Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 4, No. 3, April : 2023

DDOS ATTACK DETECTION USING SOFT VOTING CLASSIFIER

T. S. Savita Assistant Professor, Department of Computer Science, University College for Women (OU), Koti, Hyderabad, Telangana, India.
Dr. M. Raghavender Sharma Assistant Professor, Department of Statistics, University College of Science, Osmania University, Telangana, India.
Email: [1]id:savitats@rediffmail.com, [2]id:drmrsstatou@gmail.com

Abstract
A DDoS (Distributed Denial of Service) attack is an attempt to overwhelm a server or website with traffic from multiple sources, making it inaccessible to legitimate users. Detecting a DDoS attack is essential to prevent it from causing damage to the target system. By integrating the predictions of many different machine learning models, soft voting is a method that may be used to identify distributed denial of service (DDoS) assaults. This paper presents a DDoS attack detection model using soft voting classifier. The soft voting classifier combines AdaBoost Classifier, XGB Classifier and Extra Trees Classifier. The final categorization of the model should include as few erroneous "true positives" as possible. This should be the aim. As compared to employing a single classifier, soft voting gives results that are more accurate since it combines the predictions of numerous classifiers. This is because different classifiers may perform better on different parts of the data. Soft voting leverages the strengths of each classifier to improve overall accuracy.
Keywords: DDoS, soft voting, AdaBoost Classifier, XGB Classifier, Extra Trees Classifier

I. Introduction

A sort of cyberattack known as a distributed denial of service (DDoS) attack [1] involves an attacker trying to overload a website, server, or network with a large volume of traffic or requests coming from several sources, rendering the target system unreachable to its users. DDoS attacks can be launched using a variety of techniques and can range from simple to sophisticated [2].
These are the most typical kinds of DDoS attacks::

1. Volumetric attacks: The network or server that is the target of this kind of assault gets inundated with a significant volume of traffic, causing it to slow down or crash. These attacks are typically carried out using botnets, which are networks of compromised devices that can be remotely controlled by the attacker.
2. Protocol attacks: These attacks exploit weaknesses in the communication protocols used by the target system, such as the TCP/IP protocol, to overwhelm it with traffic. An example of this would be an attacker bombarding a target server with a huge number of SYN requests, which would cause the server to stop responding.
3. Application layer attacks: These kinds of assaults focus on the application layer of the system that is being attacked, such as the web server or the database server. They are intended to take advantage of weaknesses in the application code or the infrastructure, which will result in the target system being inaccessible or useless.

DDoS attacks can have serious consequences for businesses and organizations, including lost revenue, reputational damage, and legal liabilities [3]. Firewalls, Intrusion Detection and Prevention Systems (IDPS), and Content Delivery Networks (CDNs) are just some of the tools and strategies that businesses may employ to protect themselves against Distributed Denial of Service (DDoS) assaults [4]. It's important to have a proactive approach to security, including regular vulnerability testing and a comprehensive incident response plan.

DDoS attacks have been a persistent problem in recent times, with attackers using increasingly sophisticated techniques to carry out these attacks. Here are some of the key challenges and trends in DDoS attacks:

1. Increase in attack size: DDoS attacks have become larger and more complex in recent years, with some attacks exceeding hundreds of gigabits per second (Gbps). Attackers are using botnets made up of thousands or even millions of compromised devices to generate massive amounts of traffic.
2. Emergence of IoT botnets: The widespread use of Internet of Things (IoT) gadgets has resulted in the formation of massive botnets, which may then be utilised to launch distributed denial of service assaults. Attackers are targeting vulnerable IoT devices such as routers, cameras, and smart home appliances, which can be easily compromised and used as part of a botnet.
3. Increasing use of encrypted traffic: Attackers are increasingly using encrypted traffic to hide their attacks, making it difficult for security solutions to detect and mitigate them.
4. Shift to application layer attacks: Application layer attacks, which target the web applications and APIs of a target system, have become more prevalent in recent times. These assaults are more difficult to identify and defend against than standard volumetric attacks, and they have the potential to do severe harm to the system they are directed against.
5. Use of multi-vector attacks: Attackers are using multiple attack vectors in combination, such as combining volumetric and application layer attacks, to increase the effectiveness of their attacks and evade detection.

Attacks using DDoS continue to pose a substantial risk to companies and organisations [5], and require a multi-layered approach to defence that includes network infrastructure protection, application security, and proactive monitoring and response [6].

Machine learning can be a powerful tool for detecting DDoS attacks by analyzing patterns in network traffic and identifying anomalous behaviour [7]. By detecting and responding to DDoS attacks in real-time, machine learning can help protect organizations from the damaging effects of these types of cyber-attacks. Algorithms that use machine learning can do real-time analysis of network traffic, which enables them to swiftly identify and react to distributed denial of service assaults (DDoS) as they occur [8]. They can identify patterns of behavior in network traffic that are indicative of an attack, such as a sudden surge in traffic or unusual activity from specific IP addresses. It can adapt to new attack patterns and techniques by continuously learning from new data and updating its algorithms.

II. Literature

Tong Anh Tuan et al [9] presented the findings of a methodical examination into the use of machine learning algorithms for the detection of botnet-based distributed denial of service attacks, and analysed the data. In order to carry out the evaluation, the well-known publicity datasets UNBS-NB 15 as well as KDD99 that are used for the identification of botnet DDoS attacks are used. The research makes use of these different datasets. Machine learning techniques like Support Vector Machines (SVM), Artificial Neural Networks (ANN), Naive Bayes (NB), Decision Trees (DT), and Unsupervised Learning are utilised in order to investigate the accuracy, false alarm rate (FAR), sensitivity, specificity, false positive rate (FPR), area under the curve (AUC), as well as Matthews correlation coefficient (MCC) of datasets. Other machine learning techniques include area under the curve (AUC) and Matthews correlation coefficient (MCC) (USML).

