CALIFORNIA STATE UNIVERSITY SAN MARCOS

PROJECT SIGNATURE PAGE

PROJECT SUBMITTED IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

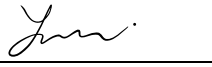
PROJECT TITLE: Comparing Machine Learning and Deep Learning for IoT Botnet

Detection.

AUTHOR: Rishabh Gandhi

DATE OF SUCCESSFUL DEFENSE: 03/19/2021

THE PROJECT HAS BEEN ACCEPTED BY THE PROJECT COMMITTEE IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF
SCIENCE IN COMPUTER SCIENCE

Dr. Yanyan Li		3/19/2021
PROJECT COMMITTEE CHAIR	SIGNATURE	DATE
Dr. Nahid Majd		3/19/2021
PROJECT COMMITTEE MEMBER	SIGNATURE	DATE
Name of Committee Member		
PROJECT COMMITTEE MEMBER	SIGNATURE	DATE

Comparing Machine Learning and Deep Learning for IoT botnet detection.

Master's Project

Presented to
The Faculty of the Department of Computer Science
California State University, San Marcos

In Partial Fulfillment
Of the Requirements of the Degree
Master of Science

By
Rishabh Gandhi
Date- 03/19/2021

Acknowledgement

I would like to thank my project advisor Dr Yanyan Li and my committee member Dr Nahid Majd for their continued support for the duration of my project. This project wouldn't be possible without the constant guidance of my project advisor who was there with me in each and every part of my project and helped me to complete my project in a timely manner. I would also like to thank my department members for providing me an opportunity to take this project to complete my Master's Degree in California State University, San Marcos.

Table of Contents

Figures and Tables

Tables

A. Abstract

Botnet has become a major threat to Internet of Things (IoT) devices due to the low security settings from manufacturers and the lack of security awareness from end users. Many ports are open by default and default user credentials are left unchanged. To solve the increasingly popular botnet attack, many detection approaches have been proposed. However, most of them are targeting on one particular approach or one botnet dataset. There is lacking a comprehensive comparison between different machine learning and deep learning approaches on this task under different datasets collected from different ways. One of the main areas of study about the botnet attack is the comparison about different datasets and how different machine learning algorithms and deep learning algorithms are able to detect the difference between traffic data. In this work, we have measured the performance of 5 machine learning and 2 deep learning-based approaches on 4 recently published IoT botnet datasets collected using real and virtual IoT devices under Mirai malware attack. Our comparison results have shown that decision tree achieved the best detection accuracy as well as the shortest training and testing time.

Chapter 1 – Introduction and Related Work

1.1 - Introduction

Botnet attacks are a prominent category in the field of cybersecurity. The first-ever botnet attack was discovered by EarthLink back in 2001. Since then, a number of botnets have been identified by various programmers from time to time. One of the famous attacks of all time has been Mirai botnet attack of 2016 in which the security of various IoT devices were compromised. Since then, the Mirai source code was made public and many researchers have formed and uploaded their own datasets to carry forward the research work for efficient detection of botnet attacks. There are many publicly available datasets that consists of benign data as well as infected data. From our literature study it was clear that a need for comprehensive comparison study was important to find the difference between the various datasets.

The number of IoT devices are growing at a rapid pace. IoT is defined as ``The interconnection, via the internet, of computing devices embedded in everyday objects, enabling them to send and receive data" [1]. The Dyn Cyberattack in 2016 compromised a network of smart home devices and combined them into a web of botnets (or Zombie Army) with an infectious malware known as Mirai [1]. The attack targeted a number of DNS (Domain name servers) at Dyn and attack throughput is estimated to be in the order of 1.2 Tbps [2]. The source code of Mirai Botnet is publicly available online since it was leaked in 2017 and since then it has been used to carry out various DDoS attacks compromising the security of various IoT devices [2]. Detection of botnets has become a major issue considering the growth rate of botnets. Machine learning algorithms have proved to be effective in differentiating between the normal and malicious traffic [3]. There are various techniques that can be used to detect botnet and these can be categorized into signature based, anomaly based, DNS-based and machine learning based [4].

Recent researches have shown the success of applying machine learning in detecting the malicious traffic or the botnet traffic in IoT devices [5]. One of the major advantages of using machine learning for botnet detection lies in the fact that it produces less number of false negatives and false positives as compared to other methods [6]. For botnet detection task, both supervised and unsupervised learning can be used. Supervised learning requires labelling of data during the training process so that it can be used during the classification process. There is no need for data labelling in unsupervised learning [6]. These algorithms are used on the data-based approach to train the classifier and detect the botnet attack [7].

The comparison study of botnet detection can be done in a number of methods. But the important question lies in the fact that what method can be the most effective and efficient in detecting the presence of botnet in the network traffic. There are many papers discussing

botnet detection, but most of them target on one particular approach or one botnet dataset [8]. There is lacking a comprehensive comparison between different machine learning and deep learning approaches on this task. Also, many studies are done based on the public datasets to efficiently complete their research. Using a dataset depends upon the type of research being conducted. Both the public datasets and creating our own dataset have their pros and cons respectively. The data cleaning process and feature extraction in the public dataset can be a cumbersome task. On the other hand, creating our own dataset is also a very difficult as it requires a lot of hardware and software configuration.

For this comparison study, we used some of the new public datasets as they are very efficient and well labeled. Also, the new datasets had many new devices that were previously not seen in any other datasets. Some of the main contributions of this project are:

1. Compared five different machine learning and two deep learning algorithms on IoT botnet datasets.

2. Applied Chi-Squared feature selection method to choose the best features for our algorithms.

3. Identified the method that works best in botnet detection regardless of the datasets and features being used.

4. Tabulating the results to get the performance of all the algorithms and identify which algorithm works best regardless

The rest of the report is organized as follows. Chapter-2 shows the comparison framework. Chapter-3 highlights the experiment and evaluation results. Chapter-4 presents the future work and Chapter-5 contains all the references

1.2 - Related Work

Botnet Detection Based on Machine Learning Approach - For botnet detection, Nadia Chaabouni [1] introduces a Network Intrusion detection system (NIDS) based on machine learning techniques. They have surveyed different papers in their research and based on the survey, they have applied various machine learning algorithms on various public dataset for their research. Ayush Kumar and Teng Joon Lim [2] shows early detection of IoT malware network using machine learning techniques. They introduce their own dataset by analyzing the traffic patterns for IoT malware through testbed experiments and packet capture utilities. For early detection, they have used gateway-level traffic classification by including only TCP packets with SYN flag activated. Zhipeng Liu [3] proposed an anomaly detection using machine learning by exploring a new dataset called IoT Network Intrusion Dataset and they perform their experiments using multiple machine learning algorithms. The dataset created by them uses AI speaker and smart cameras. Shao-Chien Chen [4] uses Convolutional Features with Neural Networks for botnet detection. They extract a bunch of flow- based features from the packets header and train the neural network on those features. They proved that using the convolutional features, they were able to provide an accuracy of 94.7% for P2P botnets. Rohan Doshi [5] uses IoT specific network behaviors to detect botnets using various machine learning

algorithms and neural networks. They use their own dataset for the research that is created using a network of simulated IoT devices. Their pipeline is able to perform data collection, feature extraction and binary classification for traffic data that consists of DDoS attacks. The main focus of the paper is to identify the IoT botnets at local network level.

Botnet Detection using Machine Learning and Deep Learning Approach- Ruchi Vishwakarma [6] uses a new technique using the honeypots along with machine learning to detect IoT based botnet attack detection. The honeypots are used to attract various hackers and store the information about their style of invading and capturing malware properties in a log file. They use this log file as the input for algorithms. They show the advantage of using this approach as the model learns unknown variants of malware families. Shamsul Haq [7] also uses Machine Learning for botnet detection. They have compared the signature-based approach to the anomaly-based approach for their study. They have also conducted a literature survey of different papers and different algorithms. Hadeel Alazzam [8] introduces Supervised Detection of IoT Botnet. The use the UCI dataset i.e., N-BaIoT dataset for their work and tried to classify between malicious traffic data and benign traffic data using machine learning approach. Mohit Goyal [9] detects HTTP botnet using Network traffic analysis. They use the source of the botnet in the traffic data to detect botnet. The traffic data is grouped according to the host name. The main focus of the paper is to detect the botnet by focusing on the source rather than on the bot used. In [10] deep learning approaches are used on the network flow traffic. The have also used the t-distributed stochastic neighbor embedding visualization technique to show the characteristics to dataset being used by them.

