Adaptive Machine Learning for Enhanced DDos Detection

Dinesh Reddy Kothur
Master's in Data Analytics
San Jose State University
dineshreddy.kothur@sjsu.edu

Ganaprathyusha Puluputhuri Muni
Master's in Data Analytics
San Jose State University
ganaprathyusha.puluputhurimuni@sjsu.edu

Harshitha Reddy Lingala
Master's in Data Analytics
San Jose State University
harshithareddy.lingala@sjsu.edu

Neeharika Patel
Master's in Data Analytics
San Jose State University
neeharika.patel@sjsu.edu

Sai Kiran Baghavathula
Master's in Data Analytics
San Jose State University
saikiran.baghavathula@sjsu.edu

***Abstract*—In the modern, digitally interconnected world, IoT technology has vastly increased efficiency and productivity in sectors such as healthcare, transportation, and smart homes. The ability of IoT devices to monitor, control, and transmit data in real time has transformed traditional practices, bringing automation into everyday life. However, these IoT networks are susceptible to Distributed Denial of Service (DDoS) attacks, which can cause significant disruptions. Detecting and mitigating these attacks is vital for maintaining the integrity and reliability of network infrastructures. This project aims to use the NetBIOS dataset to assess the effectiveness of different machine learning algorithms in detecting DDoS attacks. By comparing their performance, can identify the most efficient and accurate solutions for practical application. This approach includes comprehensive data preprocessing, feature selection, and the implementation and evaluation of various machine learning models such as Logistic Regression, Random Forest, XGBoost, K-Nearest Neighbors, Support Vector Machine, and Decision Trees. These models demonstrated high accuracy, with several achieving perfect results after hyperparameter tuning. The adaptive machine learning strategy proposed here offers a robust and flexible solution for DDoS detection, enhancing network security and reliability for businesses and individuals, thus helping to prevent financial losses and service interruptions. The anticipated outcome is a machine learning-based system capable of accurately detecting DDoS attacks with minimal false positives and negatives, significantly bolstering the security of network infrastructures against such threats.**

***Index Terms*—DDoS, NetBIOS, IoT, Machine Learning, Logistic Regression, Random Forest, XGBoost, K-Nearest Neighbors, Support Vector Machine, Decision Trees, Hyperparameter Tuning**

I. Introduction

THE IoT- The Internet of Things is a network of interconnected devices that connect and communicate over a network. These are devices equipped with sensors, software, and processors that can collect, analyze, and react to the data available around them. The application set of IoT devices is extensive, including but not limited to home automation, health care, transportation, manufacturing, agriculture, etc.

In today's digital era, more than 25 billion IoT devices are outnumbering non-IoT devices by more than 15 billion speaks about the transition the world went through over the last decade. With an increasing number of smart devices on a large scale, Software Defined Networking (SDN) is the best way for administrators to manage and oversee network traffic instead of traditional hardware-based monitoring. SDN separates traditionally coupled Control Plane and Data Plane in the network paradigm. This provides centralized control over the network thereby simplifying network management. It allows rapid adjustment concerning changing conditions. By decoupling the control plane and data plane, SDN is a cost-effective measure to avoid spending on specialized hardware for network infrastructure and maintenance.

While making everyday devices smart, each of these devices is a gateway into the network. Thus, opening many portals for a cyber-attack on the network. DDoS (Distributed Denial of Service) is one of the most common types of cyber-attack which is designed to disrupt services by overwhelming the target network with malicious traffic. Malicious traffic is usually generated from a swarm of infected devices, often distributed globally. With 25 billion IoT devices out in the world, any device without good security can potentially be used as a bot to generate DoS traffic. Below Figure illustrates the architecture of DDOS attack.

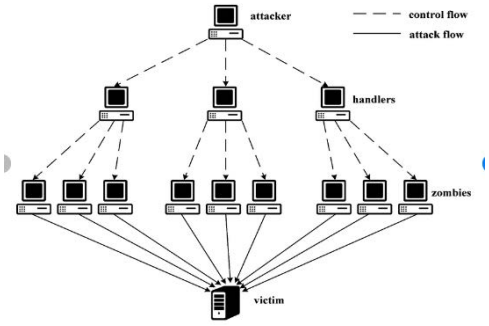

Fig. 1. Architecture of DDOS Attack

These attacks can be categorized into different types:

Volumetric Attacks: Flooding the target with high volumes of traffic to consume bandwidth. Protocol Attacks: Exploiting

weaknesses in network protocols to overwhelm resources. Application Layer Attacks: Targeting specific applications to exhaust server resources.

The evolving nature of these attacks, coupled with their increasing frequency and intensity, necessitates more sophisticated detection mechanisms. DDoS attacks on a business can lead to significant downtime and financial loss. According to a study done by Zayo Group in 2023, an average DDoS attack costs businesses around half a million dollars.

As it is always better to be safe than sorry, detecting a DDoS attack is the best way to avoid it. While there are traditional DDoS detection methods, they are increasingly inadequate for the ever-evolving tactics of modern attackers. Traditional DDoS detection mechanisms typically rely on predefined rules and signatures to identify malicious traffic. These methods include:

Rate Limiting: Restricting the number of requests a user can make in each amount of time. Traffic Analysis: Monitoring network traffic for patterns that indicate an attack. Signature-based Detection: Identifying known attack patterns and signatures.

While these methods provide a baseline level of defense, they suffer from several limitations:

Static Approach: These techniques rely on static rules and signatures, making them ineffective against new, unknown attack patterns.