Muhammad Aamir et al [10] suggested a method that uses feature engineering as well as machine learning in a systematic procedure in order to detect distributed denial-of-service attacks (DDoS). Utilizing feature selection strategies such like backward elimination, chi2 scores, as well as information gain scores are some examples of the techniques that are utilised in the pursuit of the objective of feature engineering, which is to produce datasets of multiple dimensions that contain important features. A number of supervised machine learning models are applied to the feature-

engineered datasets in order to highlight the flexibility of datasets for machine learning under optimal tuning of parameters within specified ranges of values. These sets of values have been determined in advance. These models are designed to illustrate how easily datasets may be adapted to new circumstances. According to the results, it should be possible to significantly cut the amount of attributes in order to make DDoS detection faster and more optimum with just a little effect on performance. This study aims to provide a framework at a strategic level that incorporates the main components of machine learning and feature engineering in combination with a specified flow of experimentation. The objective of this research is to give a framework at a strategic level. In addition, the models are verified using a process called cross-validation, and they are assessed using an area-under-curve method.

Alaeddine Mihoub et al [11] presented a study on the detection of denial of service and distributed denial of service attacks on the internet of things using machine learning techniques. The authors recommend adopting a new architecture that consists of two parts: detection of DoS/DDoS attacks and mitigation of their effects. The detection component provides detection with a high degree of granularity since it identifies the specific sort of attack that was carried out as well as the packet type that was utilised in the attack. This allows it to provide detection with a high level of accuracy. It will be possible, if things are handled in this way, to apply the appropriate mitigation countermeasure to certain packet types. In order to identify DoS and DDoS attacks, a multi-class classifier that employs the "Looking-Back" idea has been proposed as an essential component. After that, the Bot-IoT dataset is used to evaluate the performance of this classifier.

Swathi Sambangi et al [12] provided three incredibly important contributions to the discussion. (i) A unique gaussian-based traffic attribute-pattern similarity function for evolutionary feature clustering to achieve feature transition dimension reduction, (ii) A Gaussian-based network traffic similarity function for similarity computation between instances of network traffic, and finally (iii) A machine learning model known as SWASTHIKA that uses feature transformation traffic for the detection of low rate and high rate network attacks. The most recent benchmark dataset, which is known for its applicability in experimental research and goes by the name of the IoT DoS and DDoS attack dataset and can be found on the IEEE Dataport website. This is due to the fact that the dataset used to study IoT DoS and DDoS attacks comprises extremely non-linear traffic instances that are analogous to traffic in the real world. The performance of the proposed machine learning model SWASTHIKA is assessed by taking into consideration a range of classifier assessment metrics including metrics such as accuracy, precision, detection rate, and F-score.

Francesco Musumeci et al [13] We investigated at the prospect of using Artificial Intelligence and Machine Learning (ML) techniques in order to perform out automatic Distributed Denial of Service Attack Detection (DAD), with a specific focus on SYN flood assaults that target Transmission Control Protocol. The authors examine and compare two distinct DAD architectures, known as Standalone DAD as well as Correlated DAD, in which traffic feature collecting as well as attack detection are respectively tried to carry out locally at network switches as well as within a single entity. Standalone DAD and Correlated DAD are both known as Standalone DAD (such as in an SDN controller). They integrate the capabilities of ML and P4-enabled data planes in order to perform real-time DAD.

Qian Li et al [14] presented PCA-RNN is the revolutionary new framework to identify distributed denial of service attacks (Principal Component Analysis-Recurrent Neural Network). In order to have a comprehensive understanding of the traffic, the authors have chosen the bulk of the network parameters to characterise it. They make further use of the PCA approach to reduce the dimensionality of the features in order to cut down on the amount of time required for the detection process. PCA allows for a significant reduction in the amount of time required for prediction, all while maintaining the bulk of the data from the source. An RNN is fed data that has been pre-processed to minimise the dimensions of the data before it is used to train and generate a detection model.

Raj Kumar Batchu et al [15] designed a novel automated detection approach by narrowing the feature space, which in turn decreases the amount of time the model spends overfitting itself and the amount of computing work it must accomplish. The first step in increasing the generalizability of the model is to undertake data pre-processing. Then, feature selection is used to identify the qualities that are most important to the issue at hand, which adds to an improvement in the accuracy of the classification. This is done so that the problem at hand may be solved more effectively. In addition, the performance of the model may be enhanced by using hyperparameter tweaking, which involves selecting the appropriate parameters for different learning approaches. This can be done by determining the appropriate parameters for the model. At the last stage, the most useful features and hyperparameters are fed into a number of different supervised learning algorithms. These methods include logistic regression (LR), decision tree (DT), gradient boost (GB), k-nearest neighbour (KNN), as well as support vector machine (SVM) (SVM). The CICDDoS2019 dataset acts as the foundation upon which each and every one of these investigations is conducted.

Nisha Ahuja et al [16] presented a technology known as machine learning to differentiate between traffic that is benign and traffic that is part of a DDoS assault. The key contribution of this study is the identification of potential new features that might be used to detect distributed denial of service attacks. In order to generate the dataset, new characteristics are first entered into a CSV file. After this step, machine learning algorithms are trained on the dataset that was constructed using the SDN dataset. The vast bulk of the work that has been done up to this point for the purpose of detecting DDoS attacks has either employed a dataset that was not gathered from SDN or the research data has not been made accessible to the general public. Both of these scenarios are problematic. The categorization is carried out with the assistance of an innovative hybrid machine learning model.