Botnet Detection based on Custom Approaches - Xiaoyu Liang [11] introduces a long short term memory enabled framework for DDoS Detection. They have implemented the long short term memory algorithm to classify the malicious traffic data and benign data in the network flow. The show that small number of packets is also sufficient to learn the dynamic behaviors in the algorithm. Francisco Sales de Lima Filho [12] uses a smart detection approach for botnet detection using an approach that does not require traffic redirection or connection intermediation that can detect the botnet in early stages. This approach also makes sure that the data remains private. They use their own custom dataset along with other ones. Huy-Trung Nguyen, 2020 [13] introduced a PSI-Rooted subgraph for botnet detection using classifier algorithms. In this approach, a PSI-rooted subgraph-based feature is used in a static manner. These features are based on both IoT-botnet information and the topology of the PSI- graph used. In [14] the machine learning approach is expanded for IoT enabled smart cities. They have proposed an architecture in which rather than using a single unit for their computational needs they use a number of less powerful units that run parallel to each other. Gonzalo [15] uses distributed deep learning approach to detect internet of things attacks. In the paper they have designed a joint training method that uses less communication and resource usage for the anomaly detection in traffic data.

Chapter 2- Botnet and Botnet Attacks.

2.1 - What is a Botnet?

Botnet is a collection of IoT connected devices that are compromised by an attacker to perform unauthorized tasks from the devices. Botnets have become very common nowadays to their volatile nature as they can spread very easily from one device to another. Botnets are commonly used to perform distributed denial of service attacks or DDoS attacks. Most commonly, a command-and-control server is deployed at the backend of these devices that controls the spread of the malware and controls all the attacks that can be performed by it. One of the most historic botnet attacks is the Mirai botnet attack of 2016 that was used to bring down some of the biggest companies like twitter, Netflix etc. The botnet attack was so huge that it bought down internet for the entire country of Liberia. The source code for the Mirai botnet was published on the internet soon after the attack and since then it is used by researchers all around the world to detect methods to prevent the attacks. Some of the recent work is being done to stop these attacks at the early stages. The figure below shows all the steps used in deployment of botnet.

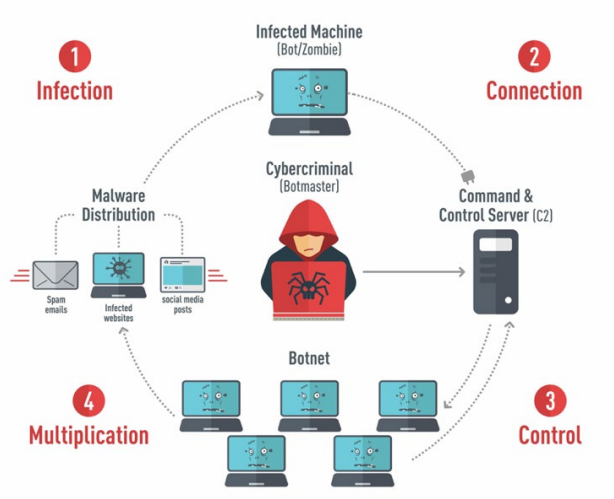


Figure 1[26] – Implementation of botnet attack

2.2 - Types of Botnet Attack.

The botnet attack can be used to carry out various types of attacks. Therefore, the botnet attack can be categorized into different categories according to the type of botnet attack carried out.

1.) Pishing Attacks – As we know that the botnets can easily be transferred from one device to another, therefore one of the common botnet attacks are the pishing attacks that can be sent out by the attacker by using the pishing emails.

2.) Distributed Denial-of-Service attack (DDoS attack) – Most advanced category of botnet attacks is the distributed denial of service attacks or DDoS attacks. In this type of attack, the botnet sends a large number of requests to the attacking server particularly more requests than the server can handle which ultimately crashes the server. The two types of DDoS attacks are network layer attack and application layer attack. The network layer attack consists of SYN floods, UDP floods etc. In these attacks, the entire bandwidth of the network gets consumed by the requests which results in the crashing of the server. The application layer attacks consist of HTTP floods, RUDY attacks, zero-day attacks etc. In this attack, the botnet targets the vulnerabilities in the application which is used to crash the server.

3.) Snooping – Another type of botnet attack is the snooping attack that is particularly used to eavesdrop on credentials of the user and steal it. This type of botnet, monitors the network traffic of the user to inject the malicious code into the network HTTP traffic.

4.) Bricking – Bricking is the attack used by attackers to remove any clue that might have been left behind in the initial attack. Due to the size of IoT devices and less computational power, most of the IoT devices have a weak security. To take advantage of this, attackers deploy botnets that removes the software from these IoT devices which results in making them useless or bricked. As there is no evidence left behind on the device, the investigation cannot reveal any results about the attacks on these devices.

Chapter 3 – Comparison Framework

3.1 - Libraries Used

For this project we will be working with various machine learning algorithms that are provided by various libraries in python. In machine learning, we have a dataset that is used to predict the properties of unknown data. Therefore, we can divide the problem category into two classes namely supervised learning and unsupervised learning.

In supervised learning, we have data that has many attributes that we want to predict. These kinds of problems consist of classification and regression. In unsupervised learning, we have a set of input variable that that do not have any target values defined.

For the dataset, we divide the dataset into testing dataset and training dataset. For training dataset, we use a part of dataset to train the algorithms so that it can have knowledge of the data that is defined and stores that knowledge. For the testing dataset, we use the gathered knowledge from the training dataset and apply it on the rest of the dataset to check for its properties. In machine learning, we have many libraries that provide us with algorithms that we can use for this classification. In our project, we have used Scikit learn, Keras, Pandas, Matplot libraries

Scikit Learn - The Scikit learn library is a fundamental library of python that provides us with various machine learning algorithms with supervised learning such as logistic regression, support vector machines, decision tree etc. This library helps us in modelling of data for example clustering, cross validation, manifold learning, ensemble methods etc. The main use of this library is prototyping and production of projects. It is used by many multinational corporations nowadays.

Keras - Keras is the open-source library of python that deals with deep learning neural networks that are called artificial neural network. It is mainly useful for image recognition, robotics and artificial intelligence. Keras runs on top of many other python libraries such as TensorFlow, CNTK- Cognitive toolkit and Theano. Features of this library consists of Highly scalable computation, easy of use with CPU and GPU, highly consistent with files etc. This library is used for deep learning algorithms in our project.

Matplot Library – Visualization of the results is the easiest way to read the output from our program. This is made possible with the matplot library in python. This library helps us in plotting various graphs such as line graph, bar chart, histogram etc. In this project we have used matplot library to construct ROC curve that is Receiver Operator Characteristics that plots a graph between true positives and false positives. These are shown in the Results section.

3.2 Chi-Square Feature Selection Method

Feature selection is an important part of machine learning. Feature selection helps in reducing the number of features and select those features that are best suited for our algorithms. It also helps us in improving the performance of our model and also in reducing the computational time of our algorithm. There are two types of feature selection methods namely supervised and unsupervised feature selection method. For this project, we have selected Chi-Square feature selection method. Chi-Square feature selection method comes under the category of Filter feature selection method that take in count the data type of the variable that we input and computes the relationship of this with the target variable. Chi square feature selection method consists of a series of step to be evaluated [19] [20].

Firstly, it checks if the variables are independent or not. This step is called defining hypothesis. Secondly, we use a Contingency table to check the correlation between two variables. Next, we find the expected value of two events using the probability factor.

