve a moderate to high level of accuracy on the segment of the dataset which was used to train and test each of the models. In order from highest to lowest accuracy, random forest was able to achieve an accuracy of 99.98%, decision tree was able to achieve an accuracy of 99.98%, logistic regression was able to achieve an accuracy of 99.86%, convolutional neural network was able to achieve an accuracy of 99.67%, k-Nearest Neighbors was able to achieve an accuracy of 99.101%, support vector machine was able to achieve an accuracy of 99.1%, Artificial Neural Network was able to achieve an accuracy of 98.35%, and Naive Bayes was able to achieve an accuracy of 86.11%.

Every technique besides Naive Bayes is very well-suited to the task of identifying DDoS attack network traffic among benign network traffic, as they were each able to achieve accuracies over 98.35%. The techniques of decision tree and random forest, which is built using an ensemble of decision trees, were able to achieve the highest accuracy, seemingly only misclassifying a couple dozen samples out of a total of almost one hundred thousand. The success of the decision tree and random forest classifiers can be partially attributed to the fact that the depth of the decision trees tested was not limited, so during training, the decision trees were able to learn from every feature included in the data set in as precise of a way as needed.

On the other hand, the Naive Bayes model is an outlier for having such a low accuracy score compared to the other seven models. The low accuracy score returned by the Naive Bayes model can be explained by certain qualities of that machine learning technique. Naive Bayes relies on the assumption that all features in the dataset are independent given the class label. While the preprocessing performed on the data removed all of the features that are clearly derived from other features, such as those that are a ratio of two other features, features likely still remain in the dataset that have complex relationships with other features.

Additionally, Naive Bayes is a linear model in a certain space, so it may have struggled given that the true decision boundary between the different classes is not truly linear. Furthermore, there is far more benign than malicious traffic in the dataset, so any machine learning technique which struggles with imbalanced classes may be impacted in a negative way.

Considering that the data in the network capture is temporally based, it is interesting that machine learning techniques such as convolutional neural networks and artificial neural networks performed so well, considering that those machine learning techniques are not inherently temporal. This could imply that the order in which network traffic is received does little to indicate whether any specific piece of traffic is malicious or benign when compared to other features that can be extracted from the network traffic.

As logistic regression is a classification algorithm that is specified for binary or multi-class classification, it is no surprise that logistic regression performed well on this task, classifying pieces of network traffic from within the segment of the dataset studied as either a benign piece of network traffic or as part of one of several different DoS/DDoS attacks.

It is unusual that the best value of k for the k-nearest neighbors model was found to be 1, rather than some higher value. The model based on k-nearest neighbors performed very well, with an accuracy of over 99%, when only the very closest neighbor was considered, which could have several implications for the data. For instance, this could demonstrate that the classes are distinct in the feature space with clear boundaries between them. This is almost certainly the case, as so many of the machine learning techniques were able to perform so well on the data, which indicates that the features provide plenty of information to separate the classes from each other. Another reason the best value of k could be 1 is that there are few data points in some classes. Compared to the number of data samples which represent benign traffic, 234,286, there are few data points for some of the DoS attacks. DoS slowloris has only 2,327 data points and Heartbleed has only 4,154 data points. When some of the classes are so imbalanced, considering a large number of a point's closest neighbors could prevent data points from smaller classes from being identified correctly.

The primary research question studied was to gauge how well different machine learning algorithms would be able to perform at classifying network traffic as either benign network traffic or as part of a DoS/DDoS attack. In successfully building eight different models utilizing different machine learning techniques and gathering accuracy and other metrics from each, our team successfully gauged how well each of the machine learning techniques performs at classifying network traffic from the dataset. The final ranking of performance between the eight different techniques can be found at the beginning of this discussion section.