High False Positives and Negatives: They often misclassify legitimate traffic as malicious and fail to detect novel attack vectors. Scalability Issues: As attack volumes grow, traditional methods struggle to scale without significant manual intervention.

"Change is the only constant" is the philosophy required to adapt to the techniques of modern attackers and we aim to use Adaptive Machine Learning models to achieve dynamic detection of DDoS attacks. Adaptive machine learning offers a transformative approach to DDoS detection. By leveraging the ability to learn from data and adapt to new information, ML-based systems provide a dynamic and resilient defense mechanism against DDoS attacks. Key characteristics of ML models include:

Dynamic Learning: Machine learning systems can identify new and evolving attack patterns since they can learn from both historical and present data.

Pattern Recognition: By analyzing massive amounts of network traffic data, machine learning algorithms can identify complex patterns and anomalies that are suggestive of DDoS attacks. This makes it possible for more accurate detection by reducing the likelihood of false positives and negatives.

Very Little Human Involvement: When ML-based systems are implemented, they require minimal manual oversight. They can autonomously adjust their parameters and improve their performance over time, allowing cybersecurity teams to focus on more strategic tasks.

Preprocessing steps such as normalization, feature extraction, and anomaly detection are crucial to enhance data quality and relevance. To overcome the data imbalance, we have used SMOTE (Synthetic Minority Over-Sampling technique) techniques and Random under-sampling. To retain most of the variance, we have performed Dimensionality reduction using PCA.

After preprocessing data, we have fed the cleaned, balanced and representative data into machine learning models which is very important to build robust and precise predictive models for DDoS detection. Models chosen for the project based on research of other works mentioned below are Linear regression, Support Vector Machine, Decision Tree, Random Forest, K-nearest neighbors, and XG Boost.

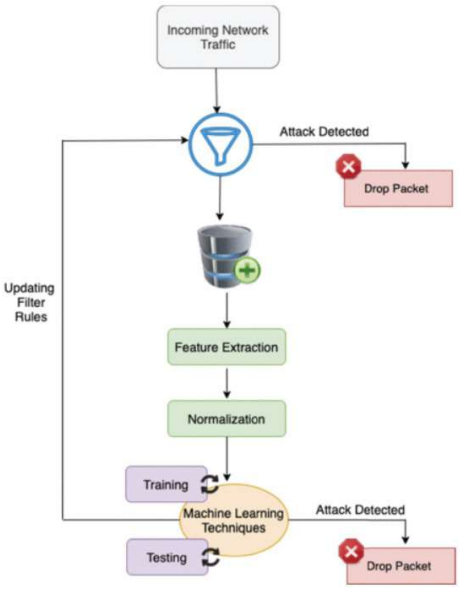

Fig. 2. Adative machine learning for networks.

II. Related works

Alzahrani, Rami & Alzahrani, Ahmed. (2021) [2] study assesses the efficacy of different machine learning algorithms in DDoS detection and offers insights into their performances. The predictive accuracy of the decision tree and random forest achieved 99% each.

In a study by Saeed et al. (2023) [3], a DDoS dataset was used to analyze and preprocess the data before building classification models using Random Forest, Decision Tree, SVM, Naïve Bayes, and XG Boost where the highest accuracy achieved by the random forest model of 99.954% accuracy explains its robustness in classifying DDoS.

In a 2015 study, Kim and Park [5]assessed how well several machine learning algorithms performed in DDoS detection. According to their research, Decision Trees and Support Vector Machines (SVM) performed especially well, demonstrating the potential of ML techniques in cybersecurity.

Sahoo and Pateriya (2018)[6] investigated sophisticated machine-learning methods for detecting DDoS attacks in Internet of Things settings. They showed how adaptive models may greatly increase detection rates while lowering false positives, which is crucial for shielding IoT networks from dynamic threats.

A deep learning method for network intrusion detection was introduced by Shone, Ngoc, and Phai (2018) [7]. Neural networks are used to identify intricate patterns in network traffic data. Their research demonstrated how deep learning may be used to improve the precision and resilience of DDoS detection systems.

Ahmed and Mahmood (2016) [4] analyzed several network anomaly detection techniques that were reviewed, emphasizing the use of machine learning to detect DDoS attacks. The survey emphasized the benefits of adaptive learning models in improving detection capacities while offering a thorough review of both conventional and machine learning-based techniques.

III. Data Understanding and Preprocessing

A. Data Understanding

For this study we utilized NetBIOS dataset which is made openly available by the University of New Brunswick; Canada. The dataset contains 88 columns and 3.4 million rows including the Target label. This data includes network traffic logs, server logs, and other relevant metrics.

Network traffic logs form a significant part of the NetBIOS dataset. These logs record in-depth details about the exchanges that take place between devices on a network. The sources and destination IP address, ports, protocols utilized, and the amount of data transferred are usually included in each entry of the network traffic logs. Patterns of normal and attacked behavior can be found by examining these logs. Port scanning is a frequent precursor to a cyberattack and it may be indicated, for example by repeated access attempts to many ports from a single IP address. Network traffic records are useful for intrusion detection, performance monitoring, and usage analysis.

Another important component of the NetBIOS dataset is provided by server logs. Records of server activities, such as requests made or received, responses sent, error messages, and other operational information are kept in these logs. For monitoring purposes and the health of servers, these logs are important. These can help in identifying problems including resource usage, failed procedures, and server outages. Server logs include request details, error logs, and access logs. Request details contain information on each request received by the server. Error logs contain complete records of error messages if any encountered, their type, severity, and causes. Access logs contain information on user access, which includes timestamps, user IDs, and access patterns. Apart from network traffic and server logs, the NetBIOS dataset has other features that enhance more context of the information.

