Botnet Detection in IoT Devices using Gradient and Ada Boosting Algorithm

MSc Research Project
Cybersecurity

Sririshi Veeranam Shanmugam
Student ID: X21167672

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	……. Sririshi Veeranam Shanmugam …………………………………………………………………………
Student ID:	…………x21167672………………………………………………………………………………………
Programme:	………Cybersecurity……………………………………………… **Year:** ……2022……………
Module:	………Research Project…………………………………………………………………………
Supervisor:	……………Michael Pantridge………………………………………………………………
Submission Due Date:	………………15/12/2022………………………………………………………………………
Project Title:	Botnet Detection in IoT Devices using Gradient and Ada Boosting Algorithm………………………………………………
Word Count:	………4997…………………………… **Page Count**……………20………………………

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	……Sririshi Veeranam Shanmugam………………………………………………
Date:	………15-12-2022……………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Botnet Detection in IoT Devices using Gradient and Ada Boosting Algorithm

Sririshi Veeranam Shanmugam (x21167672)
National College of Ireland, Mayor Street, IFSC, Dublin 1, Ireland

Abstract:

Most people think of "Internet of Things" (IoT) gadgets as unusual computers that can send and receive data across a network through wireless connections. As a result of the convenience they provide, IoT devices have become commonplace in people's daily routines. Considering that the vast majority of IoT gadgets aren't computers, they lack basic security features. It's because of this that hackers target people's IoT gadgets in an effort to get their hands on their passwords and financial information. In this study, we use the public IoT botnet dataset to investigate the prevalence of botnets in IoT devices and propose a machine learning approach for detecting them quickly and accurately. In this study, we employed the Gradient boosting technique to efficiently process a massive dataset while maintaining high standards of precision, throughput, and detection. Also, the Ada boosting algorithm has been integrated for a higher prediction speed in the botnets.

Keywords: IoT (Internet of Things) devices, Attack, Botnet, detection, Boosting algorithm, Gradient Boost, Ada Boost

1 Introduction:

The ground-breaking technology known as the Internet of Things (IoT) enables any object in the world to be linked to the web and speak with one another in real time. A recent analysis estimates that "the world of IoT" will be worth $1567 million by the year 2025. It predicted that by 2030 there might be 75 billion connected devices.

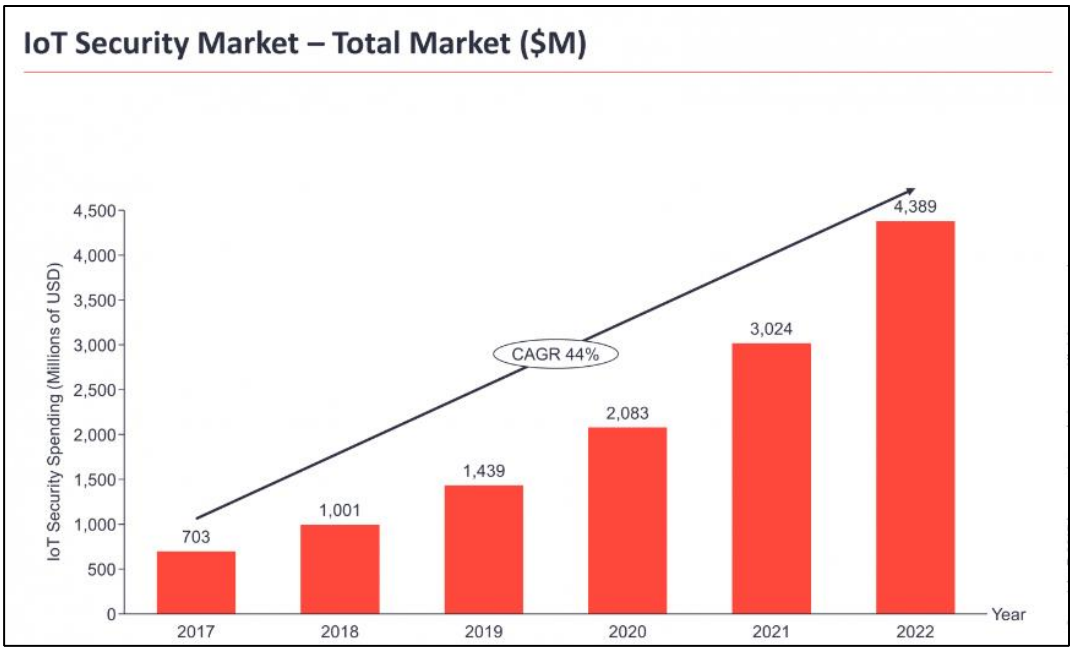


Fig 1: IoT Security market

The graph states that the global security market is expected to grow at a healthy clip between 2017 and 2022, as seen by the following graph. The significant danger of impacting IoT devices has contributed to the growth of the IoT security industry from $703 million in 2017 to $4.4 billion in 2022.