Akshat Gaurav et al [17] focused the issues that potential company owners should be aware of before starting a firm, then introduce the filtering system that effectively detects DDoS attacks in the COVID-19 scenario, which is the focus of the study. The recommended method uses both machine learning and statistical methods to distinguish between data from DDoS attacks and communications from other sources.

M. Revathi et al [18] provided a discrete scalable memory driven support vector machine solution for DDoS attack plus SDN mitigation architecture for attack detection. The input data may be pre-processed using the Spark standardisation approach before the threat detection procedure even begins, replacing missing values and removing unnecessary data. After that, a method known as semantically multilinear component analysis is used in order to get the features extracted. A cutting-edge discrete scalable memory-based support vector machine (DSM-SVM) approach is used for the purpose of predicting the target, which is the responsibility of the classifier. Attack detection was followed by the mitigation procedure. In this step, the mitigation server may detect the danger by judiciously excluding harmful bot traffic and absorbing the remaining traffic. In this case, the recommended technique reduces attack traffic while maintaining benign traffic. For the KDD dataset, the authors examined the whole procedure. Before it was used in an environment for SDN threat detection and mitigation, the recommended network model underwent training as part of the assessment process.

Indraneel Sreeram et al [19] developed a quick and accurate application layer DDoS assault (App-DDOS Attack) detection method based on bio-inspired anomalies. The suggested methodology uses a bat algorithm with biological inspiration to provide quick and accurate App-DDOS using HTTP flood detection.

Sagar Pande et al [20] presented the WEKA tool was used for doing machine learning as the method for detecting DDoS assault, Ping of death was used as a method for conducting the attack. Throughout this experiment, the NSL-KDD dataset was used.

III. Soft voting based DDoS attack detection

Algorithms that employ machine learning may be put to use to detect distributed denial of service attacks by examining patterns in network traffic and locating unusual patterns of behaviour that are suggestive of an attack. The following is a list of some of the most important processes involved in using machine learning for the purpose of DDoS detection:

1. Data collection: The gathering of data on typical network traffic is the first stage involved in applying machine learning to the detection of DDoS attacks. This data may be used to teach machine learning algorithms to understand typical patterns of behaviour by providing examples of such patterns.
2. Feature extraction: Next, the relevant features of the network traffic are extracted, such as packet size, packet rate, and protocol type. These characteristics are then sent into the various machine learning algorithms as inputs.
3. Training the model: The machine learning algorithm is trained to detect regular patterns of behaviour in the network traffic by making use of the characteristics that were retrieved from the data. This involves using labeled data (i.e., data that is already known to be either normal or part of a DDoS attack) to teach the model what patterns of behavior to look for.
4. Testing the model: When the training process is complete, the model is evaluated using a fresh set of data to see how well it can identify DDoS assaults. This involves using unlabelled data (i.e., data that is not known to be normal or part of an attack) to evaluate the model's accuracy and effectiveness.
5. Deployment: When the machine learning algorithm has been tested and verified, it can then be deployed in a manufacturing environment to constantly monitor network traffic as well as identify DDoS assaults in real time. This is possible after the testing and validation processes have been completed.

3.1 Soft voting classifier

Soft voting is a method that is used in the field of machine learning to combine the predictions that are produced by several classifiers in order to get a single prediction that is aggregated. In soft voting, each classifier predicts a probability or likelihood of each class, rather than simply predicting the class itself. These probabilities are then averaged across all classifiers to produce a final prediction.

The process of using soft voting in a classifier involves the following steps:

1. Training multiple classifiers: First, multiple classifiers are trained on a dataset using different algorithms or parameter settings. Each classifier produces a probability estimate for each class, rather than just a binary prediction.
2. Aggregating probabilities: Once the classifiers are trained, their probability estimates are aggregated using soft voting. This involves computing the average probability estimate across all classifiers for each class.
3. Making a prediction: At last, the category for which the probability estimate is greatest is selected to serve as the conclusive forecast. If there is a tie, the class with the highest combined confidence across all classifiers is chosen.

The advantage of using a soft voting classifier is that it can improve the accuracy of predictions by combining the strengths of multiple classifiers. This can help compensate for weaknesses in individual classifiers and improve overall prediction performance. Additionally, soft voting can provide more nuanced predictions by taking into account the confidence levels of each classifier, rather than simply relying on binary predictions. The predictions of many classifiers may be combined using the helpful method of soft voting, which results in predictions that are both more accurate and more nuanced. In circumstances in which various classifiers have varying strengths and limitations, it might be very valuable to have this kind of information. Nevertheless, before putting soft voting classifiers into production, it is essential to thoroughly examine the performance of these tools and take into consideration the amount of computational complexity they provide.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 4, No. 3, April : 2023