There are a few key features of the dataset. Destination port, which facilitates locating the desired service and identifying service-specific assaults, for instance, one can analyze the destination port. Certain services relate to specific ports like port 80 for HTTP.

Flow-based features are one of the key features that give a higher level of overview of traffic flow, which is useful for identifying large volumes of unidirectional traffic, mainly DDoS attacks. They are Flow duration which indicates the total time duration of the flow, Total Forward packets represents total packet numbers sent in the forward direction, Total Backward packets has the total number of packets sent in the backward direction, total length of forward packets has length of packets which sent in the forward direction, Total length of backward Packets has the length of the packets which sent in the backward direction.

Another key feature is Flag counts which have SYN, RST, PSH, Ack, and URG flag counts. These flags are components of the TCP header and control how TCP connections are established and closed. Specific combinations of these signals may be indicative of distinct DDoS attack types. For instance, a high number of SYN flags without the corresponding ACK flags may be a sign of SYN flood assault. This is a common type of DDoS attack in which a target server is overwhelmed by many SYN requests.

Packet-base features are another key feature at the packet level in identifying network behavior including possible security risks like DDoS attacks. This has detailed information about the packets exchanged within the network. Key packet-based features are forward packet length (Max, Min, Mean, Std), Backward packet length (Max, Min, Mean, Std), Packet length stats (Mean, std, and variance).

The success of the machine learning models models depend largely on the quality of the data. To enrich the dataset this study several data preprocessing and data transformation techniques including data sampling and dimensionality reduction were employed.

B. Data Preprocessing

Upon inspecting the dataset, a total of 6 missing values were found. As the missing values were minimal and all occurred within rows labeled as 'NetBIOS', rows with the missing values were dropped from the dataset. Some of the features have infinite values and these values can occur due to the errors in data collection, it can negatively affect the model performance. Overall detected 130,555 infinite values in the dataset, which were dropped as they do not constitute even for 1% of the dataset, and dropping these wont impact the model's performance. Outlier detection has been done on the dataset , the dataset has many outliers, this can skew the results. So dropped the columns with outliers to remove the skewness.

There are many features in the dataset that do not influence the target variable. Some of these features include :Unnamed: 0', 'Flow ID', 'Destination IP', 'Source IP', 'Timestamp', 'Fwd PSH Flags', 'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk', 'Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate','SimillarHTTP'. Also the 'Fwd Header Length' column in the dataset is repeated twice under a different name: 'Fwd Header Length.1', which is dropped to maintain the integrity of the dataset.

C. Data Sampling

The initial understanding of the data revealed that this is a binary classification problem and the class 1 which is

'NetBIOS'class dominates the dataset heavily and the class 0 which is the 'Benign' class has a very negligent number. This suggests that the dataset is heavily imbalanced. To address this problem, we employed an integrated data sampling strategy, which is a combination of both under-sampling the majority and over-sampling the minority class.

For under-sampling, 'RandomSmapler' from 'imblearn' library, with a sampling_startegy of 0.2 was used to under-sample the majority class, meaning after undersampling the majority class instances will be in a ratio of 5:1 with the minority class instances.

After undersampling, the dataset was split into train and test sets using the 'train_test_split' method of 'Scikitlearn' library in the ratio of 80:20. This is done before applying the over-sampling strategy through 'SMOTE', as SMOTE generates synthetic data samples of the minority class and we do not want to test our model performance against synthetically generated data but on real world data instances. The sampling_strategy we used in SMOTE is 0.5, which added 50% synthetic samples of the minority class instances to the dataset, which brings the ratio of majority class to minority class to 2:1.

The class distribution of before and after applying under-sampling and over-sampling techniques is shown in figures 3 and 4 below. This integrated strategy of under-sampling and over-sampling ensures that the models are trained on a balanced dataset, which significantly helps in effective modeling by reducing overfitting and enabling the models to generalize well on new unseen datasets.

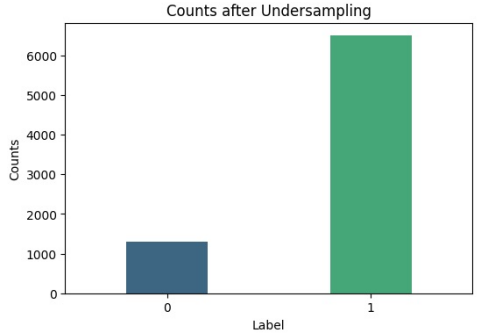

Fig. 3. Class Distribution after Undersampling

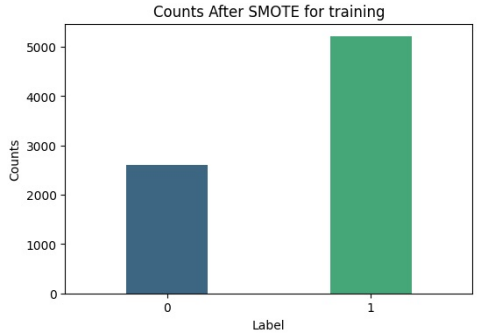

Fig. 4. Class Distribution after Oversampling

To further enrich the dataset, Principal Component Analysis (PCA) was employed to reduce the dimension complexity in the dataset by capturing the most important variance in the dataset, which helps in reducing overfitting.

D. Dimensionality Reduction: PCA

Principal Component Analysis (PCA) is a statistical technique used for dimensionality reduction, feature extraction, and data visualization. PCA transforms the data into a new coordinate system so that the greatest variances by any projection of the data come to lie on the first coordinate, known as the first principal component. The second greatest variances lie on the second coordinate, and so on.