After this the algorithm computes whether the value falls in acceptance range of 95% confidence or not. Also, we can limit the number of features that we want to use for our algorithm. For our research, the number of features varies for each dataset as each dataset has different set of features available and we have used Chi-square feature selection method on all the algorithms. Chi-square is used to compute the values using the following formula:

$$x_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Where: c = degree of freedom

O = observed values

E= Expected Values

Chi-Square feature selection method is used for feature selection in our project. This provides us with a common basis of comparison for all the algorithms. In the chi square algorithm, we can input the number of best features that we want to select out of all the features. This is the called the k value in algorithm. We have chosen 20 best features for the N-BaIoT dataset, MedBIoT dataset and Kitsune dataset, and for IoT 23 Dataset we have chosen 10 best features for all the algorithms.

3.3 Machine Learning Algorithms used:

For botnet attack detection, we have chosen 7 different machine learning algorithms that will help us in comparing the accuracy of different algorithms on different datasets. The five ML algorithms used in the project are Logistic regression, K-nearest neighbor, Naïve Bayes, Decision Tree, Random forest and two deep learning algorithms are Multilayer Perceptron (MLPN) and Long short-term memory (LSTM).

3.3.1 - Logistic Regression – As we know that in machine learning we have two kinds of problem, first is the regression and second is the classification. Logistic regression algorithm falls under the category of classification algorithm in which we get our output defining the category in which the output falls. The logistic regression algorithm uses a logistic function or the sigmoid function for classification. The sigmoid function is a function which when plotted on a graph takes the form of S shape. The graph for the sigmoid function takes the values between 0 and 1, and the graph is plotted at margins, top and bottom. When plotted, the graph takes an S shape. The equation for the sigmoid function is as follows – $y=1/(1+e^{-x})$. In the above function, the value of e is the exponential function that has a value of 2.71828 The S shape graph for the sigmoid function is shown below for an example dataset.

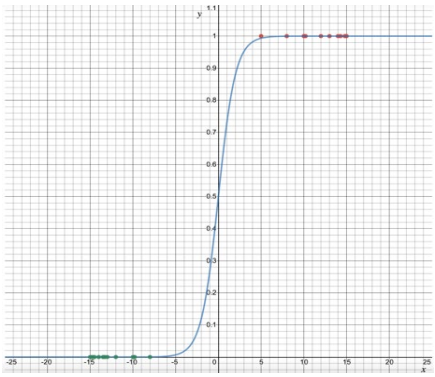


Figure 2[21] – Sigmoid Graph

In logistic regression, one of the main parameters is the threshold that is set. For example, if we have a threshold of 0.5, the all the classification above the threshold is classified according to our problem. Therefore, the logistic regression algorithm is one of the best suited algorithms for classification problems.

3.3.2 - K-Nearest Neighbors – The K-Nearest Neighbors or KNN is a machine learning algorithm that works best for classification problems. KNN algorithm works on the method of calculated prediction or an educated guess. Suppose we want to guess a data point. The KNN algorithm makes prediction for that data point based on the greatest number of points that resemble the same data point. The KNN algorithm is fairly easy to use when compared to other algorithms as it does make any assumptions about the data and is also considered very fast. The KNN algorithms do not make a generalization of the data i.e., the training for this algorithm is very

less. The KNN algorithm works on the grounds of calculating Euclidean Distance between the neighbors. The formula for the Euclidean distance is as follows:

$$d(p,q) = d(q,p) = \sqrt{(q1-p1)^2 + (q2-p2)^2 \dots \dots (qn-pn)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (qi-pi)^2}$$

The KNN algorithm computes the distance and then classifies the data based upon which data points are most similar to the nearest data points and all those have the highest probability of being similar. The figure below shows the KNN algorithm that predicts the data point that is nearest to it.

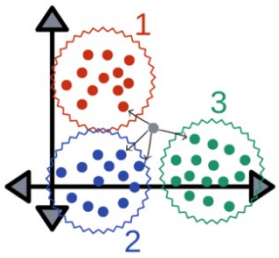


Figure 3[22]- KNN Algorithm

The KNN algorithm is used by many big corporations now a days to help in recommendations for their customers in watch movies or listening songs based upon recent activity of the user and other users who may have the same taste.

3.3.3 - Naïve Bayes – Naïve Bayes is a probabilistic classification algorithm that is based on the Bayes theorem. The algorithm is called naïve bayes because the algorithm works on the principal that the occurrence of a certain feature is purely independent of the other features. Suppose we have to classify a fruit. A fruit has many features such as shape, color, size, taste etc. Therefore, this algorithm assumes that all the make their own individual contribution in identifying the fruit rather than all of the features combined. The Bayes theorem works on conditional probability. The formula for Bayes Theorem is **P(A|B) = (P(B|A)*P(A))/P(B)** where P(A) stands for probability of occurrence of event A, P(B) stands for probability of occurrence of

Event B , P(A|B) stands for probability of occurrence of A while B is true and P(B|A) probability of occurrence of B while A is true.

3.3.4 - Decision Tree - Decision tree in python is an algorithm that uses a tree like structure to classify the problems. This algorithm can be visualized as a flowchart type structure. The root note of three represents the feature that we are using. The leaf nodes of the tree represent the outcome of the classification. The branch of the tree represents the decision rule that is used predict the outcome based of the training dataset. This structure of the decision tree helps in decision making of the tree. A general representation of the decision tree algorithm is shown in the figure below.

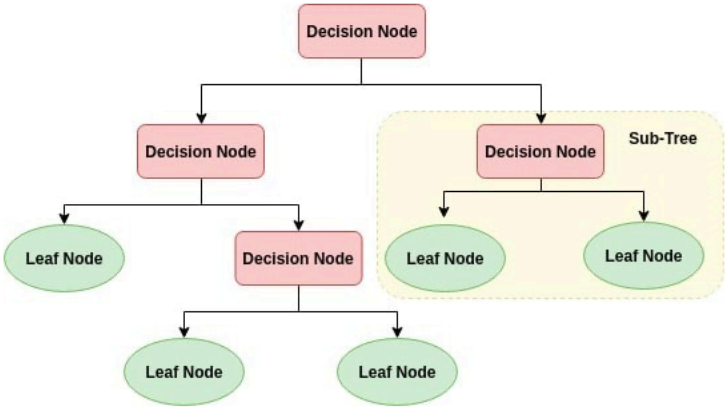


Figure 4[23] – Structure of Decision Tree

The basic concept of decision tree algorithm consists of a series of steps. First, the algorithm selects the best attribute from the training dataset. Second, it breaks the dataset into smaller dataset and repeats this process for each child node until there are no more instances left or no more of attributes are left.

3.3.5 - Random Forest – The basics of random forest algorithm are obtained from decision tree algorithm. The random forest algorithm uses a number of decision trees that classify the problem. As a decision tree algorithm classifies a problem, the random forest classifier uses a vote to count the greatest number of class predictions by different decision trees and the one with the greatest number of class prediction wins. The figure below shows the basic concept of Random forest Classifier.

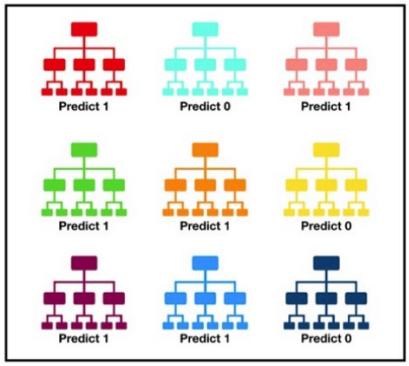


Figure 5[24] – Example of prediction by Random Forest

Although both the algorithms look the same, there are some differences between the decision tree and random forest. First, the decision tree algorithm forms a set of rules or decision to predict the class of data whereas the random forest uses random features to compute a series of decision trees.

3.3.6 - Multi-Layer Perceptron Neural Network – Multi-Layer perceptron network is a class of artificial neural network. MLPN consists of a network of fully connected network layers that are able to pass information from one layer to other. The layers in MLPN consists of input layer, hidden layer and the output layer. The MLPN network is a feedforward network. This means that the data flows from the input layer to hidden layer and finally the output layer. Hidden layers have a series of activation functions that are activated. The hidden layer goes through this process multiple times until it reaches the output layer where the final decision is made. The figure below explains the structure of multi-layer perceptron network.