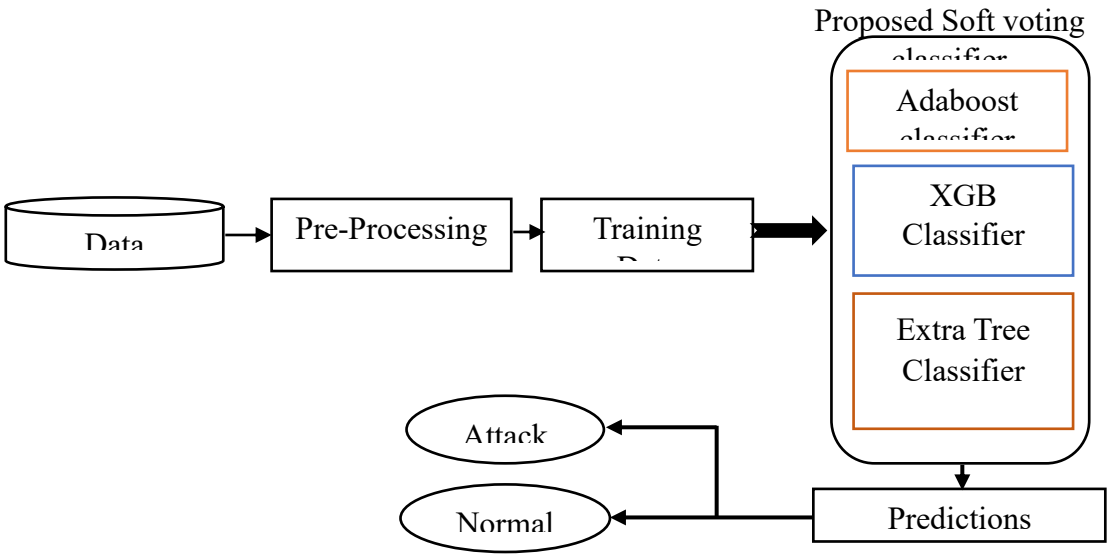

Figure 1: Proposed Soft voting classifier model

3.1.1 AdaBoost Classifier

A machine learning technique known as AdaBoost Classifier is used for a variety of classification jobs. The method achieves its results by assembling a collection of unreliable classifiers, which are then trained in a logical order and given a relative importance based on how well they function. The algorithm begins by training a base or weak classifier on the entire dataset. This weak classifier may be any classification algorithm that performs better than random chance. Once the weak classifier is trained, the algorithm evaluates its performance on the training set.

The AdaBoost Classifier algorithm then assigns higher weights to the misclassified examples and lower weights to the correctly classified examples. After then, it trains a second weak classifier using the exact same dataset, but this time it modifies the weights. This procedure is carried out in an iterative manner, for each new weak classifier being trained on a re-weighted dataset depending on the results obtained by the classifiers that came before it.

Step 1: Initialize sample weights

The AdaBoostClassifier algorithm begins by initializing the weights of all samples in the dataset to be equal. This means that initially, all samples are given equal importance in the classification process.

Step 2: Train a weak classifier

The full dataset is then used to train a simple classifier, which the algorithm does next. This weak classifier may be any binary classification system that performs better than random chance, such as support vector machines, decision trees, or logistic regression (SVMs).

Step 3: Evaluate the weak classifier

The algorithm then evaluates the performance of the weak classifier on the training set. The error rate is calculated by taking the total number of samples and dividing it by the number of samples that were incorrectly categorised.

Step 4: Update the weights of misclassified samples

The algorithm then changes the weights of the samples that were wrongly labelled so that they are higher than the weights of the samples that were correctly labelled. This is done to give the misclassified samples more importance in the next iteration of training.

Step 5: Update the weight of the weak classifier

The algorithm then calculates the weight of the weak classifier based on its error rate. A weak classifier with a lower error rate is given a higher weight, which means it will have more influence on the final classification result.

Step 6: Repeat steps 2-5

Steps 2-5 are repeated multiple times, with each iteration involving the following steps:

- Use the reweighted dataset to train a new, less accurate classifier.
- Evaluate the new weak classifier on the training set
- Update the weights of the samples that were wrongly labelled
- Update the weight of the weak classifier

Step 7: Combine the weak classifiers

When the training of all of the weak classifiers has been completed, the individual predictions from each of the weak classifiers are merged to provide a final prediction. This is done by taking a weighted sum of their predictions, where the weight of each weak classifier is proportional to its accuracy.

Step 8: Return the final prediction

The last step of the process involves the return of a prediction that is calculated using the aggregated results of all of the weak classifiers.

3.1.2 XGB Classifier

XGBClassifier is a machine learning algorithm used for classification tasks, which is based on the gradient boosting framework. It is an implementation of gradient boosting that uses a highly optimized tree-based algorithm. XGBClassifier is designed to be highly efficient, scalable, and accurate, and it has become a popular algorithm in machine learning competitions and real-world applications.

The XGBClassifier algorithm, along with its key components is discussed here:

1. Gradient Boosting Framework:

XGBClassifier is based on the gradient boosting framework, which involves iteratively adding weak learners to the model. It starts with a simple model and then adds more complex models iteratively to improve the model's performance. The primary objective of gradient boosting is to improve the accuracy of a model by fitting a new model to the residual errors of an older model. This will result in the model having a higher degree of precision.

2. Tree-based algorithm:

XGBClassifier uses a tree-based algorithm to create weak learners, where each tree represents a set of if-then rules that split the data into smaller groups. These splits are based on features that best separate the classes of the target variable. XGBClassifier uses a variant of decision trees called "CART" (Classification and Regression Trees) to create these splits.

Random Forest Classifier:

One kind of issues that may be solved with the help of the machine learning method known as random forest is regression and classification. On the other hand, for the sake of this lecture, we will be concentrating on how the random forest classifier works.

The random forest classifier is an example of an ensemble learning approach that generates a final prediction by combining the results of many decision trees. The purpose of this technique is to acquire a final result by first training a large number of decision trees on random subsets of the training data and then aggregating the results of those trees' predictions.

The operation of the random forest classifier may be broken down into two basic stages: the training stage, and the prediction stage.

Training phase:

1. Random selection of subsets: For each decision tree, the algorithm makes a randomised selection of a subset of the characteristics to use, as well as a portion of the training data. The term for this kind of sampling is termed bootstrap sampling.
2. Build decision trees: The method constructs numerous decision trees, each of which is trained on a different portion of the training data as well as a separate subset of the characteristics. The trees are then compared to one another and ranked according to their accuracy. A technique known as recursive partitioning is used to generate the trees. During this process, the algorithm continually cuts the data into more manageable groups depending on the values of the characteristics that are picked.