Another research question discussed was how machine learning techniques could be implemented into supply chains to prevent the potential damage that is inflicted by cyberattacks such as DoS and DDoS attacks. Implementations of these response mechanisms using these types of techniques were discussed in several of the papers cited as a part of the Literature Review. One paper used a random tree classifier in combination with existing IDS/IPS infrastructure to strengthen those systems, reducing both false negatives and false positives, where a piece of malicious traffic represents a true positive [6]. While only random tree classifiers were used in this case, any machine learning technique could theoretically be used in combination with IDS/IPS systems, as long as there is a network interface for network traffic to be fed into the machine learning algorithm and a response mechanism, such as blocking or reporting traffic that it confidently found to be malicious. Another paper by the authors [11] proposed using Machine Learning techniques in SDNs to detect DDoS attacks. SDNs are being leveraged more widely in supply chain systems because they provide a dynamic approach to improving network performance and monitoring. SDNs also provide real time traffic data monitoring and analysis leading to quicker DDoS attack detection. The paper proposes using the Random Forest algorithm for the classification process. The authors ran the algorithm on data collected by an SDN-based supply chain network where the data was accumulated by switches, hosts, controllers, and contained parameters such source, destination, packet length, and ports. The model was able to achieve an accuracy of 98%.

In contemporary supply chains, the integration of machine learning-based DDoS detection systems is becoming increasingly prevalent within Intrusion Detection and Prevention Systems (IDPS) that are part of Security Information and Event Management (SIEM) frameworks. These systems leverage real-time data analysis to identify and counteract DDoS threats by scrutinizing network traffic for irregularities at essential nodes, including distribution centers and data hubs. Typically, algorithms such as Random Forest and Neural Networks are incorporated into these systems to evaluate packet behaviors, identify malicious signatures, and accurately classify traffic as either benign or harmful.

Moreover, Software-Defined Networking (SDN) enhances DDoS defense mechanisms in these configurations by providing centralized management of data flows and enabling dynamic rerouting to thwart the spread of attacks. For instance, when a segment experiences unusual traffic spikes that suggest a DDoS attack, SDN can effectively isolate the impacted area while machine learning algorithms evaluate the severity of the threat. This synergy between SDN and machine learning facilitates swift decision-making, thereby reducing downtime and safeguarding data integrity.

However, the large-scale implementation of these systems presents several technical challenges. Extensive supply chains produce substantial amounts of data, necessitating powerful processing capabilities and adequate storage solutions. As a result, hybrid processing models are frequently utilized—employing real-time processing for critical areas and batch processing for less urgent data streams to optimize computational efficiency.

The machine learning algorithm chosen for these purposes would depend on the availability of training data. As all of the machine learning techniques discussed in this paper are supervised machine learning techniques, which need training data to train the models, these techniques would only be suitable for situations in which there is training data available. Otherwise, one would need to consider unsupervised machine learning techniques such as clustering or principal component analysis. Within the realm of supervised machine learning techniques, one would want to consider the features in the data available and the balance of classes in training data to determine which machine learning technique to choose, along with the accuracy needed for the system.

There are several limitations present in the research described in this paper. For example, as the eight different machine learning models were all built and tested by different team members on different machines, there are some inconsistencies between them. The reason that the efficiencies of different techniques are not discussed is that the times for the models to be run on the dataset would not be consistent between different models as different models are being tested on different computers. Additionally, different team members used different Python libraries as the bases for their models, with some team members primarily using the Keras library and with other team members primarily using the scikit-learn library. In order to hold as many factors constant as possible between different models, it would be best for continuing research to test all models on one machine, either using cloud computing or by having one team member test each model, and to use the same library for all models whenever possible. These measures would effectively hold those factors constant.

Another limitation of the above research is that only DDoS and DoS cyberattacks were addressed. Future research could be performed testing these machine learning techniques on the entire CICIDS2017 dataset, which includes other varieties of cyberattacks such as brute-force attacks, web attacks, infiltration attacks, and botnet attacks. It would be interesting to see whether different machine learning techniques can identify different varieties of cyberattacks with varying degrees of accuracy. While decision trees and random forests achieved the highest accuracy on the segment of the dataset used in this research, other machine learning techniques may perform better on traffic containing different attacks.

Furthermore, only one version of each machine learning technique was tested on the dataset. Taking convolutional neural networks as an example, there are several different ways in which a convolutional neural network can be constructed. Transfer learning can be utilized to dramatically increase the power of a CNN in most cases. Further research could build multiple tiers of every machine learning technique to see if most computational time and power leads to improved metrics.

VI. Conclusion

In conclusion, this study underscores the transformative potential of machine learning in enhancing DDoS attack detection and prevention within network security. By analyzing the performance of various machine learning algorithms, Random Forest and Decision Tree are recommended as the most suitable models for DDoS detection tasks, particularly in environments requiring minimal false positives and high sensitivity. Although Logistic Regression and ANN also show promise, their performance may vary with dataset complexity, while Naive Bayes' dependency on feature independence assumptions limits its effectiveness in this scenario.