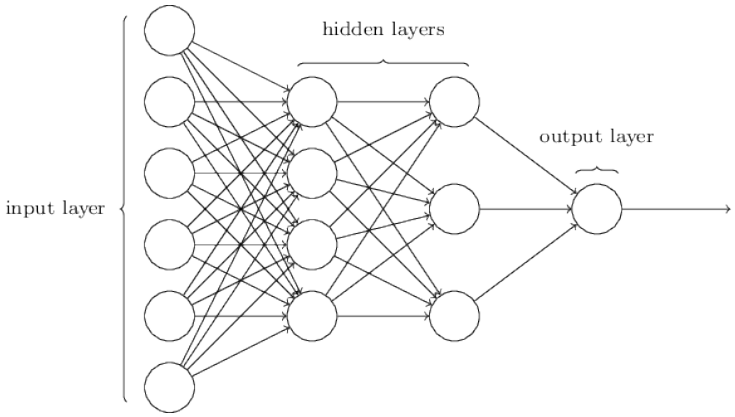


Figure 6 – Structure of MLPN

3.3.7 - Long-Short Term Memory (LSTM) - LSTM is a class of Recurrent Neural Networks (RNN) where the neurons are interconnected and input to the next neuron is the output from the last neuron. RNN have their own internal memory that is used to give the output. RNN have a possibility of leaving out important information as they have a short-term memory. They have a hard time in carrying forward the information from one to another in long sequence of information. Here, the LSTM algorithm plays an important role. The main unit in LSTM is the memory call that stores all the information and decides which information to keep and which one to forget. It consists of three gates Input gate, Output gate and Forget gate. The cell state vector in LSTM can maintain its state over time which gives it long memory. After processing the data, a model is generated for each algorithm through which the testing data is passed. The structure of LSTM memory cell is shown below in figure 7. For this project we have used Univariate LSTM along with Vanilla LSTM. The input layer in this model takes the input from the dataset and the number of hidden layers in the model is input by the user. After that, the output is provided by the output layer.

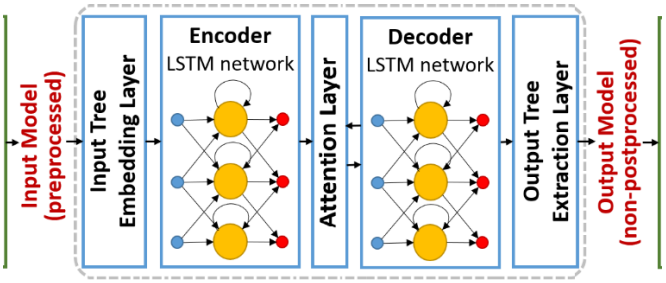


Figure 7[25] – Structure of LSTM

3.4 Hyperparameters:

In machine learning there are various internal value for each algorithm that need to be fine-tuned according to our dataset and the kind of problem we are working on with machine learning. These parameters are known as hyperparameters. The final result of any machine learning problem greatly relies on the kind of hyperparameter we are using in order to get our results. The tuning of parameters depends widely on tunability of the algorithm. There are various processes to tune the parameters that comprises mainly of automated process and manual process.

For this study, we used the manual process to fine tune the parameters that works best with our datasets. In our study we use a couple of deep learning algorithms that need tuning of hyperparameters for the best results. In the manual process, we tune the parameters and analyze the results using the ROC curve and AUC curve. ROC curve and the AUC curve can we plotted using the MATLAB library of python and can be analyzed for best results. ROC curve is the receiver operating characteristic curve that shows the performance of the algorithm that

we are using. AUC curve is the area under the curve that measure the ability of the algorithm to classify our problem.

For our multi-layer perceptron algorithm and LSTM algorithm, we have our activation as sigmoid as sigmoid provided us with best results. Other hyperparameters that can be used to tune the activation layer are softmax, softplus, softsign etc.

3.5 Botnet Traffic in Datasets

All the botnet traffic for dataset creation is packet-based approach and stored in a csv file format. In this format, all the features are represented in the columns and all the packet data is stored in each row. Therefore, each row represents a packet of data and likewise each column corresponding to that row represents features for that packet of data. This is essentially important as .csv files are relatively easy to process and access for machine learning and deep learning algorithms.

3.6 Datasets

For our project, we decided to use the publicly available datasets as they would be useful for comparison purpose. From our literature study part, we explored different datasets that would be a really useful. We came up with four different datasets each with different IoT devices infected with Mirai malware. Each of these datasets had two parts one consisting of raw traffic data or benign data and the other one with infected traffic data. The four datasets used are:

1. N-BaIoT Dataset.

2. IoT-23 Dataset.

3. Kitsune Network attack Dataset.

4. MedbIoT Dataset.

In the further sections, we will discuss in detail about each of these datasets about what devices these data use, their data collection process and the features extracted in all the four datasets along with their topology.

3.6.1 - N-BaIoT Dataset

This dataset is one of the most popular datasets for IoT botnet detection. As we have seen in our related worl, many research papers have used this dataset for detection of Mirai and Bashlite Malware. This dataset is published by University of California, Irvine. This dataset consists of 9 commercially available IoT devices infected with 2 botnets. However, for the scope of this project we focused only on Mirai. The 9 devices used in this dataset are:

1. Danmini Doorbell
2. Ecobee Thermostat
3. Ennio Doorbell
4. Philips B120N10 Baby Monitor
5. Provision PT 737E Security Camera
6. Provision PT 838 Security Camera
7. Samsung SNH 1011 N Webcam
8. SimpleHome XCS7 1002 WHT Security Camera
9. SimpleHome XCS7 1003 WHT Security Camera

The data collection process for this dataset was setup in an isolated lab environment that consisted of IoT devices, a C&C server and a server for scanning and loading. The raw traffic data was captured immediately after turning the network setup. For the sniffing of network traffic, port mirroring was used on the switch through which all the data flows and wireshark was used to collect all the data. The Mirai attacks executed in this dataset are Scan, Ack flooding, Syn flooding, UDP flooding and UDP flooding with fewer options. The basic topology of this dataset is shown below in figure 1.

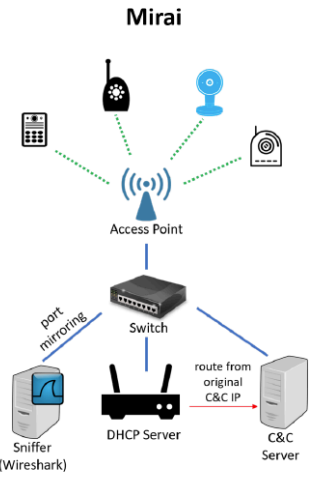


Figure 8[16] – N-BaIoT Dataset

Features – There are a total of 23 features extracted from the network traffic for five different time frames i.e., 100ms, 500ms,1.5 sec, 10sec and 1 min. All the features are extracted by

capturing the behavioral snapshot of both the host and the protocols that are used to communicate with the arrived packet. All the features are extracted from this behavioral snapshot. The 23 features are described in the table below.

Value	Statistic	Aggregated by	Total Features
Packet Size	Mean, Variance	Source Ip, Source MAC-IP, Channel, Socket	8
Packet Count	Number	Source IP, Source MAC-IP, Channel, Socket	4
Packet Jitter	Mean, Variance, Number	Channel	3
Packet Size	Magnitude, Radius, Covariance, Correlation, Coefficient	Channel, Socket	8

Table 1 – Features of N-BaIoT Dataset

3.6.2 - IoT-23 Dataset

IoT-23 dataset is one of the new datasets published in 2020 with an aim to provide a relatively large dataset with labelled traffic data and different botnets. This IoT network traffic was captured in the Stratosphere Laboratory, AIC group, FEL, CTU University, Czech Republic. This dataset and its research are funded by Avast Software, Prague. This dataset was not so commonly used by researchers and this gave us an opportunity to try and develop machine learning algorithms for this dataset. One of the advantages of this dataset is that it consists of all the real-world devices and not the simulated devices. Also, one of the devices is a smart speaker which was not found in any other datasets. The IoT devices used in this dataset are:

1. A Philips HUE Smart LED Lamp.
2. An Amazon Echo Home Intelligent Personal Assistant
3. A Somfy Smart Door lock.

The malicious data in this dataset were captured for a period of 24 hours. But as the data is generated, the size of data grows out to be too large therefore the data capture has to be stopped before 24 hours. This dataset also provides some additional information regarding the traffic data collected from the infected devices using Zeek that uses the application layer filter protocol to filter out and provide this additional information.

