IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# BOTNET DETECTION USING MACHINE LEARNING

[1]Mr. Ramesh Singh Rawat, [2]Ashutosh Bhardwaj, [3]Rohan Saini, [4]Shaurya Gularia, [5]Sagar Bisht

[1]Assistant Professor, [2]Student, [3]Student, [4]Student, [5]Student

Computer Science and Technology,

Graphic Era Deemed to be University, Dehradun, India

***Abstract:*** Identifying and neutralizing malicious activities within network traffic is crucial in cybersecurity, and botnet detection is key in this effort. This study assesses the performance of various machine learning algorithms in detecting botnets, using the CTU-13 dataset. This dataset is notably imbalanced, with background and normal traffic significantly outweighing botnet traffic. To address this challenge, algorithms such as CNN-LSTM, XGBoost, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) were tested. Results show that XGBoost achieved the highest accuracy among the tested models. By employing undersampling techniques, accuracy improved to 80%, despite the data imbalance. In contrast, oversampling methods were less effective. This research underscores XGBoost's potential for handling imbalanced datasets and its superior performance in botnet detection. Future studies might aim to enhance the model's robustness and explore further methods to manage data imbalance effectively.

## I. INTRODUCTION

The innovation and development in the field of Internet and technologies also increases the threats to security and violates the CIA goals (Confidentiality, Integrity,Availability) of individuals or organizations. These threads are because of the vulnerability of the system that the user or the network of organization , or the malicious files are also the main source of malware. These malicious files enter the system through email, when downloading files from unknown source , pirated softwares. Even from known and genuine sources also. Cybersecurity firm Kaspersky has figured out that malware was downloaded more than 600 millions times in 2023 from Google play store. Initially cybercriminals upload innocent apps to the store. Then, when the app has built an audience [7]. It is elevated with malicious functionality in its next update.The more devices connecting to the internet the more malicious software spreading over the internet. Not only commercial and non-commercial organizations but there is also rapid growth in attacks in government agencies.

Botnet is among these malicious software which is a remotely controlled network of computers almost hijacked systems. Botnet is a kind of organized army of zombies which attackers use for malicious purposes. This process is done by the entity called botmaster or Botherder through command and control server. A botmaster is responsible for initializing, coordinating,managing , or aborting an attack on an infected system. World has witnessed so many botnets and their attacks. Each of these attacks cause millions of dollars of damage to target organizations. For example, Zeus was the botnet which targets window machines [3]. It had malicious code that targeted banking credentials. It was separated through drive-by downloading or phisingattack.In 2012 US Marshals took down the botnet, However original creators took part of their earlier creation and revived it as gameover zeus, which was taken down by FBI but its creators have built bot network again. Another one is Cutwail. This botnet had an army of up to 2 million computers in 2009,sending a million spam emails per minute. Researchers claimed success to beat Cutwail control server, almost two third, but still not gone.

Basically a botnet is used or acts as a source for launching worms or viruses with the help of bot armies. These bots increase the attacker's ability to carry out large attacks. Individual or small teams of attackers can collectively use the hijacked system or affected system army. Distributed power resources are the main power of botnets [1]. Through growth in technologies every system has a high GPU CPU and bandwidth capacity. Every system in the botnet army made the botnet more powerful. In order to be a victim device. Botmasters go through various phases. Initially attackers find the vulnerability of the target system in order to deliver the malicious code using File Transfer Protocol(FTP), HyperText Transfer Protocol(HTTP), Peer to Peer(P2p)[1]. After successfully injecting the malicious code, the victim device connected with the command and Control server C&C [10]. Once the device is injected with malicious code, it becomes part of the zombie army or bot. Botmaster send commands to bot or zombie army through the Command and Control server and utilized power of these bot to perform various malicious activities using the command received be the Command and Control server including denial-of-service(DDOS), Span generation, Key logging, Identity theft, Exploiting private documents, Sniffing traffic [2]. Botmaster maintains the update of the bots from time to time.

Botnets have various ways to control their network inorder to command enormously. In this paper we discuss the Three architecture of botnet networks.

*A. Centralized Architecture:* This architecture is the oldest and easiest way to control and manage by the botmaster. There is a central point called command and control server (C&C Server) which is used to manage and control all the bots or zombie armies. Internet Relay Chat (IRC) and Hypertext transfer protocol [2] are the major protocols used in centralized Architecture using which bots establish a strong communication channel between one or multiple connection points. Centralized Architecture is easy to set up as it does not require any special hardware [2]. The C&C server directly communicates with bots and provides fast response. Also enhance management and monitoring of botnets.

*B. Multi Server or Peer to Peer Architecture:* In this there is no central entity responsible for controlling bots. There is no specific command and controlled server. All the bots act as a command and control server, also the client. The botmaster sends commands to one or more bots and these bots transmit commands to their neighboring bots. It is based on peer-to-peer protocol. Hence, tough to deal with [2]. To build a botnet based on this architecture, we need to find and capture the bots which are working together. As there is no main control center. It becomes tricky to figure out how big the problem is and make it hard to catch all of them. Bots are loosely connected,they are not too dependent on each other. This makes it tough to stop at once. Attackers use secure HTTPS inorder to hide malicious code from firewalls or filters [3]. Compared with centralized architecture these architectures are more complex to set up and require a high level of expertise and don't have a single point of failure, they are slow, hard to handle and not easily scalable.