However, a network-based approach is preferable for protecting IoT devices from cyberattacks. These are attacks are classified into three types basically:

1. **Signature based detection method**
2. **Anomaly based detection method**
    i. **Statistics based detection**
    ii. **Machine learning based detection method**
    iii. **Knowledge based detection method**
3. **Specific based detection method**

1. **Signature based detection method:**

   Detects and stops attacks by comparing network traffic to a database of rules.
2. **Anomaly based detection method:**

   Develops a profile for each device on the network based on an analysis of its regular network traffic pattern. Any glaring deviation from the usual is considered an anomaly. More specifically, it may be broken down into

i. **Statistics based detection method:**

   By analyzing the statistical distribution of dangers, this technique was able to uncover abnormalities.

ii. **Machine Learning Based detection method:**

   The packets and payload characteristics are used to identify anomalies. Typically, this approach uses machine learning to detect and thwart attacks.

iii. **Knowledge based detection method:**

   Here, network knowledge is used to spot unusual occurrences. This information is obtained from the many test cases employed to uncover network anomalies.

3. **Specific based detection method:**

   It's a user-defined collection of criteria that can spot outliers.

While signature-based approaches to botnet identification in IoT devices have proven effective in the past, the experiments conducted here evaluate and detect both known and undiscovered botnets.

In this study, we employed Boosting algorithms to rapidly and accurately identify the botnets. Boosting algorithms, such as the Gradient boosting method and the Ada boosting algorithm, have been employed. In this case, we apply the gradient boosting approach to quickly and accurately discover patterns in a big dataset. The Ada boosting algorithm uses the predicted values derived from the gradient boosting approach.

This study provides a solid introduction to and background on the topic of botnet detection in IoT devices by reviewing the work of prior researchers in the field. Then, the research procedures and criteria are outlined.

2 Related works:

This section elucidates the other researchers' efforts to identify botnets in the IoT through the use of devices. The literature review provides a comprehensive explanation of the research methodology, methods, and future potential of the study. In this research, we provide many methods for identifying botnets in the Internet of Things. This section describes the research process and the researchers' constraints. Based on the research and analysis, botnet detection is broken down into three distinct categories: machine learning-based, deep learning-based, and hybrid feature-based botnet identification. Following is a study and investigation of the relevant literature, informed by the work already done on Botnet detection in the Internet of Things.

A. Machine learning used for Botnet Detection:

In this study (Garg, Kaushik, Panwar, & Gupta, 2021) we provide an examination of machine learning technique for IoT botnet. Researchers have used a machine learning system to distinguish between typical and aberrant traffic patterns. A wide range of machine learning algorithms, including Random Forest IG, Random Forest Gini, Decision Tree, SVM, Logical Regression, and Gaussian NB, have been employed. The UNSW-NB15 dataset has been used for the analysis utilizing these machine learning algorithms. Analysis of Random Forest reveals that Gini achieves 92.65% accuracy, whereas Gaussian NB achieves only 50.46 %.

In this study (Wang et al., 2020), the authors present a method for automating botnet identification based on the traffic's flow- and graph-based characteristics. Using a combination of flow analysis and graph theory, the researchers developed an automated model called "BotMark" to identify botnets. They have developed a botnet detection system by extracting three graph-based characteristics and fifteen statistical flow-based traffic features. They thought about how comparable and stable C- flow measurement is for flow-based detection. Using the least square method and LOF (local outlier factor), graph-based detection may compute an anomaly score to highlight any discrepancies between neighbouring nodes. The researchers have combined five recently established botnets to simulate extremely high volumes of network traffic for their studies. Detection accuracy is at 99.94%.