The network topology for this dataset is shown below in the figure.

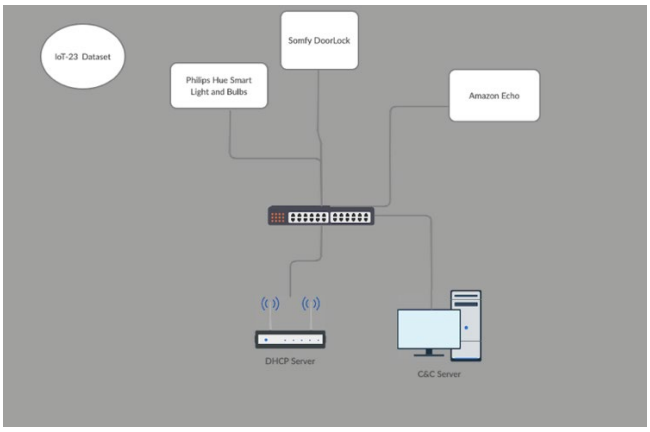


Figure 9 – IoT-23 Dataset

Features – IoT-23 is a huge dataset with several different scenarios of different botnets. For all the botnets, a total of 21 features are extracted. The list of all 21 features are as follows:

1. Flow start time
2. Unique ID
3. Source IP address
4. Source port
5. Destination IP address
6. Destination port
7. Transaction protocol
8. Service http, ftp, smtp, ssh, dns, etc.
9. Record total duration
10. Source2destination transaction bytes
11. Destination2source transaction bytes
12. Connection state
13. Source local address
14. Destination local address
15. Missing bytes during transaction
16. History of source packets
17. Flow of source bytes
18. Destination packets
19. Flow of destination bytes
20. Traffic tunnel
21. Attack label

Table 2- Features of IoT-23 dataset.

3.6.3 - Kitsune Network Attack dataset

Kitsune network attack dataset is a cybersecurity dataset that is available publicly and consists of nine network attacks in total. The network attacks include MitM, DoS and various botnet attacks. For our project, the main focus is on Mirai botnet hence we will be using part of this dataset that focuses on Mirai. The Mirai attack was performed for a total of 118.9 mins on mainly surveillance device network and IoT hardware.

The devices used in this network attack are-

1. Sony SNC-EB600
2. Sony SNC-EB602R
3. Sony SNC-EM600
4. Sony SNC-EM602RC
5. Various Home Surveillance Models

The packets are captured using some of the external libraries such as NFQueue, Tsharketc. Also for the packet parser, external libraries such as packet++ and tshark are used. The figure below shows the basic topology for the dataset.

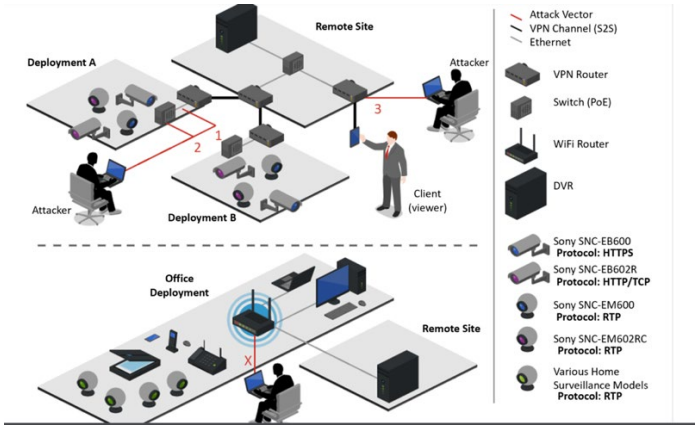


Figure 10[17] - Kitsune Network

Features – For the feature extraction part, the publishers of this dataset created a new custom framework for the process of feature extraction from a dynamic number of network channels.

Value	Aggregated By	Features	Description of Statistics
Packet Size	Source MAC-IP, Source IP, Channel, Socket	8	Bandwidth of the outbound traffic

Packet Size	Channel Socket	8	Bandwidth of outbound and inbound traffic together.
Packet Count	Source MAC-IP, Source IP, Channel, Socket	4	Packet rate of outbound traffic.
Packet jitter	Channel	3	Interpacket delay of outbound traffic.

Table 3 – Features of Kitsune Dataset

3.6.4 - MedBIoT Dataset

MedBIoT Dataset stands for medium sized IoT botnet dataset. This dataset is a very new dataset that is published in the year 2020. This dataset is published by Tallinn University of Technology, Estonia. The complete dataset consists of three major botnets i.e., Mirai, BashLite and Torii. For our research we are focused only on Mirai. The thing that makes this dataset unique is that this dataset is a combination of real and simulated IoT devices. Also, this dataset focuses on early stages of botnet deployment i.e., spreading and C&C communication. The main devices used in this botnet dataset are:

1. Sonoff Tasmota smart switch.
2. TPLink smart switch
3. TP link light bulb.
4. Lock (Simulated)
5. Switch (Simulated)
6. Fan (Simulated)
7. Light (Simulated)

The network topology for this dataset is shown in figure 4

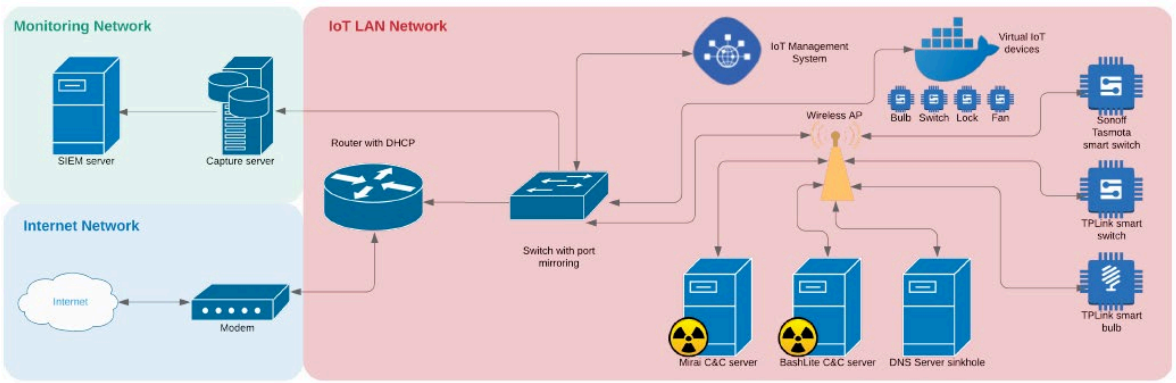


Figure 11[18] – MedBIoT Dataset

Features – MedBIoT dataset consists of a total of 100 features in four categories that are captured in five different time frames i.e., 100ms, 500ms,1.5s, 10s and min. All the four categories of features are listed below.

Categories	Features
Host-MAC & IP	Packet count, mean and variance.
Channel	Packet count, mean, variance, magnitude, radius, covariance, and correlation.
Network Jitter	Packet count, mean and variance of packet jitter in channel.
Socket	Packet count, mean, variance, magnitude, radius, covariance and correlation.

Table 4 – Features if MedBIoT dataset

3.7 - Workflow for evaluation of results.

1. Data Summary – As we are using different datasets for this evaluation, we start the process by looking at the basic quantitative data about the dataset. We have defined the show data feature in beginning that prints the data type of the dataset and describes the dataset as number of rows and columns.

2. Preparation of the Dataset- Preparing the dataset is one of the most important part in Machine Learning as it prepares our data for different machine learning algorithms. There are two classes of data in our dataset that is benign or raw data that represents the normal flow of traffic and the other one is infected data that represents the class of data that is infected by Mirai botnet. We first access the benign class of data. A new column is added at the end of benign class of data with a label 0. Likewise, the infected class of data is labelled as 1. To access this as one dataset, both the files are concatenated in one file labelled as test result. This step plays a major role as we are classifying data using machine learning and this data will be used in next steps to train and test the algorithm.