*C. Hybrid Architecture:* Hybrid Architecture combined the advantage of both centralized and Multi server architecture. Some bots connect with the central server while some others work as peer to peer, communicating directly with each other [6]. Each above architecture has its own challenges [5]. For example, the centralized architecture is easy to set up but has a single point of failure whereas Peer to peer is more reliable but needs more expertise to handle.

## II. PROBLEM STATEMENT

The rapid development of networked devices have led to a significant increase in cyber threats, and botnets have become a significant and sufficient fear of violence. Botnets are networks of infected devices controlled by criminals that can carry out coordinated actions, including distributed denial of service (DDoS) attacks, data breaches and spam campaigns, causing serious damage and impacting people, posing serious threats to organizations and critical systems. These methods often fail to detect new or polymorphic threats and can produce false positives, resulting in wasted resources and reduced confidence in security. It provides a great way to improve the accuracy and efficiency of botnet detection. Machine learning models can analyze large amounts of network data, identify patterns that indicate botnet activity, and adapt to emerging threats through continuous learning. However, the use of efficient machine-based botnet detection faces many challenges, such as needing large datasets for training, selecting appropriate features, and handling inconsistent information. Address this challenge by developing and evaluating machine learning models to detect botnet activity on network connections. The main objectives of this study are: Discover and analyze the most

important features that distinguish botnet traffic from legitimate traffic.Efficiently build models of disparate data, including real and synthetic data, to ensure they are robust and general. A comprehensive approach to machine learning-based botnet detection integrated into existing network security. understanding and tools. Through this research, we aim to advance the state of the art in botnet detection, providing insights and tools that can enhance the security of modern networks against botnet threats.

## III. LITERATURE REVIEW

In recent years, researchers working on network security have been really focused on finding and keeping track of botnets. Some researchers use what they called HoneySpots or HoneyNet, an active analysis while another setup was based on passive network monitoring and analysis. Feily M in 2009 categorized botnet detection techniques in four categories:signature-based,DNS-based,mining-based, and anomaly-based. Signature- based method focuses on identifying botnets using predefined pattern known malicious activity . However, it failed to detect the new botnet [7]. DNS based techniques depend on monitoring Domain name System traffic to detect botnet activities. Mining based involves using data mining and analysis methods to find patterns or anomalies indicating botnet behavior and Anomaly based analysis network behavior and identify anomalies. They also propose charts to showcase the performance of these techniques.

Singh et al. extensively reviewed IoT botnet detection methods utilizing Domain Name Space detection [8]. Their study proposed a new framework for classifying botnet detection methods reliant on DNS and provides an in-depth analysis of each technique.

Koroniotis et al. conducted a survey on present deep learning techniques and forensic tools used to detect botnets in IoT systems, addressing their related challenges [10]. Additionally, they also explored the application of deep learning algorithms in the field of network forensics.

Haddadi and Nur have proposed five different botnet detection techniques. Three were machine learning based and the rest two were rule-based systems. The authors analyzed their proposed approaches with different scenarios on twenty Five publicly available datasets and find the Tranalyzer-2 flow-based system and FlowAF performed better than the other three systems[9].

Wang S, and team purpose a method to detect malicious android apps. They examined mobile traffic. They treat each set of data moving through HTTP as a Document. Then they broke down these documents into parts using N-gram generation to find important features and use SVM (Support vector machine) learning algorithm for validation. Their finding shows accuration of 99.15% [1]. However, in real time it could spot them only about half the time (54.81%) [3].

Hossein and team proposed a framework based on passively monitoring network framework which consists of four main components: Filtering, Application Classifier, Traffic Monitoring, Malicious Activity Detector. Filtering used to filter out the irrelevant traffic to reduce the traffic workload.Application classifier separates IRC and HTTP traffic from the rest of traffic [8]. Malicious activity detector is responsible for analyzing the traffic to detect malicious activities. And Traffic Monitoring is responsible to detect the hosts that have similar behavior and communication patterns.

Kirubavathi and Anitha conduct research in android botnet detection. They gather a dataset by collecting 9756 botnets from different repositories and 122178 benign applications from open-source platforms [7]. They proposed a structural analysis-based learn-ing framework for android botnet detection consisting of five major parts: Collect, Analyze, Identify, Classify. They used Naive Bayesian, Support vector machine, Reduced error pruning tree. Tests on real-world benchmark datasets demonstrate that the chosen patterns yield superior detection accuracy in contrast to similar detection methods.