IoT botnet detection using a frequency-based dependency graph is presented in this work (Yassin et al., 2019). Scientists have gathered evidence of a botnet attack's suspicious registry entries. In this case, the researcher has used the graph approach to examine the mirai assault. Using the dependency graph, they were able to determine what parts of the botnet were responsible for the attack and what parts were unrelated. Seeing that they mimic his every move, they have deduced his strategy. This discovery is put to use in order to detect otherwise undetected botnets in the Internet of Things.

This work uses botnet fingerprinting to detect botnets (Blaise, Bouet, Conan, & Secci, 2020). BotFP is one detection method presented by researchers; it is employed for assessing host behavior using frequency distribution signatures. Clustering and supervised machine learning may be used to understand how benign hosts and bots behave. The CTU-13 dataset is used to verify the BotFP by the research community. This dataset includes 13 different infection scenarios, such as establishing a command and control channel and starting a DDoS assault or port scan. BotFP's small size and low computational cost make it suitable for large datasets, allowing it to be used with impressive precision. In addition, researchers have developed two Bot FP, BotFP-Clus for overfitting host signs and BotFP-ML for learning from the algorithm and analyzing and detecting new bots. Thus, this delivers results near to 100%, since it has identified all 13 bots in the datasets where it is deemed lightweight compared to graph-based methods.

In this study (Soe et al., 2020), the authors suggest a sequential architecture and machine learning method for detecting IoT botnets. Researchers have implemented a machine learning method with sequential design to identify attacks in an Internet of Things setting. The N-BaIoT dataset is used for the study, and associated feature selection is employed to cut down on extraneous information. Using an artificial neural network, naive bayes, and a J48 decision tree—three distinct machine learning algorithms—we were able to increase the total detection performance to 99%.

In this study (Jeelani, Rai, Maithani, & Gupta, 2022), the authors suggest a machine learning-based method for detecting IoT botnets. The experiment was carried out using the IoT-23 dataset. The places where they've utilized naive bayes, SVM, a decision tree, or a convolutional neural network (Convolutional neural network). Naive Bayes performed the lowest of the compared machine learning algorithms, whereas the decision tree achieved the highest level of accuracy (73%) in the shortest amount of time (less than 10 seconds).

In this study (Prokofiev, Smirnova, & Surov, 2018), the authors present a strategy for identifying an Internet of Things botnet. An IoT botnet detection algorithm based on Logistic Regression has been created by researchers. Accuracy, precision, recall, and the F-measure will be used to evaluate the success of the suggested strategy.

In this study (Malik et al., 2022), they present a method for detecting botnets based on a single categorization. This technique use a one-class KNN classifier to detect IoT botnets early on and with high precision. The researchers developed a lightweight approach that selects characteristics by utilizing popular filter and wrapper methods. Many datasets gathered from various networks are used to test the researcher's proposed technique. Using the filter and wrapper technique, we were able to shrink the feature space by as much as 72% across all datasets, freeing up computational resources in the process. The suggested technique has been shown to detect IoT botnets with a success rate of 98% to 99% and an F1 score of 0.9 or higher.

In their publication, (Siboni & Cohen, 2020), the authors suggested a method for detecting anomalies in individual sequences with practical implications. Anomaly detection for one-dimensional time series has been employed by researchers; this method can learn the system's behavior and then raise an alarm if that behavior suddenly changes. The Lempel-Ziv algorithm has been implemented to effectively analyze and categorize system data for behavioral analysis. They observed that employing this technique, botnets might be recognized without requiring extensive analysis.

B. Deep Learning used for Botnet Detection:

The authors of this research (Nguyen, Ngo, & Le, 2019) present a lightweight approach or procedure for identifying IoT (internet of things) botnets by extracting a high-level feature from various function-call graphs using a technique called PSI-Graph. For the purpose of this study, the multi-architectural challenge is addressed by employing a characteristic that allows researchers to sidestep the control flow graph employed by conventional methods. The dataset used in this study consists of 11,200 ELF files and comprises 7199 examples of IoT botnets, with an accuracy of 98.7%.