3. Processing the dataset- Another major step in Machine Learning is the processing of dataset. In this process we need to split the dataset into testing dataset and training dataset. The training dataset is used to train the machine learning model that we are using and make sure that the dataset fits our model. The next is the testing data that is used to evaluate the final model. There are two variables used to get the desire variable for our dataset. One is to determine the training size of the data and the other is random state that is used to guarantee that the same sequence of random number is generated every time. There are two general split ratios for the dataset. One is 70:30 where 70% is training data and 30% is testing data. Another

one is 60:40 where 60% is training data and 40% is testing data. We have used 60:40 in this project as it provides more testing data to the algorithms and also provides us with good results.

4. Comparing the Machine Learning algorithms- Our main study is evaluating the results of different datasets based on their accuracy of detection. We use the preprocessed data i.e., training data and testing data on different machine learning algorithms. In the final steps the results are computed and tabulated.

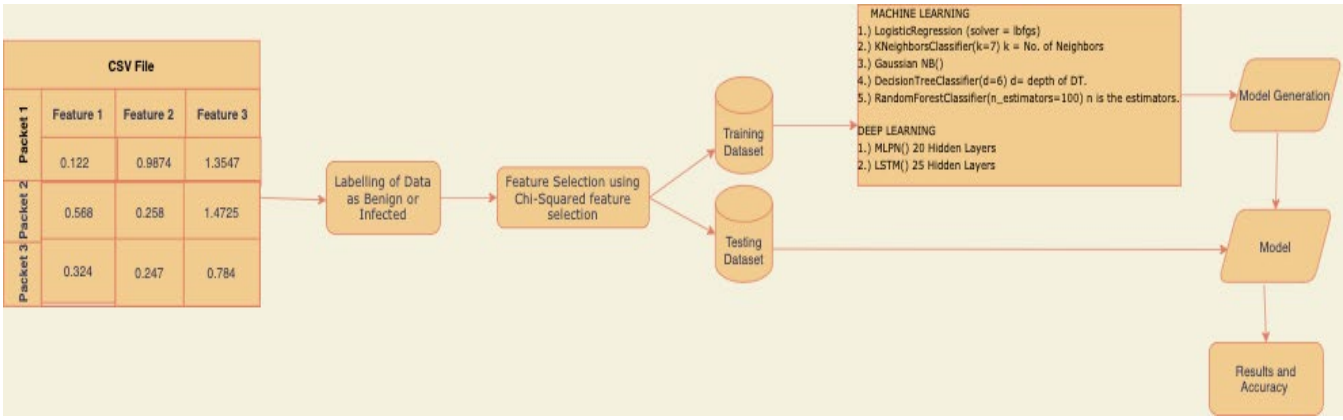


Figure 12- Summary of Workflow for evaluation of results

3.7.1 Five Stage Processing of Data for Results

There are a total of five stages through which the data is processed for accurate evaluation of results. The very first part of the project is data collection and the very last stage is calculation of results for each of the dataset. Each dataset is processed through the same series of steps to have a common ground for effective comparison. Each stage consists of their own series of steps that are used in the next stage in our workflow. The five figures below show each step-in detail:

Figure 13. Stage 1. Data Collection

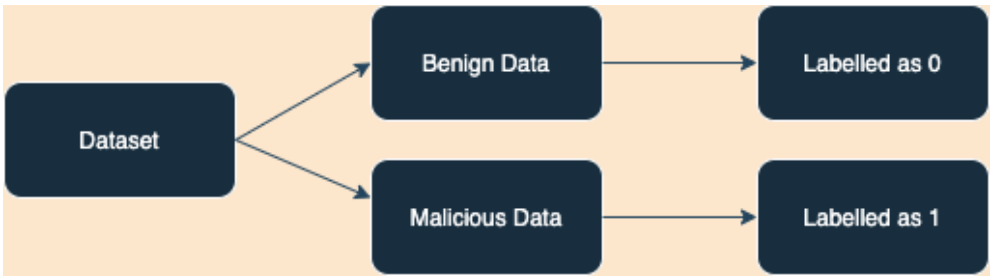


Figure14. Stage 2. Labelling of Data

Figure 15. Stage 3. Feature Selection

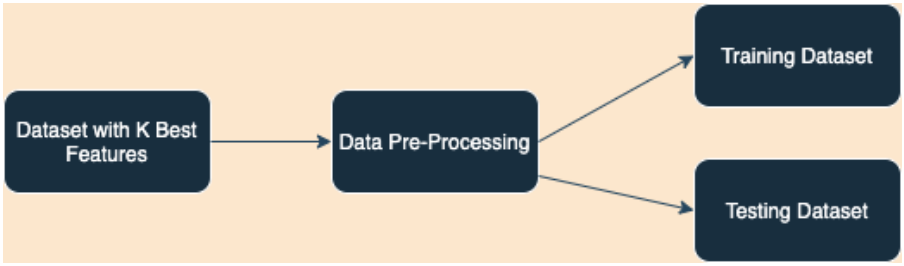


Figure 16. Stage 4. Data Pre-Processing

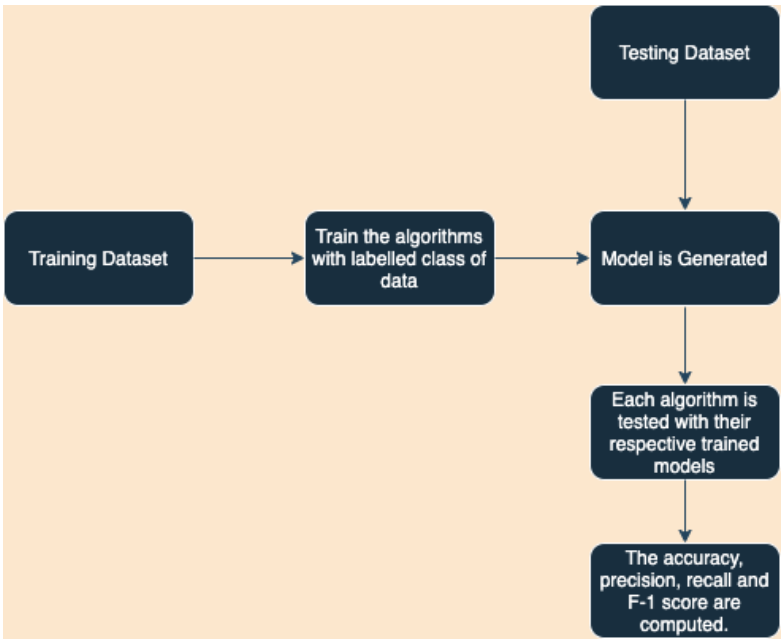


Figure17. Stage 5 – Training and testing the algorithms.

Full text: Flowchart starts on the left at “Training Dataset,” which leads to “Train the algorithms with labelled class of data.” That entry, along with a separate above entry, “Testing Dataset,” both lead to “Model is Generated.” That one leads to “Each algorithm is tested with their respective trained models” and then “The accuracy, precision, recall and F-1 score are computed.”

Chapter-4 Experiment and Evaluation

4.1 Experiment

To efficiently evaluate the accuracy of detection, we used Google Colaboratory for our research. Google provides us efficient platform with high performance CPU and GPU that speeds up the process. This in turn reduces the total time required to calculate the results. All the datasets are stored in Google drive for efficient management. All the inputs to the Google Colaboratory are provided from Google Drive.

For the evaluation process, we used five machine learning algorithms and two deep learning algorithms to compare the results.

Each model was trained with the features selected by the chi-square feature selection method. The deep learning models were fine-tuned by tuning the hyperparameters in each layer of the neural network.

Each dataset was processed for each algorithm and the results are computed. At last, the results are tabulated so that all the results can be compared efficiently.