PCA is used to reduce the number of dimensions, which is the number of features in the dataset, while retaining most of the variability present in the data. It helps simplify large datasets without losing much information. The NetBIOS dataset has extensive network traffic data with 3.4 million rows and 88 columns. Because of the high dimensionality and volume of the dataset, we perform PCA to enhance the machine learning model's performance and efficiency. By reducing the number of features, PCA helps simplify the dataset while retaining important information. Large datasets contain noise and many irrelevant features that might negatively impact model performance. PCA helps filter out the noise by focusing on the features that capture most variance in the data, improving the signal-to-noise ratio. PCA reduces the feature space dimensionality, making it more manageable and enhancing the algorithm's ability to detect patterns and anomalies in DDoS attacks. PCA also helps prevent overfitting.

To perform PCA, data standardization must first be carried out, maintaining a mean of 0 and a variance of 1. The equation (1) shows the formula to standardize the data.

$$Z = \frac{X - \mu}{\sigma} \tag{1}$$

Where σ is the standard deviation and μ is the mean of X.

The next step is computing the covariance matrix, which is used to understand the relationships between the variables. It captures how much variables vary together.

$$C = \frac{1}{n-1} Z^T Z \tag{2}$$

Next, we calculate the eigenvalues and eigenvectors of the covariance matrix. Eigenvectors determine the direction of the new feature space, and eigenvalues determine the magnitude.

$$Cv = \lambda v \tag{3}$$

Where λ are the eigenvalues and v are the eigenvectors.

The eigenvalues and their corresponding eigenvectors are sorted in descending order. The top 'k' components form the principal components, which are stored in matrix V_k. The original data is transformed into the new feature space defined by the principal components.

$$X_{PCA} = ZV_k \tag{4}$$

X_{PCA} is the transformed dataset.

Process	Dataset Size
Original Dataset	3455899 X 88
After removing missing values	3455894 X 88
After removing infinite values	3325339 X 88
After removing redundant columns	3325339 X 70
After removing Outliers	2054816 X 70
Under-sampling	7794 X 70
Train set	6235 X 70
Test set	1559 X 70
Train set after SMOTE	7812 X 70
After PCA	7812 X 19

Fig. 5. Transformed Data After PCA

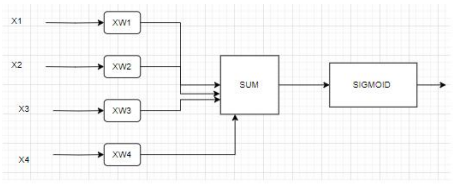

Fig. 7. Sigmoid Function

IV. Modeling

In this project, models like Logistic Regression, Random Forest, Decision Tree, KNN, XGBoost and SVM was implemented to classify the NETBIOS traffic. Figure 6 explains the data flow and model architecture of this project.

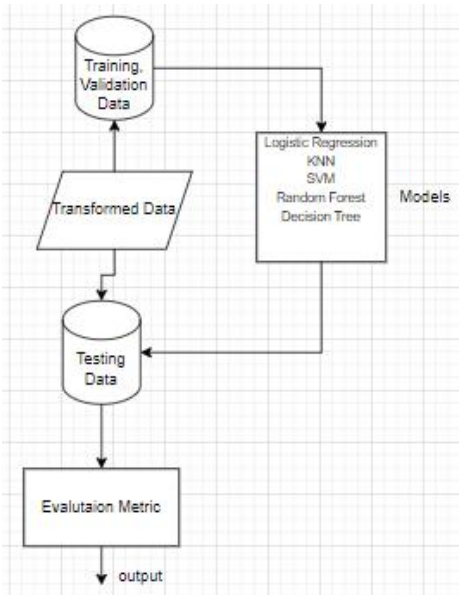

Fig. 6. Model Architecture and Data Flow

A. Logistic Regression

Logistic Regression is most commonly used algorithm in binary classifier. In this project, Logistic Regression model is implemented to estimate whether the traffic is being NETBIOS traffic or not. It makes predictions with the help of "sigmoid function". This function generally takes the numbers as the input and scales them in between the range of zero and one, makes the input numbers suitable for modeling. Figure 7 the visual representation of the working of sigmoid function. Equation 5 represents the formula for sigmoid function for z value greater than zero

$$(z) = 1/1 + e^{-z} \tag{5}$$

The model calculates the probabilities of the input data and it includes weights, which it learned during the training to determine whether the traffic is malicious or benign. Then the models use the probabilities set by sigmoid function to make classification. If the probability is greater than the threshold then it classifies the traffic as NetBIOS otherwise, it classifies as benign. Logistic Regression uses a method called Maximum Likelihood Estimation to find the good weights.

In this project, Logistic regression was applied before regularization and after regularization. To overcome multicollinearity issues, PCA has applied and the data from the PCA are given as input to the model. StandardScaler was incorporated with pipeline for the regularization model. In this project, L1 regularization was used, to make the model generalize and to perform well on the unseen data. In both the models Logistic Regression before and after regularization are initialized with random state to ensure the same data was splitted when the model is run. L1 regularization was implemented with low value of c for strong regularization effect. Both the models are trained on PCA features and tested on the initially splitted data. Created a custom function 'evaluate_predictions' to evaluate the models based on the metrics like accuracy, precision, confusion matrix and recall, which provides the insights of the performance of the model.

Scikit-Learn library was used to implement Logistic regression. PCA and Standard Scaling was used in preprocessing phase to reduce the dimensions of the dataset. To maintain the integrity of the model evaluations, pipeline was implemented which executes the steps sequentially.