These insights derived from our study suggest that the integration of machine learning into cybersecurity infrastructures not only enhances the identification of malicious network traffic but also improves response strategies. The high accuracy achieved by decision tree-based algorithms highlights the importance of ensemble learning in the context of network security, where handling large volumes of diverse data types and feature dependencies is crucial for success. Furthermore, our findings align with existing research that suggests supervised learning models trained on comprehensive datasets, such as the CICIDS2017, can achieve high levels of predictive accuracy across these network environments.

While the results show promise, they also indicate the need for further tuning and refinement. The tested algorithms could benefit from continuous model optimization, such as hyperparameter fine-tuning and exploration of hybrid models that combine multiple algorithmic approaches. Additionally, considering the use case for these algorithms are in real-time applications with high volumes of traffic and data to monitor, feature engineering techniques such as dimensionality reduction is absolutely critical to enhance efficiency and thereby upkeep with attacks that aim to overflow the system.

References

[1] Azadeh Golduzian. Predict and prevent ddos attacks using machine learning and statistical algorithms. 08 2023.

[2] Ismail, Muhammad Ismail Mohmand, Hameed Hussain, Ayaz Ali Khan, Ubaid Ullah, Muhammad Zakarya, Aftab Ahmed, Mushtaq Raza, Izaz Ur Rahman, and Muhammad Haleem. A machine learning-based classification and prediction technique for ddos attacks. *IEEE Access*, 10:21443–21454, 2022.

[3] Jessica Jones, Jason D. Hiser, Jack W. Davidson, and Stephanie Forrest. Defeating denial-of-service attacks in a self-managing n-variant system. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 126–138, Montreal, QC, Canada, 2019. IEEE.

[4] Amit V Kachavimath, Shubhangeni Vijay Nazare, and Sheetal S Akki. Distributed denial of service attack detection using naïve bayes and k-nearest neighbor for network forensics. In *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 711–717, Bangalore, India, 2020. IEEE.

[5] Francisco Sales de Lima Filho, Frederico A. F. Silveira, Agostinho de Medeiros Brito Junior, Genoveva Vargas-Solar, and Luiz F. Silveira. Smart detection: An online approach for dos/ddos attack detection using machine learning. *Security and Communication Networks*, 2019(1):1574749, 2019.

[6] Jema David Ndibwile, A. Govardhan, Kazuya Okada, and Youki Kadobayashi. Web server protection against application layer ddos attacks using machine learning and traffic authentication. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 3, pages 261–267, Taichung, Taiwan, 2015. IEEE.

[7] Laela Olsen, Anchala Balaraj, Jin Heo, Ahmad Karimi, Yesha Modi, Vignesh Mohana Velu, Nikhil Swami, and Advaith Venkatsubramanian. Colab notebooks, Oct 2024.

[8] Thejavathy Vengappa Raja, Zoheir Ezziane, Jun He, Xiaoqi Ma, and Asmau Wali-Zubai Kazaure. Detection of ddos attack on smart home infrastructure using artificial intelligence models. In *2022 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 12–18, Suzhou, China, 2022. IEEE.

[9] Bharat S Rawal, Sudhanshu Patel, and Mithileysh Sathiyanarayanan. Identifying ddos attack using split-machine learning system in 5g and beyond networks. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, volume 1, pages 1–6, 2022.

[10] D Satyanarayana and Aisha Said Alasmi. Detection and mitigation of ddos based attacks using machine learning algorithm. In *2022 International Conference on Cyber Resilience (ICCR)*, pages 1–5, Dubai, United Arab Emirates, 2022. IEEE.

[11] Anass Sebbar and Karim Zkik. Enhancing resilience against ddos attacks in sdn -based supply chain networks using machine learning. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 230–234, Rome, Italy, 2023. IEEE.

[12] Iman Sharafaldin, Arash Habibi Lashkari, and Ali Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *4th International Conference on Information Systems Security and Privacy*, pages 108–116, Funchal, Madeira, Portugal, 01 2018. Science and Technology Publications.

[13] Xiaoyong Yuan, Chuanhuang Li, and Xiaolin Li. Deepdefense: Identifying ddos attack via deep learning. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8, Hong Kong, China, 2017. IEEE.