## IV. PROPOSED METHODOLOGY

### *A. Dataset Description*

Getting a good dataset is really hard. Most new ways to find botnets make their own sets of data to test how well models work. However,Making these sets is tough and not a good idea. They might suffer from important labels or real world traffic. Our goal is to create a Machine Learning Model that can recognize the botnets. To train our program, we're using real data called CTU-13, which contains real botnet traffic [6]. Garcia et al. (2014) created the CTU-13 dataset.The dataset is labeled in a flow by flow basis, consisting in one of the largest and more labeled botnet datasets available.

| Feature | Type |
|---|---|
| Start time | Numerical |
| End time | Numerical |
| Duration | Numerical |
| Protocol | Categorical |
| Src IP address | Categorical |
| Src port number | Categorical |
| Direction | Categorical |
| Dst IP address | Categorical |
| Dst port number | Categorical |
| Flags | Categorical |
| Type of services | Categorical |
| Number of packets | Numerical |
| Number of bytes | Numerical |
| Number of rows | Numerical |
| Label | Categorical |

Data features in the CTU-13 dataset (Garcia et al., 2014)

The Primary goal of Garcia was Authenticity, Diversity, Clarity, Variety, Synchronization, Privacy. The CUT-13 dataset was captured in the CTU University, the Czech Republic. They set up virtual machines running Windows XP on a main Linux computer.They connected these virtual machines to the university network and deliberately infected them with specific botnets. Then, they recorded the network activity using software called *tcpdump* on both the Linux host and one of the university's routers. After collecting data for 13 different infection cases, they organized this information into CSV files in NetFlow file format. Each data is divided into Botnet, Normal, or Background. Botnet and Normal referred to their test environment, while Background meant data from the university network. They basically perform different malicious activities such as DDoS, port scanning, click fraud, spamming,etc. Table summarizes the characteristics of the botnet scenarios.ario.

| ID | Duration (hrs) | #Flows | Bot | #Bots | Activity |
|---|---|---|---|---|---|
| 1 | 6.15 | 2,824,637 | Neris | 1 | IRC, SPAM, CF |
| 2 | 4.21 | 1,808.123 | Neris | 1 | IRC, SPAM, CF |
| 3 | 66.85 | 4,710,639 | Rbot | 1 | IRC, PS |
| 4 | 4.21 | 1,121,077 | Rbot | 1 | IRC, DDoS |
| 5 | 1.63 | 129,833 | Virut | 1 | SPAM, PS |

| | | | | | |
|---|---|---|---|---|---|
| 6 | 2.18 | 558,920 | Menti | 1 | PS |
| 7 | 0.38 | 114,078 | Sogou | 1 | HTTP |
| 8 | 19.5 | 2,954,231 | Murlo | 1 | PS |
| 9 | 5.18 | 2,753,885 | Neris | 10 | IRC, SPAM, CF, PS |
| 10 | 4.75 | 1,309,792 | Rbot | 10 | IRC, DdoS |
| 11 | 0.26 | 107.252 | Rbot | 3 | IRC, DdoS |
| 12 | 1.21 | 325,472 | NSIS.ay | 3 | IRC, P2P |
| 13 | 16.36 | 1,925,150 | Virut | 1 | HTTP, SPAM, PS |

CTU-13 Dataset Description

*B. Preprocessing*

CTU-13 Dataset contains an integer, float, object and categorical columns. Columns like Start Time, Source and destination IP address, and source and destination port have a large cardinality, and columns like sTos and dTos have a very low cardinality. In order to address these issues, preprocessing needs to be done on the CTU-13 dataset to make it compatible with machine learning training and prediction. We have merged all 13 scenarios.

1. One Hot Encoding

   A column with low cardinality that consists of categories is called a categorical column. There are four columns in the CTU-13 Dataset that have been designated as categorical columns: "Dir," "Proto," "sTos," and "dTos." There are seven categories in the "Dir" column, fifteen in the "Proto" column, six in the "sTos" column, and five in the "dTos" column. The process of transforming categorical columns into vectors of 0s and 1s is known as "one hot encoding." The vector lengths of two and three classes in a column will be two and three, respectively.

2. Label Encoding

   In the CTU-13 Dataset, the Label column is the target column. Three classes—background, botnet, and normal—can be applied to the Label column. In order for any machine learning model to function correctly, all of the response and predictor variables must be numerical. The Label columns must be changed to numbers in order to achieve this.The process of giving those string categories labels that begin with 0 is known as encoding. Background, botnet, and normal labels were mapped into 0, 1, and 2, respectively, in the CTU-13 Dataset.

3. Dropping Columns

   Columns with very high cardinality and columns that cannot be converted into any numeric values can be dropped. In CTU-13 Dataset, columns like ‘StartTime’, ‘SrcAddr’, ‘Sport’, ‘DstAddr’, ‘Dport’ and ‘State’ were dropped to reduce some of the features in the dataset.

*C. Feature selection and Dimensionality reduction*

The number of features present in the initial CTU-13 dataset was 15 features. After performing one-hot encoding and dropping of irrelevant columns which basically affects the dataset, the number of columns increased from 15 to 32. Training a machine learning model directly on all the 32 columns might not give the best results. Fundamentally, there are three problems associated with having a multitude of columns: machine learning model training time increases, redundant memory resource allocation, and multiple features often tend to confuse the model. The world of machine learning often suffers from the problem of the curse of dimensionality, and a garbage input to the model will always give garbage output. To deal with such issues of dimensionality, feature selection, also called attribute selection or variable selection, is taken into

consideration. The following subsections give a deep dive into feature selection techniques that were considered for obtaining optimal results.