The computation of the model depends on the number of iterations. If the number of iterations is high, it requires high computational resources. Logistic regression before regularization is simple can easily prone to overfitting. Logistic regression after regularization, which has strong L1 regularization, which increased the understanding of the model by reducing the number of features. To scalability and efficiency of the large dataset like NetBIOS, 'saga' solver was used. Below Figure 8 shows summary of key aspects used in Logistic regression before and after regularization.

Aspect	Initial Model	Optimized Model
Setup	5trees, max depth of 10	Optimized parameters
Parameter Settings	Fixed	Varied
Parameter Tuning	Static Parameters	Used RandomizedSearchCV to find best parameters
Model Complexity	Simpler	Complex
Computational Effort	Lower	Higher

Fig. 8. Comparison of Model for Before and After Regularization

B. Random Forest

Random Forest is a popular ensembled technique used in the classification projects like Adaptive Machine Learning for Enhanced DDos Detection. Assumed, individual decision trees in the forest makes independent predictions and also some of the predictors of the dataset are relevant to target variable. It also handles large datasets like NETBIOS effectively. Random Forest is a popular ensembled technique used in the classification projects like Adaptive Machine Learning for Enhanced DDos Detection. Assumed, individual decision trees in the forest makes independent predictions and also some of the predictors of the dataset are relevant to target variable. It also handles large datasets like NETBIOS effectively. Scikit-Learn

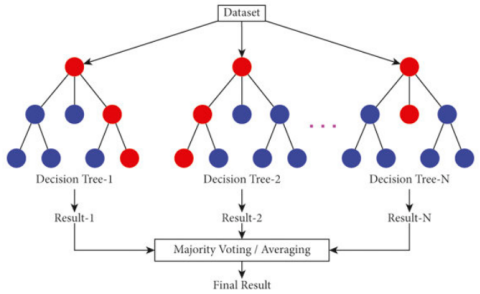

Fig. 9. Class Distribution after Undersampling

library was used to implement Random Forest Classifier. This model in this project was implemented in two ways, one with specified parameters and other is an improved version using RandomizedSearchCV for hyperparameter tuning. Th dimensionally reduced data was given as input to the model and also assumed more trees, deeper trees gives better performance of the model and also can lead to overfitting.

Initially, the model was configured with five trees and a maximum depth of 10. RandomizedSearchCV was used to explore different values for parameters like max_depth, min_samples_split, n_estimatores and applied cross validation to optimize the accuracy. It is evaluated on the metrics to understand the performance of the model. In case of computation, model was configured initially with 5 trees and increased upto 200 trees and different depths during hyperparameter tuning. Requirements of Computational resources increases when the number of estimators and tree depth increses. Figure 10 comapres the key aspects of the initial Random Forest model and Optimized model of this project.

Aspect	Initial Model	Optimized Model
Setup	5trees, max depth of 10	Optimized parameters
Parameter Settings	Fixed	Varied
Parameter Tuning	Static Parameters	Used RandomizedSearchCV to find best parameters
Model Complexity	Simpler	Complex
Computational Effort	Lower	Higher

Fig. 10. Comparison of Simple and Optimized Model

C. K - nearest neighbors(KNN)

KNN is a straightforward algorithm used in both classification and regression tasks. K is defined as the number of nearest neighbors, want to consider for making consideration. The distance between the points is calculated by Manhattan distance method. Equation 6 represents the formula for Euclidean distance.

$$Euclideandistance = |x1–x2| + |y1–y2| \quad (6)$$

Based on the calculated distance, model selects the top k closest points. The class with the majority votes becomes the prediction. Figure 11 illustrates the flow diagram of KNN algorithm.

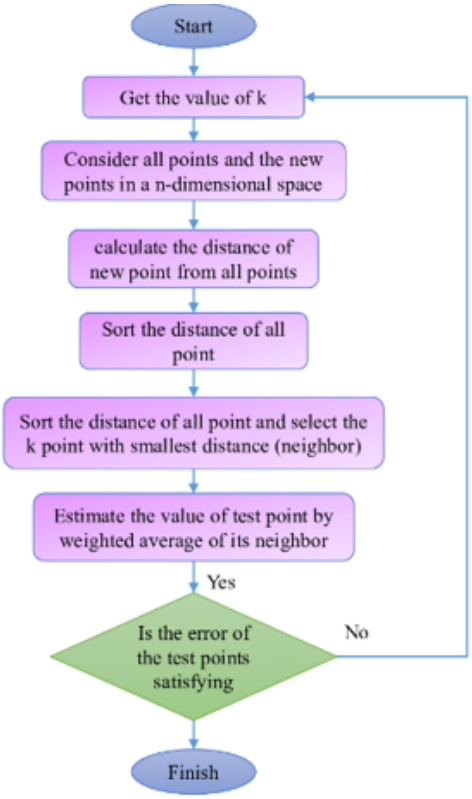

Fig. 11. KNN Flow Chart

KNN model is trained on the PCA transformed data and tested on test data. This is implemented by using Scikit-Learn library, which is powerful tool for handling machine learning algorithms. Standardscaler and pipeline are integrated to ensure feature scaling is applied correctly. The KNN model in this project is configured with 3 neighbors and used Manhattan distance for calculating the distance. The model is evaluated on different metrics like accuracy, recall, precision and F1 score.

KNN is expensive computationally, when there are large number of features and data points. The limitations of KNN is not that scalable.