4.2 Results

The results are evaluated for four datasets using five different machine learning and two deep learning algorithms. The accuracy, Precision, Recall, F1 score, testing time and training time are comprehensively compared in the tables below. The accuracy is calculated as (TP+TN)/(TP+TN+FP+FN). Precision is calculated as TP/(TP+FP). Recall is calculated as TP/(TP+FN). F1 Score is calculated as 2*precision*recall/ (precision + recall). ROC curve is the receiver operating characteristic curve that shows the performance of the algorithm that we are using by plotting a graph between the true positive rate and the false positive rate.

The evaluation results for the four datasets are listed in Table 5 to 8, and the ROC curves are depicted in Figure 18-21.

From our results it can be seen that decision tree performs best for all the datasets as it gives the most accurate classification result. The testing and training time is also less for decision tree algorithm as compared to other algorithms.

The reason decision tree performs best might be related to the chi-square feature selection method we use and the features selected by chi-square are favored by decision tree algorithm. Also, it is known that decision tree requires very less effort for data pre-processing when compared to other algorithms. But there is not a significant difference in the accuracy when we look at the results. Also, another observation from our results is deep learning-based approaches, MLPN and LSTM, performs not well on Kitsune and MedBIoT dataset. One of the possibilities might be the combined Mirai and other DDoS attack traffic in Kitsune dataset, and the virtual and simulated device traffic in MedBIoT dataset. Considering the training and testing time, deep learning-based approaches always take longer due to their complex nature.

4.2.1 Results for N-BaIoT Dataset

The table 5 and figure 18 below shows the results and ROC curve for N-BaIoT dataset respectively.

Algorithm	Accuracy	Precision	Recall	F1 Score	Training Time (Sec)	Testing Time (Sec)
Linear	94.8	94.2	99.1	96.3	55.32	3.2
K Nearest Neighbor	99.8	99	99.1	99	68.2	5.1
Naïve Bayes	62.2	99.3	56.3	71.2	2.2	1.4
Decision Tree	99.9	99.5	99.4	96.3	32.2	4.4
Random Forest	99.9	99.3	99.4	99.2	82.4	3.0
MLPN	92.8	94.4	97.3	95.3	100.4	8.4
LSTM	99.9	100	99	99	140.3	9.5

Table 5 – Results of N-BaIoT Dataset

Below are the results for comparing the results of decision tree algorithm without any feature selection method and results of some of the existing work.

Algorithm	Accuracy	Precision
Decision Tree (Without Feature Selection)	98.2	98
Linear [4]	80.9	80.4
Naïve Bayes [4]	45.6	88.4
KNN [4]	99.8	99.8

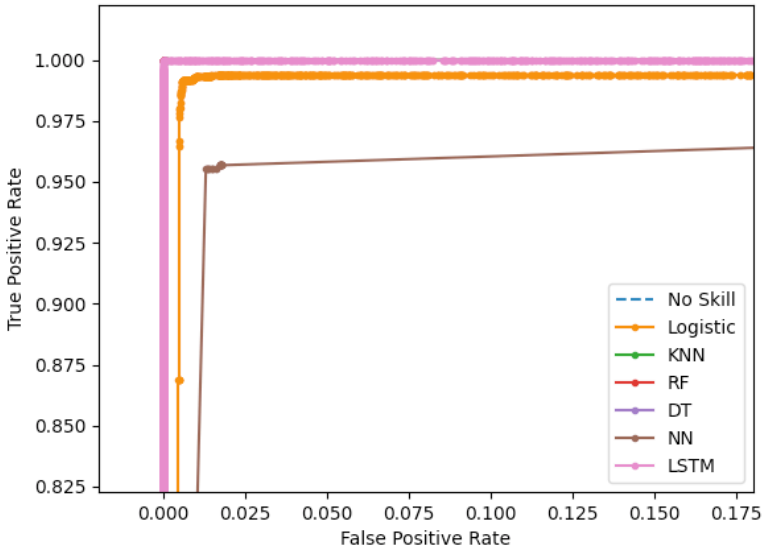


Figure 18. ROC curve for N-BaIoT dataset

4.2.2 Results for IoT-23 Dataset

The table 6 and Figure 19 below shows the results of IoT-23 Dataset and ROC curve for IoT-23 dataset respectively.

Algorithm	Accuracy	Precision	Recall	F1 Score	Training Time (Sec)	Testing Time (Sec)
Linear	99.5	99.6	99	99	42.1	1.3
K Nearest Neighbor	99.9	99	99	99	49.4	5
Naïve Bayes	99.5	99	99	99	12.5	1.8
Decision Tree	99.9	99	99	99	14.8	3.8
Random Forest	100	100	100	100	78.4	4.0
MLPN	99.6	99	100	99	410.4	9.4
LSTM	99.8	100	99	99	1427.7	89.7

Table 6. Results of IoT 23 Dataset

Below are the results indicating the Decision Tree algorithm without feature selection method and results of some of the existing work.

Algorithm	Accuracy	Precision
Decision Tree (Without feature Selection)	97.62	98
LSTM [27]	99.1	100
Random Forest [27]	91.1	93

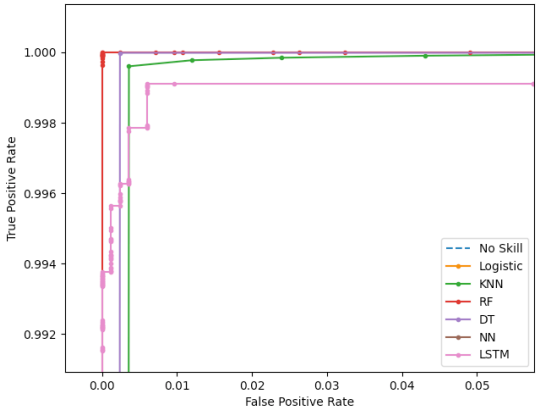


Figure 19. Result of IoT-23 Dataset

4.2.3 Kitsune Dataset

The Table 7 and Figure 20 below shows the results for Kitsune Dataset and ROC curve for kitsune dataset respectively.

Algorithm	Accuracy	Precision	Recall	F1 Score	Training Time (Sec)	Testing Time (Sec)
Linear	98.9	99	99	99	12.3	2.9
K Nearest Neighbor	99.9	99	99	99	47.1	5.3
Naïve Bayes	93.8	100	92	95	10.5	2.6
Decision Tree	100	100	100	100	31.9	13.4

Random Forest	100	100	100	100	88.4	5.0
MLPN	88.5	100	86	92	821.8	19.8
LSTM	91.1	99	89	93	2988.9	148.7

Table 7 – Result of Kitsune Dataset

Below are the results indicating the Decision Tree algorithm without feature selection method.

Algorithm	Accuracy	Precision
Decision Tree (Without feature Selection)	98.24	98.1

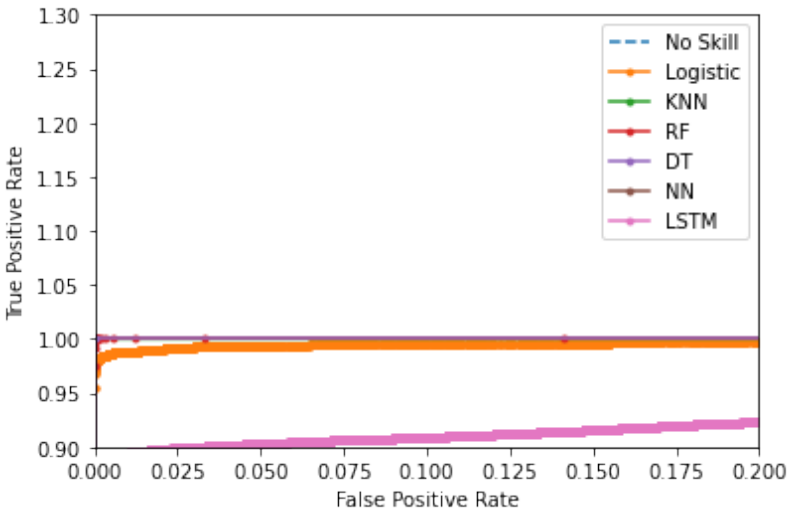


Figure 20. ROC curve for Kitsune Dataset

4.2.4 Results for MedBIoT Dataset

The Table 8 and Figure 21 below shows the Results for MedBIoT dataset and ROC curve for MedBIoT dataset respectively.