3. Voting mechanism: Once all the decision trees are built, they vote on the class label of each instance in the test data. After then, the forecast that stands is the one that received the most votes.

Prediction phase:

1. Input: An input instance is provided to the model.
2. Traverse trees: The input instance is passed down each decision tree, and the output of each tree is recorded.
3. Aggregate: The outputs from all the trees are then aggregated, and the final prediction is made.
3. Boosting:

XGBClassifier uses boosting to combine the predictions of multiple weak learners into a strong learner. With every iteration of the boosting procedure, XGBClassifier constructs a brand-new decision tree with the purpose of predicting the residuals of the trees that came before it. The residuals are the difference between the values that were predicted for the target variable and the values that were actually observed for that variable. The new tree helps to repair the mistakes made by the earlier trees by making predictions about the residuals, which also contributes to an overall improvement in the accuracy of the model.

4. Regularization:

In order to avoid the model from being too specific to its data and to enhance its capacity for generalisation, XGBClassifier incorporates a number of regularisation strategies. These techniques include:

- L1 and L2 regularization: penalizes large weights in the model to prevent overfitting.
- Max depth: restricts the amount of depth for every tree in the ensemble may have in order to avoid overfitting.
- Min child weight: defines the bare minimum of instances that must be present in each child node for the splitting process to proceed. This prevents the creation of child nodes with very few instances, which may lead to overfitting.

5. Optimization:

XGBClassifier is highly optimized to improve its efficiency and scalability. It uses a number of techniques to speed up the training process and reduce memory usage, such as:

- Approximate greedy algorithm: uses an approximate algorithm to find the best split points, which is faster than an exhaustive search.
- Parallel processing: uses parallel processing to train multiple trees simultaneously, which reduces training time.
- Out-of-core computing: allows XGBClassifier to train on datasets that are too large to fit into memory by reading data in chunks from disk.

3.1.3 Extra Tree Classifier

The Extra Trees Classifier is a modification of the Random Forest Classifier method that, in order to complete classification problems, makes use of a collection of decision trees. Like Random Forest Classifier, it generates multiple decision trees and aggregates the results of each tree to make a final prediction. However, Extra Trees Classifier has a few key differences in how it constructs its decision trees compared to Random Forest Classifier. the Extra Trees Classifier algorithm is described here:

1. Data preparation: The first stage, which is also the first step for other machine learning methods, is to prepare the data. This involves identifying key features or characteristics, cleaning and preparing the data, and separating the data into test and training sets.
2. Building the trees: The Extra Trees Classifier method constructs a group of decision trees after the data have been prepared. To create each decision tree, it randomly selects a subset of features and then randomly splits the data at each node. Unlike Random Forest Classifier, which selects the best split from among a random subset of features, Extra Trees Classifier

selects splits at random locations without searching for the best split. This makes the algorithm faster than Random Forest Classifier.

3. Making predictions: Once the ensemble of decision trees is built, the Extra Trees Classifier algorithm makes predictions by aggregating the results of each individual decision tree. When it comes to jobs involving categorization, the algorithm chooses the class label that obtains the most votes from the individual trees.

Here are the key equations used in the Extra Trees Classifier algorithm:

1. Random Split Selection: Extra Trees Classifier selects a random split at each node without searching for the best split. This is done using a random value within the range of the feature values, like so:
2. Aggregation: The predictions of each decision tree are aggregated to produce a final prediction. In classification tasks, the class label with the most votes is chosen. In regression tasks, the mean of the individual tree predictions is taken.

Extra Trees Classifier has some advantages over Random Forest Classifier, such as being faster to train and having a lower risk of overfitting due to its use of random splits. However, its lack of a search for the best split can also lead to less accurate models in some cases. Overall, Extra Trees Classifier is a useful algorithm for classification and regression tasks when speed is a priority and when overfitting is not a major concern.

IV. Experimental Analysis

The University of New Brunswick developed this dataset initially with the intention of doing research on DDoS data. This dataset derives all of its information from 2018. The dataset itself was derived from the logs of the university's servers, which revealed a number of different DoS assaults over the time for which the data was publicly accessible.

4.1 Evaluation parameters

Metrics like as accuracy, precision, recall, F1 score, as well as Cohen's kappa are often used in classification tasks to measure the performance of a machine learning approach. Other metrics include kappa. Here's an explanation of each metric, including its equation:

1. **Accuracy:** The percentage of properly identified samples compared to the overall amount of samples is the definition of accuracy. It determines how well the model can predict the class labels given the data.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

where TP = True Positive, TN = True Negative, FP = False Positive, and FN = False Negative.

2. **Precision:** The percentage of genuine positive predictions produced by a model relative to all positive predictions that the model generated is what precision measures. It demonstrates the model's ability to correctly identify the appropriate class.

$$\text{Precision} = TP / (TP + FP)$$

3. **Recall:** The percentage of genuine positive predictions made relative to the total number of positive samples in a dataset is what recall measures. It demonstrates how successfully the model is able to locate all relevant samples.

$$\text{Recall} = TP / (TP + FN)$$

4. **F1 Score:** The F1 score represents the mathematical middle ground between accuracy and recall. It offers a fair measurement of both measures and is helpful in situations in which the classes do not compare evenly.

$$\text{F1 score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

5. **Cohen's Kappa:** Cohen's kappa is a statistic that determines the degree to which two raters agree with one another. As used to the field of machine learning, this metric evaluates how

well the predicted labels match up with the actual ones. It runs from -1 to 1, where a score of 1 denotes complete agreement and a score of 0 denotes agreement that is no better than random chance.