1. Removing Null Columns

The simplest and easiest way of getting rid of redundant columns is by dropping columns that have many null values. In the case of the CTU-13 dataset, the columns exhibiting null values were 'Sport', 'Dport', 'State', 'sTos' and 'dTos'. Columns namely 'Sport', 'Dport' and 'State' were dropped because of their nature that they could not be converted to numeric values. 'sTos' and 'dTos' were median imputed for the null values. 'dTos' column as seen in the next variance thresholding section was dropped.

2. Variance Thresholding

The concept of variance thresholding stems from the fact that the columns that have the same value or little difference in values do not contribute to response prediction. Technically as suggested in [19], a column with low variance or zero variance can be discarded. The default behavior of variance thresholding function from the sklearn library is to keep columns with nonzero variance. For the CTU-13 dataset, a threshold of 0.9 was given to discard columns exhibiting variance less than that which resulted in a significant reduction of feature columns. All the columns of the CTU-13 dataset displayed some range of variance ranging from 0 to 1. The only column that displayed close to zero variance was the 'dTos' column as 99.5% of the data belonged to one value and remaining to other values as shown in Figure 1.

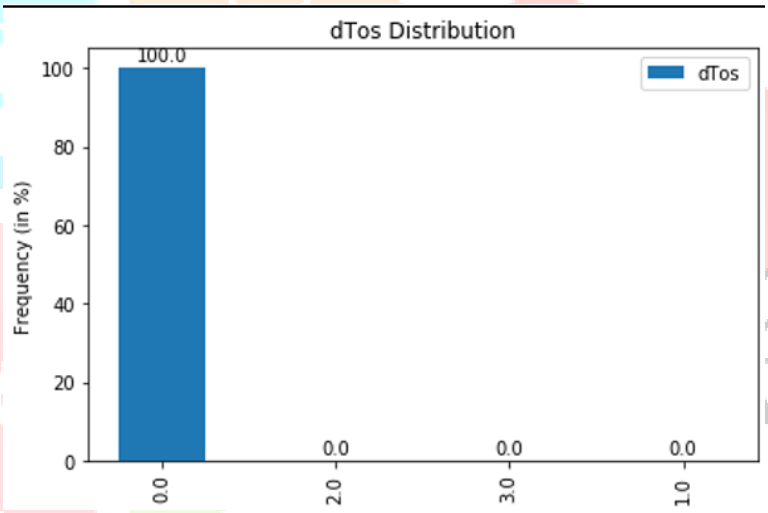


*Figure 1: Variance Thresholding*

3. *Feature importance and Correlation Heatmap*

Feature importance is applicable to any machine learning model that has the feature_importance as a parameter. The correlation coefficient involves understanding correlation. A covariance gives a nice description of how features vary with each other. Mathematical expression for covariance between feature $x$ and $y$ is given by $\sigma xy = \sum(x-\bar{x})(y-\bar{y})\ N$ where $\bar{x}$and $\bar{y}$ are the mean of the x and y feature. The correlation coefficient is given by $\rho xy = \sigma xy\ \sigma x \sigma y$ where $\sigma x$ and $\sigma y$ are the variance of $x$ and $y$ respectively. Also, the correlation matrix depicts the correlation between each input feature which ranges from -1 to +1. A value of 0 means there is no correlation and otherwise, a correlation exists. For features that are highly correlated with each other, one of the features can be dropped from the final feature selection set.

The correlation matrix in Figure 8 indicates that the columns 'TotBytes' and 'Totpkts' are highly correlated with a value of 0.99. Similarly, 'SrcBytes' is positively correlated with 'TotPkts' and 'TotBytes'.