Algorithm	Accuracy	Precision	Recall	F1 Score	Training Time (Sec)	Testing Time (Sec)
Linear	86.8	87	94	91	21.2	2.4

K Nearest Neighbor	95.1	95	98	96	52.4	5.3
Naïve Bayes	87.6	85	98	91	22.4	11.2
Decision Tree	95.3	95	98	96	24.3	13
Random Forest	95.6	95	98	96	200.7	8.0
MLPN	89.1	91	91	91	612.9	19.0
LSTM	87.6	85	98	96	2987.8	84.5

Table 8. Results of MedBIoT Dataset

Below are the results indicating the Decision Tree algorithm without feature selection method and results of some of the existing work.

Algorithm	Accuracy	Precision
Decision Tree (Without feature Selection)	93.2	94.5
KNN [18]	90.25	90.82
Decision Tree [18]	93.15	94.48
Random Forest [18]	95.32	95.80

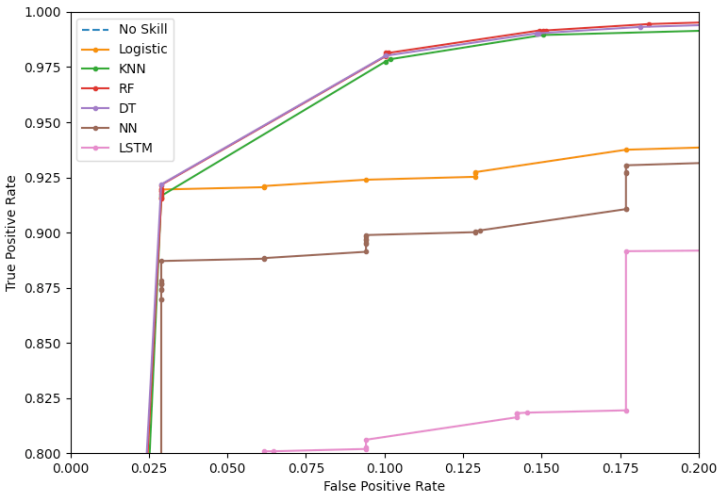


Figure 21. ROC curve for MedBIoT Dataset.

Chapter-5 Future Work and Conclusion

5.1 Future Work

In our study, our main aim was to evaluate the comparison study of various machine learning algorithms on already available new datasets. For our future works we would like to expand this project by collecting our own dataset that helps in overcoming many of the difficulties that we face in using other datasets. Also, it is clear from the study that there has been some work done in detecting the early detection of botnet attacks but we feel that there is a lot of work that can be done to accurately detect botnet attack in the early stages. Also, there are many other machine learning algorithms that can be evaluated to increase the efficiency of botnet detection.

5.2 Conclusion

From our research, we conclude that the methodology used for comparison in this paper has greatly served our purpose. We were able to compare the works of different papers in a tabulated form that provides a great clarity for further researchers. Also, we were able to find the accuracy of four different dataset using our comparison methods. We used Chi-Squared feature selection method to select the best features suited for our algorithms. We also used some of the best machine learning and deep learning algorithms that were best suited for our algorithms. In the end, we were able to compute the best accuracy for botnet detection that can be really helpful in the future research and provide a comprehensive comparison to carry out future works.

B. References

[1] Nadia Chaabouni, Mohamed Mosbah , Akka Zemmari, Cyrille Sauvignac, and Parvez Faruki``Network Intrusion Detection for IoT Security Based on Learning Techniques'' IEEE Communications Surveys \& Tutorials, Vol. 21, No. 3, Third Quarter 2019

[2] Ayush Kumar and Teng Joon Lim." EDIMA: Early Detection of IoT Malware Network Activity Using Machine Learning Techniques." 2019 IEEE 5th World Forum on Internet of Things (WF-IoT)

[3] Zhipeng Liu,Niraj Thapa,Addison Shaver,Kaushik Roy,Xiaohong Yuan and Sajad Khorsandroo ``Anomaly Detection on loT Network Intrusion Using Machine Learning"

[4] Shao-Chien Chen, Yi-Ruei Chen, and Wen-Guey Tzeng." Effective Botnet Detection Through Neural Networks on Convolutional Features." 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications.

[5] Rohan Doshi,Noah Apthorpe,Nick Feamster."Machine Learning DDoS Detection for Consumer Internet of Things Devices."2018 IEEE Symposium on Security and Privacy Workshops.

[6] Ruchi Vishwakarma, Ankit Kumar Jain."A Honeypot with Machine Learning based Detection Framework for defending IoT based Botnet DDoS Attacks."2019 International Conference on trends in Electronics and Informatics.

[7] Shamsul Haq,Yashwant Singh."Botnet Detection using Machine Learning."5th IEEE International Conference on Parallel, Distributed and Grid Computing(PDGC-2018), 20-22 Dec, 2018, Solan, India.

[8] Hadeel Alazzam,Abdulsalam Alsmady,Amaal Al Shorman."Supervised Detection of IoT Botnet Attacks."

[9] Mohit Goyal Ipsit Sahoo G. Geethakumari,"HTTP Botnet Detection in IOT Devices using Network Traffic Analysis."

[10] Sriram S, Vinayakumar Ry, Mamoun Alazabz, Soman KP. "Network Flow based IoT Botnet Attack Detection using Deep Learning."

[11] Xiaoyu Liang, Taieb Znati."A Long Short-term Memory anabled framework for DDoS Detcetion."

[12] Francisco Sales de Lima Filho ,Frederico A. F. Silveira ,Agostinho de Medeiros Brito Junior, Genoveva Vargas-Solar and Luiz F. Silveira."Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning." Hindawi Security and Communication Networks Volume 2019, Article ID 1574749.

[13] Huy-Trung Nguyen, Quoc-Dung Ngo, Doan-Hieu Nguyen, Van-Hoang Le."PSI-rooted subgraph:A novel feauture for IoT botnet detection using classifier algorithms. The Korean Institute of Communications and Information Sciences.

[14] Md Arafatur Rahman, A Taufiq Asyhari, L.S. Leong, G.B Satrya, M.Hai Tao. " Scalable machine learning based intrusion detection system for IoT-enabled smart cities."

[15] Gonzalo De La Torree Parra, Paul Rad, Kim-Kwang Raymond Choo, Nicole Beebe."Detectiong Internet of Things attack using distributed deep learning."

[16] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Dominik Breitenbacher, Asaf Shabtai and Yuval Elovici."N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders."

[17] Yisroel Mirsky, Tomer Doitshman and Asaf Shabtai."Kitsune: An Ensemble of autoencoders for online network intrusion detection."

[18] Alejandro Guerra-Manzanares,Jorge Medina-Galindo, Hayretdin Bahsi and Sven Nomm."MedBIoT: Generation of an IoT botnet datset in a Medium-sized IoT Network."

[19] Sumaiya Thaseen Ikram, Aswani Kumar, Cherukuri." Intrusion detection model using fusion of Chi-Square feature selection multi class SVM."

[20] Sumaiya Thaseen, Aswani Kumar, Ameer Ahmed." Integrated Intrusion Detection Model Using Chi-Square Feature Selection and Ensemble of Classifiers."

[21] Kambria (https://kambria.io/blog/logistic-regression-for-machine-learning/)

[22] Medium (https://medium.com/capital-one-tech/k-nearest-neighbors-knn-algorithm-for-machine-learning-e883219c8f26)

[23] Datacamp (https://www.datacamp.com/community/tutorials/decision-tree-classification-python)

[24] Toward Data Science (https://towardsdatascience.com/understanding-random-forest-58381e0602d2)

[25] Modeling Languages (https://modeling-languages.com/lstm-neural-network-model-transformations/)

[26] Emsisoft (https://blog.emsisoft.com/en/27233/what-is-a-botnet/)

[27] Vibekananda Dutta, Michał Chora´s, Marek Pawlicki and Rafał Kozik. “A Deep Learning Ensemble for Network Anomaly and Cyber-Attack Detection”