Cohen's kappa = (Observed agreement - Expected agreement) / (1 - Expected agreement)
where Observed agreement = (TP + TN) / (TP + TN + FP + FN), and Expected agreement = ((TP + FP) / (TP + TN + FP + FN)) * ((TP + FN) / (TP + TN + FP + FN))

4.2 Data Pre-processing

The first step in processing the data is to remove any values that are not numbers and any rows that are duplicates. The processing that equalises the data is the most important phase. While doing a classification job, the term "data balance" refers to the process of making the amount of samples for each class the same. Imbalanced datasets can lead to biased models that perform poorly on the minority class.

The Synthetic Minority Over-Sampling Method, more often known as SMOTE, is a technique that is frequently used for balancing out unbalanced datasets. It generates synthetic samples for the minority class by interpolating new samples between existing ones.

Here's how SMOTE works:

1. Determine which underrepresented class samples need additional representation by oversampling.
2. Choose one of the examples from the minority class and the k people who live closest to it.
3. At random select one of the k closest neighbours, and then produce a new sample by interpolating between both the previously chosen sample and its neighbour.
4. Repeat steps 2 and 3 until the required quantity of synthetic samples has been produced.Add the synthetic samples to the dataset, making sure to keep the original samples.
5. Repeat steps 1-5 until the minority class is balanced with the majority class.

By generating synthetic samples, SMOTE helps to create a balanced dataset without the need to collect more data. It is possible for it to increase the performance of a machine learning system by decreasing the bias towards the class with the bulk of the data and increasing the variety of the data used for training.