In this study (Nugraha, Nambiar, & Bauschert, 2020), the authors suggest a deep learning technique for gauging botnet detection efficiency. In order to conduct their study, researchers have employed four distinct deep learning approaches, including Convolutional Neural Networks (CNNs), Long Short-Term Memories (LSTMs), a hybrid CNN-LSTM, and Multilayer Perceptrons (MLPs) (Multi Layer Perception). The studies performed on the CTU-13 botnet traffic dataset. They have retained several metrices, such as accuracy, sensitivity, specificity, precision, and F1 score, to test for identifying known and unknown botnet traffic patterns. This study's findings demonstrate that a deep learning model is capable of reliably identifying both well-known and previously novel forms of botnet data traffic.

In this study (De La Torre Parra, Rad, Choo, & Beebe, 2020), the authors suggest employing distributed deep learning to identify attacks on the Internet of Things. The researchers have utilized a distributed deep learning algorithm that runs on the cloud to identify phishing and botnet attempts. In this study, we employ DCNN (Distributed Convolutional Neural Network) and LSTM (Long Short-Term Memory) as two of the security key mechanisms (Long Short-Term Memory). While LSTM is stored on the host's backend and used to detect botnet attacks, DCNN is utilized in IoT devices to identify phishing attacks. The suggested CNN model was trained using a dataset containing both phishing and non-phishing URLs, and the NBaIoT dataset was used for backend training of LSTM in the context of the botnet. The experiment shows that the accuracy for detecting phishing URLs is 94.3% and the F1 score is 93.58%. With an accuracy of 94.80 percent for the LSTM model on the back end, the procedure may be utilized to spot intrusion attempts.

In this study, the authors (Ahmed, Jabbar, Sadiq, and Patel, 2020) offer a methodology for identifying a botnet assault. This technique has been used to the study of zero-day botnet attacks and their detection in real time. Researchers employ the following procedures to identify botnet assaults utilizing the suggested deep learning DNN and the ANN feed- forward propagation technique: dataset selection, feature selection, training, data normalization, validation, and testing. The investigation was conducted using the CTU-13 dataset. According to the results of this experiment, detecting botnets can be done with 99.6 percent accuracy.

In this research (Maeda et al., 2019), the authors suggest utilizing deep learning to detect botnets on SDNs. Malware traffic data obtained from the current network has been evaluated, and normal and malicious traffic has been categorised using deep learning. To stop host infections and spot collateral harm in infected hosts, they propose using deep learning to the SDN (software-defined network). As a result, the researchers not only created an isolation network to protect against internal infection, but they also blocked connections to the outside network. In the end, a detection rate of 99.2% is discovered, which is higher than that of any other available technique.

In order to improve the detection of botnet assaults in numerous sensors, the authors of this study (Hezam et al., 2021) recommend merging deep learning models. The BiLSTM-CNN model combines LSTM and CNN. CNN is used for data optimization while BiLSTM is utilized for classification. The experiment is run using the N-BaIoT dataset, yielding a success rate of 89.79%, an error rate of 0.1546, and a precision rate of 93.92%. Its 89.50% accuracy rate is lowest of all classifiers when compared to that of the CNN.

Network-based anomaly detection with deep auto encoder was proposed in a recent research (Meidan et al., 2018). For the experiment, researchers infected nine consumer-grade IoT gadgets using popular IoT botnets like Mirai Bashlite. Using Local outline factor, support vector machine, and Isolation Forest, the researchers were able to evaluate how well each method performed in detecting the attack.

C. Hybrid model used for Botnet Detection:

In this study (Desai, Shi, & Suo, 2021), the authors provide a combined strategy for detecting IoT Botnets. Multiple supervised and unsupervised machine learning algorithms have been employed by researchers, creating what is known as a hybrid model. In these settings, the supervised and unsupervised algorithm will be used together to identify the latest assault in the stream of communication. The unsupervised algorithm is constructed first, and then the findings of the experiment are fed into the supervised machine learning algorithm. K-means clustering serves as the unsupervised method, while Decision Tree provides the supervision.