D. XG Boost

XGBoost is a machine learning algorithm widely used in supervised learning tasks like prediction of NETBIOS traffic. XGBoost uses sequential building to build the decision trees. It uses two components loss function and regularization to measure the model's predictions and to prevent overfitting. It first calculates the gradient of loss function and then builds the trees to predict these gradients. It also has in built quality to handle missing values. In this project, XGBoost is set up for

the binary classification task. And initialized the parameters like n_estimators, learning_rate, max_depth, subsample and colsample_bytree. The model is trained, tested on the PCA transformed data and validated using metrices like accuracy, precision and recall. Figure 12 describes the architecture of XGBoost. XGBoost library was used to implement this model.

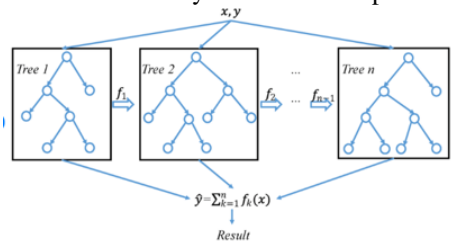

Fig. 12. Architecture of XGBoost

Utilized pipeline for maintaining model's performance. It is generally highly efficient but, it becomes inefficient when the dataset is very large like NETBIOS.

Support Vector Machines

SVM is a versatile supervised machine learning model. It works by finding the margin which divides the classes by large gap. This margin is found by using support vectors. SVM is highly efficient for complex datasets, where classes are not clearly divided. One of the limitations SVM possess is it struggles with the noisy data. Figure 13 illustrates the classification by SVM. In this project, SVM is configured

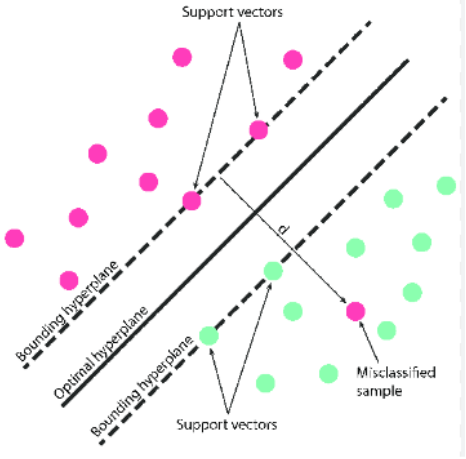

Fig. 13. SVM Classification

with a linear kernel and with c equals to 0.1. SVM is trained on the dimensionally reduced dataset. Used metrices like accuracy, precision, recall and F1 score to know the model's performance.

Scikit-Learn library was used to implement the SVM model. Utilized pipeline to maintain the integrity and accuracy of model's performance. SVM's, with linear kernel are efficient but, becomes computationally expensive when the number of features and samples increase. And it requires careful selection of c value and the type of kernel.

Decision Tree

Decision tree is non-linear model used extensively in machine learning for classification tasks. It works by considering into smaller subsets. Each internal node of the tree corresponds to a test on an attribute. Each branch indicates the outcome of the test and each leaf node represents class label. Classification rules are defined by paths from root to leaf. At each step, It asks questions about the feature. Based on the answers, the data is split into smaller groups. After the entire splitting, each group ends with an answer. In this project, basic decision tree and tuned decision tree was implemented. Figure 14 illustrates the classification by Decision Tree.

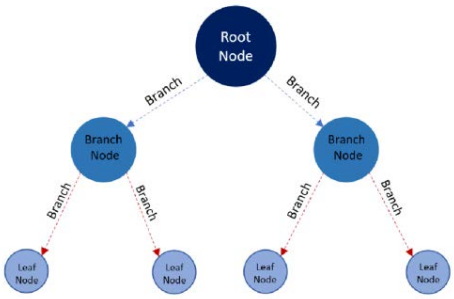

Fig. 14. Architecture of Decision Tree

The first decision tree is simple, with only max_depth, which is regularized to prevent overfitting and cannot capture the complex patterns in data. Second decision tree consists of different parameters to manage complexity and prevent overfitting. Both the models are trained on the dimensionally reduced dataset and tested on the same dataset to make predictions. The model's performance was evaluated using F1 score, precision, recall and accuracy.

Scikit-Learn's DecisionTreeClassifier was used to implement this model. Utilized pipeline for maintaining the validity of model's performance. Simpler Decision Tree is computationally less complex and second model, with parameters was computationally demanding and better at handling complex data like NETBIOS.

Decision tree generally has inherent feature selection. But, in the second model max_features and max_leaf_nodes are used to manage the scalability effectively.

V. EVALUATION

A. Model Evaluation Metrics

1) Confusion Matrix: Confusion matrix provides a comprehensive overview of the model's performance. In a binary classification problem, it captures four possible outcomes: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). It is depicted as given in the figure below:

2) Precision: Precision measures the accuracy of a model's positive predictions, calculating the proportion of true positives out of all predicted positives. A higher precision indicates that the model is accurate and reliable in its predictions, with fewer false positives.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{7}$$

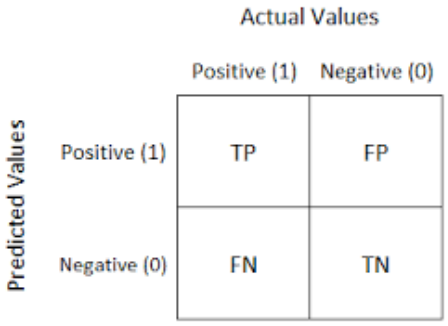

Fig. 15. Confusion Matrix

3) Recall: Recall measures the proportion of true positive predictions out of all actual positive instances, evaluating a model's ability to detect all instances of the positive class. A higher recall indicates that the model is good at detecting all positive instances, but may also lead to more false positives.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{8}$$

4) F1 Score: F1 score is a harmonic mean of precision and recall, providing a single metric that balances both accuracy and completeness. It rewards models that achieve a good balance between precision and recall, making it a useful metric for evaluating model performance.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

5) Accuracy: Accuracy measures the proportion of correct predictions out of all predictions made, evaluating the model's overall performance. A higher accuracy indicates that the model is good at making correct predictions, but may not reveal the full story (e.g., a model with high accuracy but low recall may be missing many positive instances).