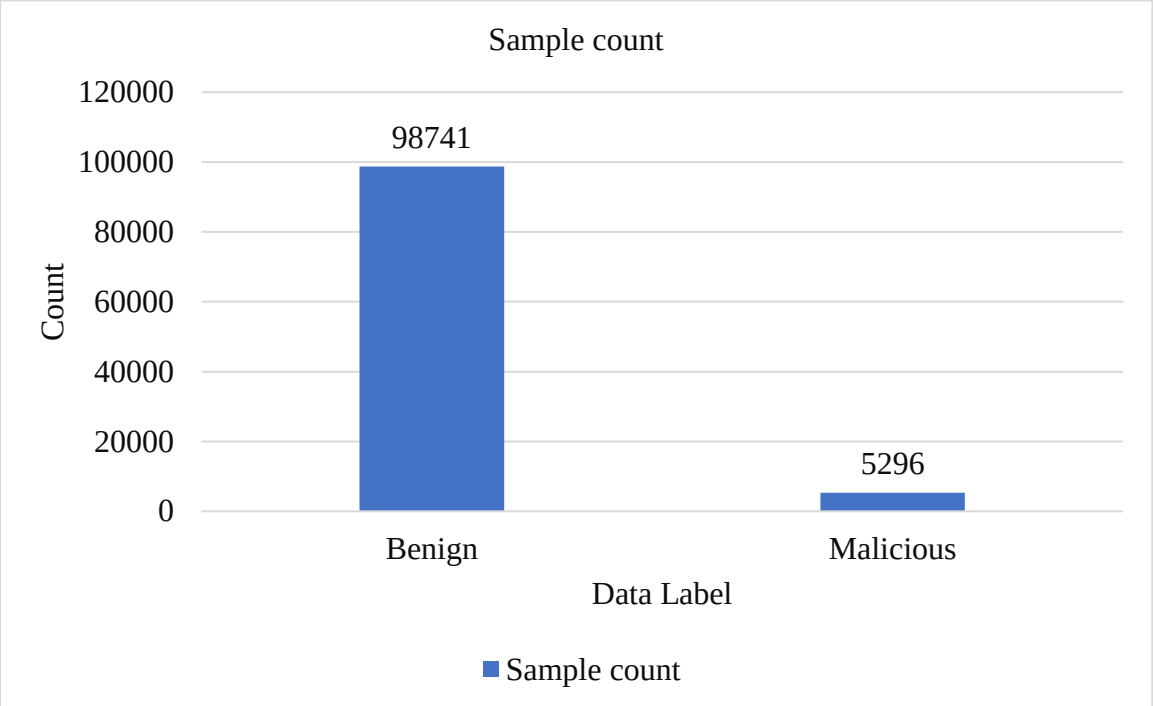

Figure 2: Sample Count Before pre processing

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 4, No. 3, April : 2023

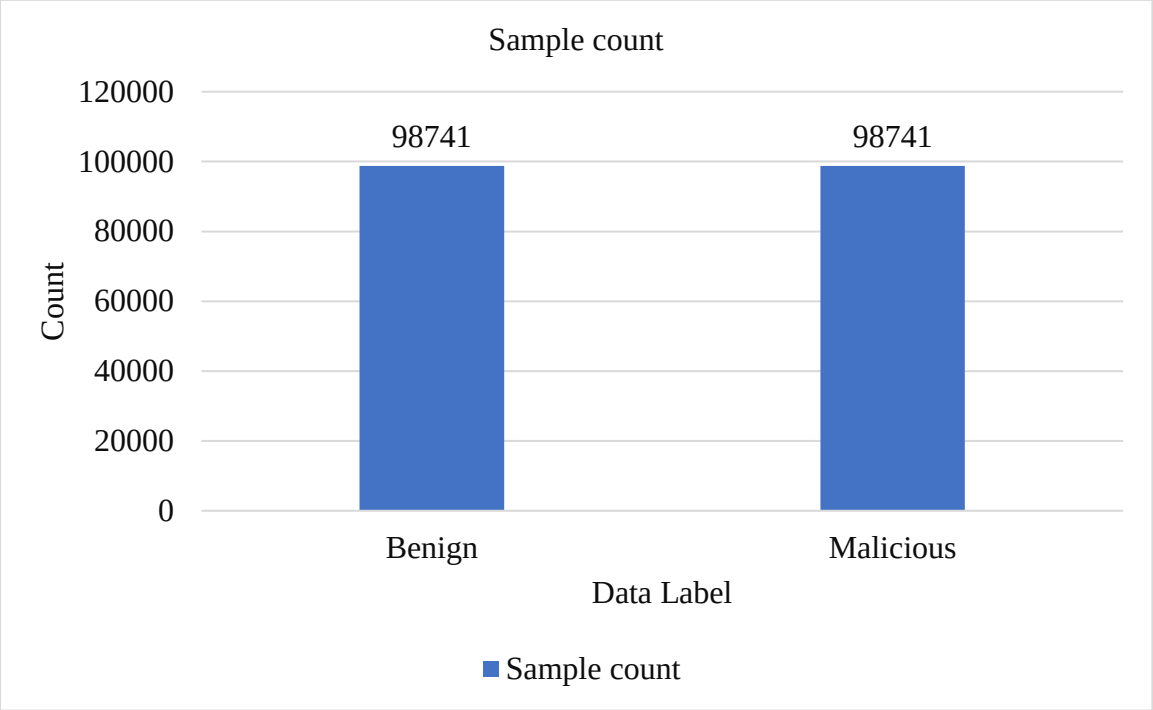

Figure 3: Sample Count After pre processing

Table 1: Evaluation parameters

Accuracy	0.999975
Precision	0.999975
Recall	0.999975
F1 score	0.999975
Cohens kappa	0.999949

The evaluation parameters that were acquired by the proposed model are shown in Table 1. There is a 99.99% chance that the suggested model is accurate. It has been determined that the suggested model has an accuracy of 0.999975. The suggested model has an impressive recall rate of 0.999975. The suggested model has a score of 0.999975 on the F1 scale. Cohen's kappa is calculated to be 0.999949. In general, a classification or prediction model with superior performance will have higher values of precision and recall, as well as a higher F1 score and kappa value from Cohen.

- Precision: If the model has a higher accuracy score, it means that it is more accurate at predicting genuine positives and has less instances of false positives.
- Recall: A higher recall score shows that the model is better at discovering all of the genuine positives and also has fewer false negatives. This is because the model has fewer false positives than it does false negatives.
- F1 score: When the F1 score is greater, it suggests that the model is both precise and recallable to a high degree.
- Cohen's kappa: When the Kappa score between the annotators or between an annotator and a model is greater, this implies that the annotators are more in agreement with one another or with the model. It is often used in situations where the class distribution is imbalanced.

Table 2: Confusion matrix

	Benign	Malicious
Benign	79002	3
Malicious	1	78962

There have been just 4 instances of false positives. There were 79005 benign samples and 78963 malicious samples out of the total number of samples tested. A lower false positive rate is helpful in a classification system because it reduces the number of false positive predictions, which means the

system makes fewer incorrect positive predictions. When a model incorrectly predicts a positive result when the actual outcome is negative, this is referred to as a false positive. Reducing the false positive rate is especially important in situations where the cost of a false positive is high. In attack detection systems, a false positive could result in a genuine request being flagged as attack, which could harm the customer's reputation or cause inconvenience.

Table 3: Comparative analysis

	False Positives
Logistic Regression [20]	191
K Nearest Neighbour [21]	16
Support Vector Machine [22]	59
Random Forest Classifier [23]	13
Gradient Tree Boosting [24]	436
Light Gradient Boosting [25]	21
Proposed Soft Voting model	4

The comparative study of the proposed model with the various current methodologies is shown in Table 3. The false positives produced by logistic regression is 20. The false positives produced by K Nearest Neighbour is 16. The false positives produced by Support Vector Machine is 59. The false positives produced by Random Forest Classifier is 13. The false positives produced by Gradient Tree Boosting is 436. The false positives produced by Light Gradient Boosting is 21. The proposed soft voting classifier produced only 4 false positives.

V. Conclusion

An effort to overload a server or website with traffic from various sources is what's known as a DDoS assault, which stands for "distributed denial of service." Those who are permitted to use the server or website will be unable to access it as a result of this action. It is essential to identify an attack that uses distributed denial of service in order to prevent the attack from causing damage to the system that is being targeted. Soft voting is a technique that may be used to detect distributed denial of service (DDoS) attacks. This approach works by merging the predictions that have been generated by a wide variety of machine learning models. The objective of this study is to develop a model for the identification of DDoS attacks that makes use of a voting-based classification system. The AdaBoost Classifier, the XGB Classifier, and the Extra Trees Classifier are all combined to create the soft voting classifier. The final categorization of the model should include as few erroneous "true positives" as possible. This should be the aim. As compared to employing a single classifier, soft voting gives results that are more accurate since it combines the predictions of numerous classifiers. This is due to the fact that several classifiers may have greater performance on various subsets of the data. The total accuracy is improved by the use of soft voting, which plays to the strengths of each classifier.

References

[1] ur Rehman, Saif, Mubashir Khaliq, Syed Ibrahim Imtiaz, Aamir Rasool, Muhammad Shafiq, Abdul Rehman Javed, Zunera Jalil, and Ali Kashif Bashir. "DIDDOS: An approach for detection and identification of Distributed Denial of Service (DDoS) cyberattacks using Gated Recurrent Units (GRU)." *Future Generation Computer Systems* 118 (2021): 453-466.

[2] Vishwakarma, Ruchi, and Ankit Kumar Jain. "A survey of DDoS attacking techniques and defence mechanisms in the IoT network." *Telecommunication systems* 73, no. 1 (2020): 3-25.