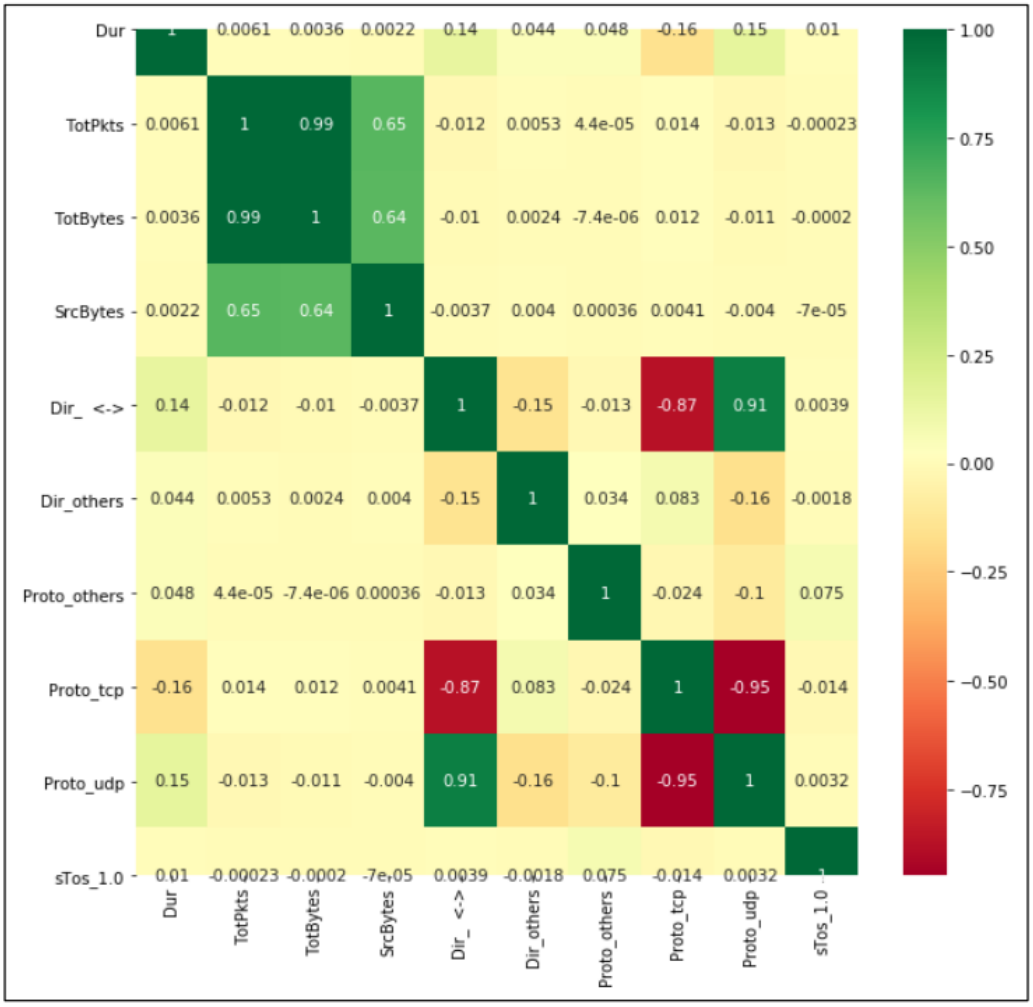


*Figure 2: Correlation Heatmap Matrix*

*D. Addressing Data Imbalance*

1. *Undersampling*

Under-sampling can be used to lower the number of samples from the majority class to equalize it with samples from minority classes when the number of samples from the majority class is comparatively high.

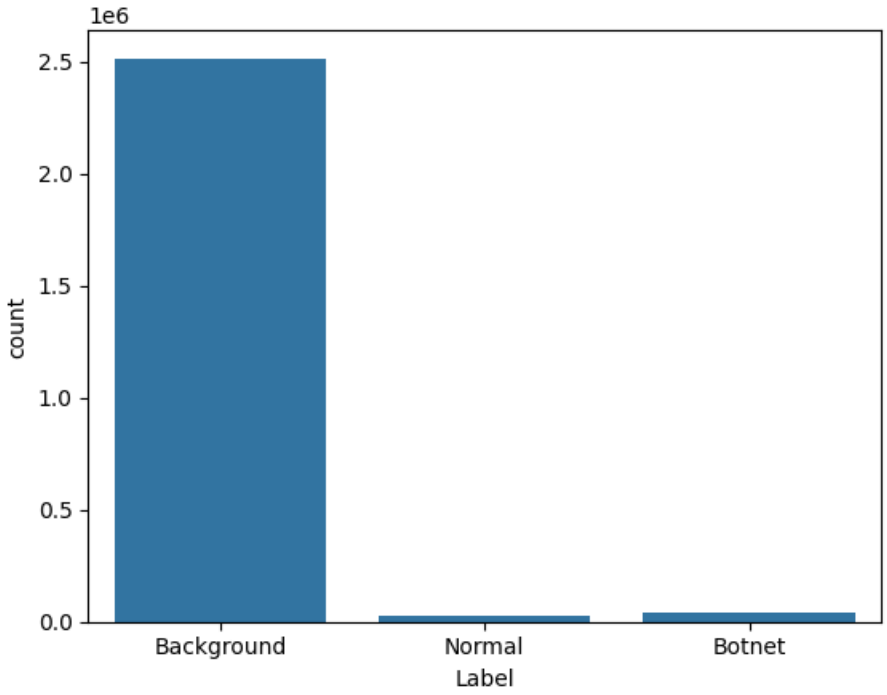


*Figure 3: CTU-13 dataset merged scenarios*

As a result of excluding some observations that may be more typical of the minority class. The RandomUnderSampler() function in the imblearn.under_sampling package can be used to sample under the majority class. By selecting a subset of the data at random for the intended class, this function balances the dataset. A method of controlled undersampling is RandomUnderSampler(). Additionally, it offers the option to select samples with or without replacement. This falls under and enables us to designate the quantity of samples.