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Predictions}} \tag{10}$$

B. Model Evaluation

1) Logistic Regression : We initially chose Logistic Regression for its simplicity and effectiveness in binary classification problems. Initially, the model was trained with specific parameters and up to 1000 iterations, achieving a perfect accuracy of 100% on the test dataset with no classification errors. However, this perfect performance indicated potential overfitting, where the model performs well on training data but poorly on new data. We applied L1 regularization to mitigate this, which reduces model complexity by adding a penalty. After regularization, the model's accuracy decreased to 82.8%, correctly identifying all 1287 true positives but misclassifying 268 out of 272 true negatives. This trade-off between accuracy and generalization underscores the importance of balancing these factors to ensure better performance on unseen data.

2) Random Forest Classifier: Following the execution of RandomizedSearchCV, the optimal model achieved a remarkable accuracy of 100%. The confusion matrix illustrates the model's flawless classification, with all 272 true negatives and all 1287 true positives correctly identified, resulting in no false negatives or false positives. Additionally, the classification report confirms these perfect results, showing precision, recall, and F1-scores of 1.0 for both classes. These outcomes indicate the model's ability to perfectly differentiate between the classes in the test dataset. The Random Forest classifier, fine-tuned using RandomizedSearchCV with various parameters such as the number of trees, maximum depth, minimum samples for splitting a node, and minimum samples required at a leaf node, demonstrates exceptional robustness and predictive accuracy.

3) XGBoost: We chose XGBoost for its powerful gradient boosting framework, which excels in handling large datasets and complex patterns, offering superior performance and scalability compared to many other algorithms. The XGBoost model achieved an accuracy of 82.6%. However, the confusion matrix reveals a notable limitation: the model did not identify any true negatives, misclassifying all 272 instances as positives. Conversely, it accurately classified all 1287 true positives, resulting in no false positives. Although the classification report shows strong metrics for class 1, metrics for class 0 are irrelevant due to the lack of correctly identified true negatives. This indicates the model's proficiency in detecting positive cases but highlights its struggle with negatives. Further refinement might be necessary to improve the model's ability to correctly classify negative instances.

4) KNN Classifier: The K-Nearest Neighbors (KNN) model attained an accuracy of 100%. The confusion matrix demonstrates perfect performance, with all 272 true negatives and 1287 true positives correctly identified, resulting in no false negatives or false positives. This flawless classification is further supported by the classification report, which shows perfect precision, recall, and F1-scores of 1.0 for both classes. These outcomes highlight the model's excellent capability to accurately differentiate between the classes in the test dataset. The KNN model, using three neighbors and the Manhattan distance metric, showcases its robustness and high predictive accuracy, as reflected in the perfect accuracy score and comprehensive metrics in the confusion matrix and classification report.

5) Support Vector Machine (SVM): We chose the Support Vector Machine (SVM) model for its effectiveness in high-dimensional spaces and its robustness in handling linear classification tasks. The SVM model achieved an outstanding accuracy of 99.9%. The confusion matrix reflects excellent performance, with only one misclassification each for false negatives and false positives out of 1559 instances. Specifically, among the 272 true negative cases, 271 were correctly identified, and all 1287 true positive cases were accurately classified. These results are further supported by the classification report, which shows near-perfect precision, recall, and F1-scores for both classes. Utilizing a linear kernel and a regularization

parameter, C, set at 0.1, the SVM model demonstrates strong predictive ability and robustness, as evidenced by the nearly flawless accuracy score and comprehensive metrics in the confusion matrix and classification report.

6) Decision Trees: The Decision Tree model was chosen for its simplicity, interpretability, and ability to handle both numerical and categorical data effectively. When properly tuned, Decision Trees are robust against overfitting, making them a reliable choice for classification tasks. The model achieved a perfect accuracy score of 100%, demonstrating flawless performance on the test dataset, with the confusion matrix showing 272 true negatives and 1287 true positives correctly classified and no misclassifications. The classification report confirmed these results, with perfect precision, recall, and F1-scores of 1.00 for both classes. Configured with advanced regularization parameters, including a maximum depth of 1 and 5, minimum samples split of 20, minimum samples leaf of 10, maximum features of the square root, and maximum leaf nodes of 30, the model demonstrated robust predictive capability and resilience against overfitting, underscoring its exceptional ability to distinguish between classes in the test dataset accurately.

VI. DISCUSSIONS

The analysis show diverse performances among various algorithms. Logistic Regression demonstrated an accuracy of 100%, suggesting model overfitting and after applying regularization, its accuracy marginally reduced to 82.8%, making the model generalizable to new unseen data. While the Random Forest Classifier exhibited robustness, maintaining an accuracy of 100% even after hyperparameter tuning. And, XGBoost showcased a strong predictive power with an accuracy of 82.6%. KNN Classifier showed an accuracy of 100%, suggesting effective classification based on nearest neighbors. SVM achieved near perfection with an accuracy of 99.9%, showcasing excellence in instance classification using support vector machines. Similarly, Decision Trees showed an accuracy of 100%, indicating clear and effective decision-making based on the dataset. These insights offer a comprehensive understanding of each model performance, underscoring their individual strengths and capabilities in accurately classifying data.