[3] Lee, In. "Cybersecurity: Risk management framework and investment cost analysis." *Business Horizons* 64, no. 5 (2021): 659-671.

[4] Li, Zihao, and Weizhi Meng. "Mind the amplification: cracking content delivery networks via DDoS attacks." In *Wireless Algorithms, Systems, and Applications: 16th International*

Conference, WASA 2021, Nanjing, China, June 25–27, 2021, Proceedings, Part II 16, pp. 186-197. Springer International Publishing, 2021.

[5] Franco, Muriel, Erion Sula, Bruno Rodrigues, Eder Scheid, and Burkhard Stiller. "ProtectDDoS: a platform for trustworthy offering and recommendation of protections." In *Economics of Grids, Clouds, Systems, and Services: 17th International Conference, GECON 2020, Izola, Slovenia, September 15–17, 2020, Revised Selected Papers 17*, pp. 28-40. Springer International Publishing, 2020.

[6] Umer, Muhammad Azmi, Khurum Nazir Junejo, Muhammad Taha Jilani, and Aditya P. Mathur. "Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations." *International Journal of Critical Infrastructure Protection* (2022): 100516.

[7] Das, Saikat, Deepak Venugopal, and Sajjan Shiva. "A holistic approach for detecting ddos attacks by using ensemble unsupervised machine learning." In *Advances in Information and Communication: Proceedings of the 2020 Future of Information and Communication Conference (FICC), Volume 2*, pp. 721-738. Springer International Publishing, 2020.

[8] Gaur, Vimal, and Rajneesh Kumar. "Analysis of machine learning classifiers for early detection of DDoS attacks on IoT devices." *Arabian Journal for Science and Engineering* 47, no. 2 (2022): 1353-1374.

[9] Tuan, Tong Anh, Hoang Viet Long, Le Hoang Son, Raghvendra Kumar, Ishaani Priyadarshini, and Nguyen Thi Kim Son. "Performance evaluation of Botnet DDoS attack detection using machine learning." *Evolutionary Intelligence* 13 (2020): 283-294.

[10] Aamir, Muhammad, and Syed Mustafa Ali Zaidi. "DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation." *International Journal of Information Security* 18 (2019): 761-785.

[11] Mihoub, Alaeddine, Ouissem Ben Fredj, Omar Cheikhrouhou, Abdelouahid Derhab, and Moez Krichen. "Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques." *Computers & Electrical Engineering* 98 (2022): 107716.

[12] Sambangi, Swathi, Lakshmeeswari Gondi, and Shadi Aljawarneh. "A feature similarity machine learning model for ddos attack detection in modern network environments for industry 4.0." *Computers and Electrical Engineering* 100 (2022): 107955.

[13] Musumeci, Francesco, Ali Can Fidanci, Francesco Paolucci, Filippo Cugini, and Massimo Tornatore. "Machine-learning-enabled DDoS attacks detection in P4 programmable networks." *Journal of Network and Systems Management* 30 (2022): 1-27.

[14] Li, Qian, Linhai Meng, Yuan Zhang, and Jinyao Yan. "DDoS attacks detection using machine learning algorithms." In *Digital TV and Multimedia Communication: 15th International Forum, IFTC 2018, Shanghai, China, September 20–21, 2018, Revised Selected Papers 15*, pp. 205-216. Springer Singapore, 2019.

[15] Batchu, Raj Kumar, and Hari Seetha. "A generalized machine learning model for DDoS attacks detection using hybrid feature selection and hyperparameter tuning." *Computer Networks* 200 (2021): 108498.

[16] Ahuja, Nisha, Gaurav Singal, Debajyoti Mukhopadhyay, and Neeraj Kumar. "Automated DDOS attack detection in software defined networking." *Journal of Network and Computer Applications* 187 (2021): 103108.

[17] Gaurav, Akshat, Brij B. Gupta, and Prabin Kumar Panigrahi. "A novel approach for DDoS attacks detection in COVID-19 scenario for small entrepreneurs." *Technological Forecasting and Social Change* 177 (2022): 121554.

[18] Revathi, M., V. V. Ramalingam, and B. Amutha. "A machine learning based detection and mitigation of the DDOS attack by using SDN controller framework." *Wireless Personal Communications* (2021): 1-25.

[19] Sreeram, Indraneel, and Venkata Praveen Kumar Vuppala. "HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm." *Applied computing and informatics* 15, no. 1 (2019): 59-66.

[20] Yadav, Satyajit, and S. Selvakumar. "Detection of application layer DDoS attack by modeling user behavior using logistic regression." In 2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions), pp. 1-6. IEEE, 2015.

[21] Dong, Shi, and Mudar Sarem. "DDoS attack detection method based on improved KNN with the degree of DDoS attack in software-defined networks." *IEEE Access* 8 (2019): 5039-5048.

[22] Mehr, Shideh Yavary, and Byrav Ramamurthy. "An SVM based DDoS attack detection method for Ryu SDN controller." In *Proceedings of the 15th international conference on emerging networking experiments and technologies*, pp. 72-73. 2019.

[23] Najar, Ashfaq Ahmad, and S. Manohar Naik. "DDoS attack detection using MLP and Random Forest Algorithms." *International Journal of Information Technology* 14, no. 5 (2022): 2317-2327.

[24] Verma, Parag, Shayan Anwar, Shadab Khan, and Sunil B. Mane. "Network intrusion detection using clustering and gradient boosting." In *2018 9th International conference on computing, communication and networking technologies (ICCCNT)*, pp. 1-7. IEEE, 2018.

[25] Khafajeh, H. A. Y. E. L. "An efficient intrusion detection approach using light gradient boosting." *Journal of Theoretical and Applied Information Technology* 98, no. 5 (2020): 825-835.