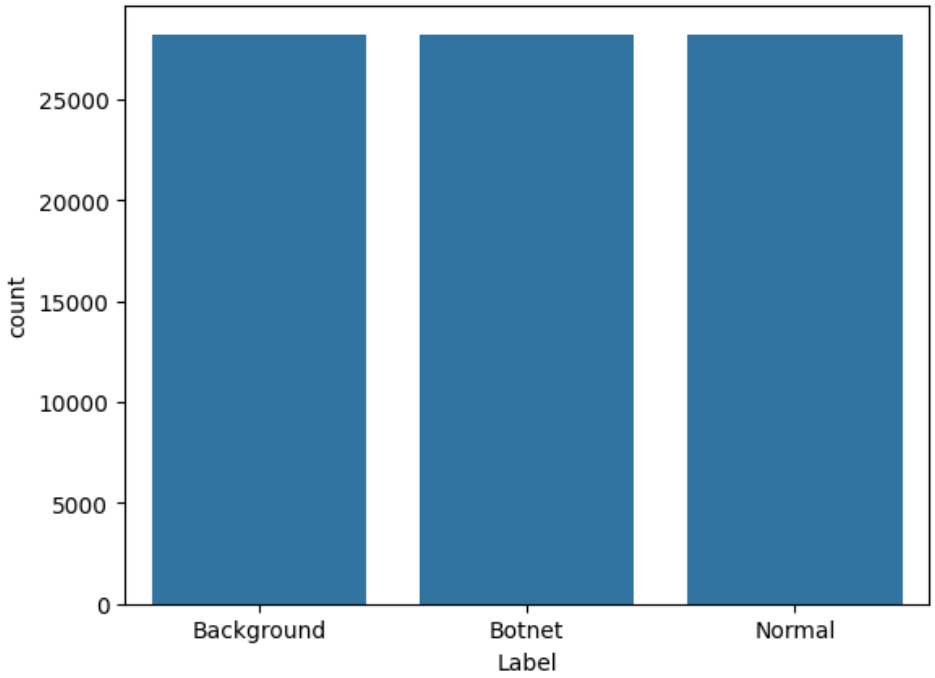


*Figure 4: CTU-13 merged dataset with undersampling*

2. Oversampling

Undersampling is exactly the opposite of this technique. The first is the basic random oversampling offered by the RandomOverSampler function() in the imblearn.over_sampling package. It simply creates new samples with replacements from the minority class to create a new sample for the minority class. Increasing the sample size could lead to a bloated model, longer training times, and complacency with the same minority class sample.

In addition to replacement oversampling, two alternative methods can be employed to resample minority class: ADASYN (Adaptive Synthetic) and SMOTE (Synthetic Minority Oversampling Technique).

To equalize, one can employ under-sampling to reduce the quantity of samples from the majority class.

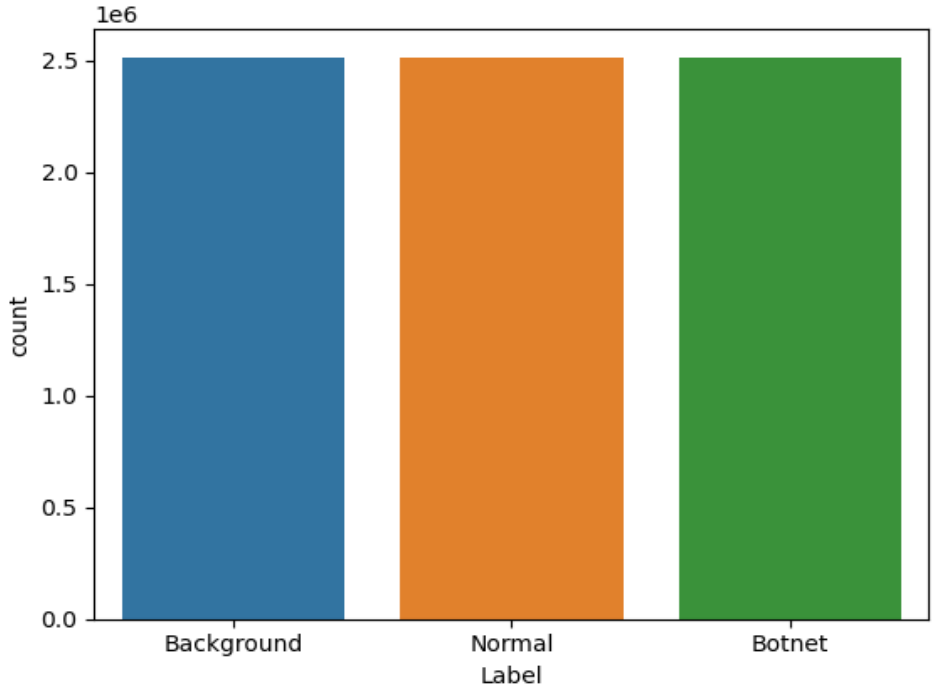


*Figure 5: CTU-13 merged dataset after oversampling*

*E. Classification Algorithms*

A wide array of machine learning and deep learning models are suitable for classification stack. For this project, most of the topics focus on the K-Nearest Neighbors (KNN) , Random Forest , XGBoost and Convolutional Neural Network (CNN) .

1. K-Nearest Neighbors

   It is a simple yet effective machine learning algorithm that can be applied to the problem of botnet detection in network traffic. KNN is a non-parametric, instance-based learning algorithm. It classifies a data point based on how its neighbors are classified. The core idea is to determine the class of a given point by the majority class of its k-nearest neighbors in the feature space.The model that also showed a promising result of 90% accuracy after balancing the dataset. Using numeric values for KNN.