Model Name	Accuracy
Logistic Regression	100%
Logistic Regression with Regularization	82.8%
Random Forest Classifier	100%
RF with hyperparameter Tuning	100%
XGBoost	83%
KNN Classifier	99.5%
SVM	99%
Decision Trees	100%
Decision Trees with hyperparameter Tuning	100%

Fig. 16. Comparison of Models

VII. DEPLOYMENT

Our adaptive machine learning model for DDoS detection is being implemented with the intention of improving attack identification errors mitigation. Our model's scalability is ensured by utilizing cloud and edge computing, which allows for effective management of substantial traffic volumes. The model's ability to adjust to changing attack patterns is made possible by continuous learning mechanisms, which are integrated with current security frameworks to reinforce overall network defense. Together, these developments strengthen our DDoS detection system's efficacy and resilience, fortifying it against complex cyberattacks.

VIII. CONCLUSION

In conclusion, goal was to create an adaptive machine learning model capable of detecting DDoS attacks within IoT networks, overcoming the drawbacks of traditional detection methods. We accomplished this by utilizing the extensive NetBIOS dataset, which encompasses detailed network traffic and server logs. Through meticulous preprocessing steps such as normalization, feature extraction, and data balancing using techniques like SMOTE, we prepared the dataset for model training. We then implemented and assessed various machine learning models, including Logistic Regression (82.8%), Random Forest (100% with hyperparameter tuning), XGBoost (83%), K-Nearest Neighbors (99.5%), Support Vector Machine (100%), and Decision Trees (100% with hyperparameter tuning). These models exhibited high accuracy, with several achieving perfect scores after tuning. This adaptive methodology, which requires minimal human intervention, provides a robust and flexible solution for DDoS detection, greatly improving network security and reliability for both businesses and individuals and helping to prevent financial losses and service disruptions.

ACKNOWLEDGMENT

We wish to extend our deepest appreciation to Professor Shih Yu Chang, our esteemed instructor for the Machine Learning course, for his invaluable guidance and support throughout this project. We are also grateful to the University of New Brunswick, Canada, for providing the extensive NetBIOS dataset, which played a pivotal role in our research. Special thanks are also extended to our mentors and advisors for their indispensable assistance and advice. Furthermore, we acknowledge the collaborative efforts of our team members, whose dedication and expertise significantly contributed to the success of this project. Lastly, we express our heartfelt gratitude to our families and friends for their unwavering support and encouragement.

REFERENCES

[1] A. Elsaeidy, A. Jamalipour, and K. Munasinghe, "A Hybrid Deep Learning Approach for Replay and DDoS Attack Detection in a Smart City," IEEE Access, vol. PP, no. 1, pp. 1-1, 2021, doi: 10.1109/ACCESS.2021.3128701.

[2] R. Alzahrani and A. Alzahrani, "Security Analysis of DDoS Attacks Using Machine Learning Algorithms in Networks Traffic," Electronics, vol. 10, 2021, doi: 10.3390/electronics10232919.

[3] M. Saeed, H. Mohammed, O. Gazem, R. Saeed, H. Morei, A. Eidah, A. Gaid, A. Al-Uosfi, and M. Al-Madhagi, "Machine Learning Techniques for Detecting DDOS Attacks," in 2023, pp. 1-6, doi: 10.1109/eSmarTA59349.2023.10293366.

[4] M. Ahmed and A. N. Mahmood, "A survey of network anomaly detection techniques," Journal of Network and Computer Applications, vol. 60, pp. 19-31, 2016.

[5] H. Kim and H. Park, "A study on a DDoS detection model based on machine learning algorithms," International Journal of Security and Its Applications, vol. 9, no. 2, pp. 77-84, 2015.

[6] S. R. Sahoo and P. K. Pateriya, "Advanced machine learning techniques for DDoS attack detection in IoT environment," Journal of Information Security and Applications, vol. 41, pp. 105-111, 2018.

[7] N. Shone, T. N. Ngoc, and V. D. Phai, "A deep learning approach to network intrusion detection," IEEE Transactions on Emerging Topics in Computational Intelligence, vol. 2, no. 1, pp. 41-50, 2018.

[8] M. R. Mahdiani, E. Khamehchi, S. Hajirezaei, and A. Hemmati-Sarapardeh, "Modeling viscosity of crude oil using k-nearest neighbor algorithm," Adv. Geo-Energy Res., vol. 4, 2020, Art. no. 04.08. DOI: 10.46690/ager.2020.04.08.

[9] M. Y. Khan, A. Qayoom, M. Nizami, M. S. Siddiqui, S. Wasi, and K.-U.-R. R. Raazi, "Automated Prediction of Good Dictionary EXamples (GDEX): A Comprehensive Experiment with Distant Supervision, Machine Learning, and Word Embedding-Based Deep Learning Techniques," Complexity, vol. 2021, Art. no. 2553199, 2021. DOI: 10.1155/2021/2553199.

[10] Y. Zhang, Z. Pan, J. Zheng, L. Qian, and M. Li, "A hybrid ensemble method for pulsar candidate classification," Astrophys. Space Sci., vol. 364, 2019. DOI: 10.1007/s10509-019-3602-4.

[11] J. Cardoso-Fernandes, A. Teodoro, A. Lima, and E. Roda-Robles, "Semi-Automatization of Support Vector Machines to Map Lithium (Li) Bearing Pegmatites," Remote Sens., vol. 12, p. 2319, 2020. DOI: 10.3390/rs12142319.

[12] A. Chen, "Occupancy detection and prediction with sensors and online machine learning: Case study of the Elmia exhibition building in Jönköping," 2023.