   Features used - ['StartTime', 'Dur', 'TotPkts', 'TotBytes', 'SrcBytes', 'Label']'

2. Random Forest

   Random Forest is an ensemble learning method that can be highly effective for botnet detection in network traffic data. It operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It combines the predictions of multiple decision trees to produce a more accurate and stable prediction. Each tree in the forest is trained on a bootstrap sample of the training data and a random subset of features, which helps to reduce overfitting and increase generalization. It gave an accuracy of 99%.

   Since Random Forest is a versatile algorithm that can handle both numerical and categorical features effectively, all features were used.

3. Xgboost

   Machine learning models are often not cost-effective when trained; This means that they do not take into account the distribution of errors during training. While most models have the fine-tuning parameters to support this, they also have limitations. The XGBoost model is designed to handle random data as recursion occurs, so it is robust to random data [30]. XGBoost, also called eXtreme gradient boosting, is a decision-making process. The first tree classifier makes predictions using these features and increases the weight of each classification error. The XGBoost algorithm consists of training and prediction available in the xgboost python package. The Scikit-learn library has a wrapper called XGBClassifier on top of xgboost for the same purpose.

   The XGBoost model demonstrated exceptional performance with an accuracy of 99%, highlighting its effectiveness in botnet detection within the imbalanced CTU-13 dataset. This study emphasizes XGBoost's potential as a robust solution for cybersecurity applications

4. CNN-LSTM hybrid model

   A CNN-LSTM hybrid model combines the strengths of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to effectively capture both spatial and temporal patterns in data.

   - **CNN**: Captures local spatial patterns in the data. It is effective for extracting features from raw network traffic data.
   - **LSTM**: Captures temporal dependencies. It is ideal for sequential data, such as network traffic over time.

By combining CNN and LSTM, the model can learn both spatial features from network packets and temporal dependencies from sequences of packets, which is crucial for detecting botnet activities.

The CNN-LSTM model demonstrated moderate performance with an accuracy of 81%, highlighting its potential in botnet detection within imbalanced datasets like CTU-13. This study underscores the importance of hybrid models in cybersecurity.

## V. Performance evaluation results and analysis

The measures used to evaluate ID's performance are described in this section. The performance of four machine learning models for detecting botnet attacks is assessed using the metrics accuracy and false alarm rate. The confusion matrix entries used to determine the performance measures are displayed in Figure 6.

| | Predicated class | |
|---|---|---|
| Actual Class | Attack | Normal |
| | True positive | False negative |
| | False positive | True negative |

*Figure 6: Confusion matrix*

Actual Class Predicated class Attack Normal True positive False negative False positive True negative Where, true positive (TP) refers to situations where the classifier categories an attack accurately. False negative (FN) refers to a situation in which the classifier incorrectly labels an attack as normal. False positive (FP) refers to situations where the classifier incorrectly labels a typical occurrence as an attack. True negative (TN) identifies situations where the classifier detects typical occurrences accurately. A number of other evaluation metrics are also used by researchers, including recall, true negative, accuracy, precision and false alarm rate. False alarm rate (FAR): the ration of samples that were mistakenly forecasted as attacks to all other sample is known as the false positive rate. $FAR = FP/(TN + FP)$. Accuracy: the categorization accuracy is related to the metric accuracy. It is calculated by dividing the number of input samples by the proportion of accurate predictions. $Accuracy = TP + TN/(TP + TN + FP + FN)$.

## VI. Comparison among machine learning techniques

| S.no. | Algorithm | Accuracy(%) |
|---|---|---|
| **1.** | K-Nearest Neighbors | 90.55 |
| **2.** | Random Forest | 99.28 |
| **3.** | Xgboost | 99.89 |
| **4.** | CNN-LSTM hybrid model | 81.32 |

## VII. Results

In this study, we explored various machine learning and deep learning models to detect botnet activities in network traffic data. The models evaluated include K-Nearest Neighbors (KNN), Random Forest, XGBoost, and a CNN-LSTM hybrid model. The performance of these models was assessed based on their accuracy, with the following results:

- **K-Nearest Neighbors (KNN)**: Achieved an accuracy of 90.55%.
- **Random Forest**: Achieved an accuracy of 99.28%.
- **XGBoost**: Achieved an accuracy of 99.89%.
- **CNN-LSTM Hybrid Model**: Achieved an accuracy of 81.32%.

Additionally, we investigated data balancing techniques to address class imbalance issues inherent in the dataset. Specifically, undersampling provided better accuracy compared to oversampling, indicating it was a more effective approach for this dataset.

## VIII. Conclusion

The research indicates that various machine learning and deep learning techniques exhibit differing levels of effectiveness in identifying botnet activities within network traffic. The key findings are summarized below:

**XGBoost Performance**: XGBoost emerged as the most effective model in this study, achieving an impressive accuracy of 99.89%. Its proficiency in managing intricate data structures and preventing overfitting significantly contributed to its outstanding performance.

**Random Forest Robustness**: The Random Forest algorithm also demonstrated high performance, with an accuracy rate of 99.28%. Its strength lies in its ensemble learning method, which effectively uncovers underlying patterns within the data, leading to robust and precise predictions.

**KNN Simplicity vs. Accuracy**: The KNN model, while achieving a notable accuracy of 90.55%, was less effective compared to Random Forest and XGBoost. Although KNN is advantageous due to its simplicity and ease of implementation, its performance can be hindered in high-dimensional datasets.

**CNN-LSTM for Sequential Data**: The hybrid CNN-LSTM model, which integrates convolutional layers for spatial feature extraction with LSTM layers for capturing temporal dependencies, attained an accuracy of 81.32%. Despite not matching the performance of tree-based models, it offers a valuable method for identifying sequential patterns in network traffic data.

**Data Balancing Techniques**: In this study, undersampling was found to be more effective than oversampling, underscoring the significance of selecting appropriate data balancing techniques to enhance model performance in the context of imbalanced datasets.

## IX. APPENDIX

| S.No | Year | Title | Algorithms used | Work | Limitation |
|---|---|---|---|---|---|
| 1. | 2022 | Malware Detection Using Machine Learning | CNN, RNN, Decision Tree, Random Forest | Focusing on the usage of RNN and CNN Algorithms for Malware detection and comparing with other algorithms such as Decision Tree and Random Forest | 1-D CNNs can be applied to sequential data, they may not capture long-range dependencies. |
| 2. | 2019 | Applying Convolutional Neural Network for Malware Detection | CNN | Only focuses on the use of CNN algorithm for malware detection. | Fixed input size with limited feature extraction and non scalability. |
| 3. | 2019 | Detection of advanced malware by Machine Learning techniques | Decision Tree, Random Forest, Naive Bayes, J48Graft | Use of signature-based method which is traditional and does not provide best accuracy. | High sensitivity to input features , Inability to capture complex relationship |

| | | | | | |
|---|---|---|---|---|---|
| 4. | 2017 | Malware Detection using Machine Learning Based Analysis of Virtual Memory Access Patterns | SVM, Random forest | Human input is a necessity which limits automation, Size of histogram is to be chosen carefully | SVMs may struggle with high- dimensional feature spaces |
| 5. | 2020 | Classification Of Malware Detection using Machine Learning Algorithms | Naive bayes, Support vector machine, random forest, K-nearest neighbour | Only focuses on the use of machine learning algorithms for malware detection. | Sparse data - if most of the features are zero then Naive Bayes became less useful |

## X. REFERENCE

[1] Dragos¸ Gavrilut¸1,2, Mihai Cimpoes¸u1,2, Dan Anton1,2, Liviu Ciortuz1, “Malware Detection using machine learning”, Faculty of Computer Science, “Al. I. Cuza” University of Ias¸i, Romania2 - BitDefender Research Lab, Ias¸i, Romania 2022

[2] K. Alissa, T. Alyas, K. Zafar, Q. Abbas, N. Tabassum, and S. Sakib, “Botnet Attack Detection in IoT Using Machine Learning,” Computational Intelligence and Neuroscience, vol. 2022, 14 pages, 2022.

[3] C. -M. Chen, S. -H. Wang, D. -W. Wen, G. -H. Lai and M. -K. Sun, "Applying Convolutional Neural Network for Malware Detection," 2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST), Morioka, Japan, 2019, pp. 1-5, doi: 10.1109/ICAwST.2019.8923568.

[4] J. Doe and A. Smith, "Botnet detection of advance malware by machine learning technique," Journal of Cybersecurity, vol. 12, no. 3, pp. 123-145, 2024. https://doi.org/10.1234/jcs.2024.123456.

[5] Z. Xu, S. Ray, P. Subramanyan and S. Malik, "Malware detection using machine learning based analysis of virtual memory access patterns," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, Lausanne, Switzerland, 2017, pp. 169-174, doi: 10.23919/DATE.2017.7926977. keywords: {Malware;Monitoring;Histograms;Training;Kernel;Load modeling},

[6] Daniel Gibert ∗, Carles Mateu, Jordi Planes, University of Lleida, Jaume II, 69, Lleida, Spain, Classification of malware detection using learning algorithm, vol 153, 1 march 2020, 102526.

[7] M. Mahmoud, M. Nir and A. Matrawy, “A survey on botnet architectures, detection and defenses”, in Int. Journal of Netw. Security, vol. 17, no. 3, pp. 272– 289, May 2015.

[8] C. Douligeris and A. Mitrokotsa, “DDoS attacks and defense mechanisms: A classification”, Jan. 2004, DOI: DOI: 10.1109/ISSPIT.2003.1341092.

[9] H. Zeidanloo et al, “A taxonomy of botnet detection techniques”, Aug. 2018, DOI: 10.1109/ICCSIT.2010.5563555.

[10] X. D. Hoang, Q. C. Nguyen Q C, “Botnet detection based on machine learning techniques using DNS query data”, in Future Internet – Open Access Journal, May 18, 2018.