In this research, we present a hybrid approach to botnet identification based on host and network analysis (Almutairi, Mahfoudh, Almutairi, & Alowibdi, 2020). Researchers have built up their concept on both the host and network ends. They have included IP hopping to cover botnet-related communication protocols like HTTP, P2P, IRC, and DNS. In this case, they've employed a different algorithm, the HANABot algorithm, to preprocess the trustworthy behavior of the botnet in order to extract the necessary characteristic for the botnet's activity. As a result, the experiment is conducted using a real-world dataset collection that includes both harmful and reliable data. The end product has a very small false positive rate and a high accuracy. A network analyzer now keeps tabs on how far a botnet has spread and how often bots are communicating with their C&C server. Over time, a host analyzer will track changes to registry keys, files, and active processes to provide a detection report on compromised systems.

As a means of identifying the IoT botnet, the authors of a forthcoming study (Memos and Psannis, 2020) suggest a hybrid AI (Artificial Intelligence) honeypot. Researchers have turned to cloud computing to identify IoT botnets. Additionally, the presence or absence of the IoT botnet may be predicted using Logical Regression (LR). Good efficiency in terms of precision, accuracy, recall, fall-out, and F-measure are achieved by combining artificial intelligence, machine learning algorithm, and cloud computing with honeypot.

In this study (Francois et al., 2011), they propose utilizing MapReduce to identify botnets. The researchers employed a distributed computing framework based on a host dependence model and a modified version of the page rank algorithm. Using a Hadoop cluster, they were able to obtain the desired outcome, and the advantages of use actual network traces were discussed. Good detection, accuracy, and efficiency utilizing Hadoop clusters have been demonstrated by the research community.

3 Methodology:

In this study, the public datasets are utilized to identify IoT devices that are part of a botnet. Here python and pandas is used for coding where in using google collab to execute the code, After the datasets have been imported from the computer necessary library files have been installed and data being pre-processed, they will be divided in half for use in both development and testing. After that, it goes through a series of machine learning techniques including the gradient boosting method and the Ada boosting algorithm, before being put through an evaluation phase. (An Introduction to the Gradient Boosting Algorithm) (2022). The process of methodology goes by initialization and pre-processing, splitting of dataset and giving the split dataset into the machine learning algorithm and then model process and gives the result of the accuracy rate of detection.

The primary idea behind gradient boosting algorithms is to construct a new model based on the error information obtained from an existing model in order to lower the existing model's error. It helps with faster, more accurate identification of massive, complicated data.

The Ada boosting algorithm is a machine learning ensemble method that may be used to correct the errors introduced by the gradient boosting approach.

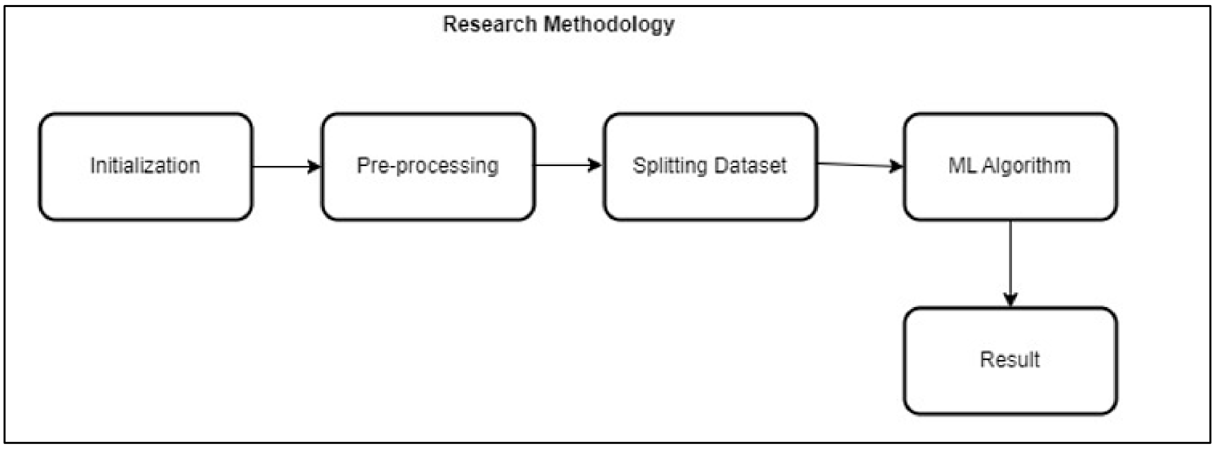


Fig 2: Research Methodology

4 Design Specification:

The things how it works is shown in the above research methodology diagram. For this there are several way of process to be followed to execute. That process is explained below

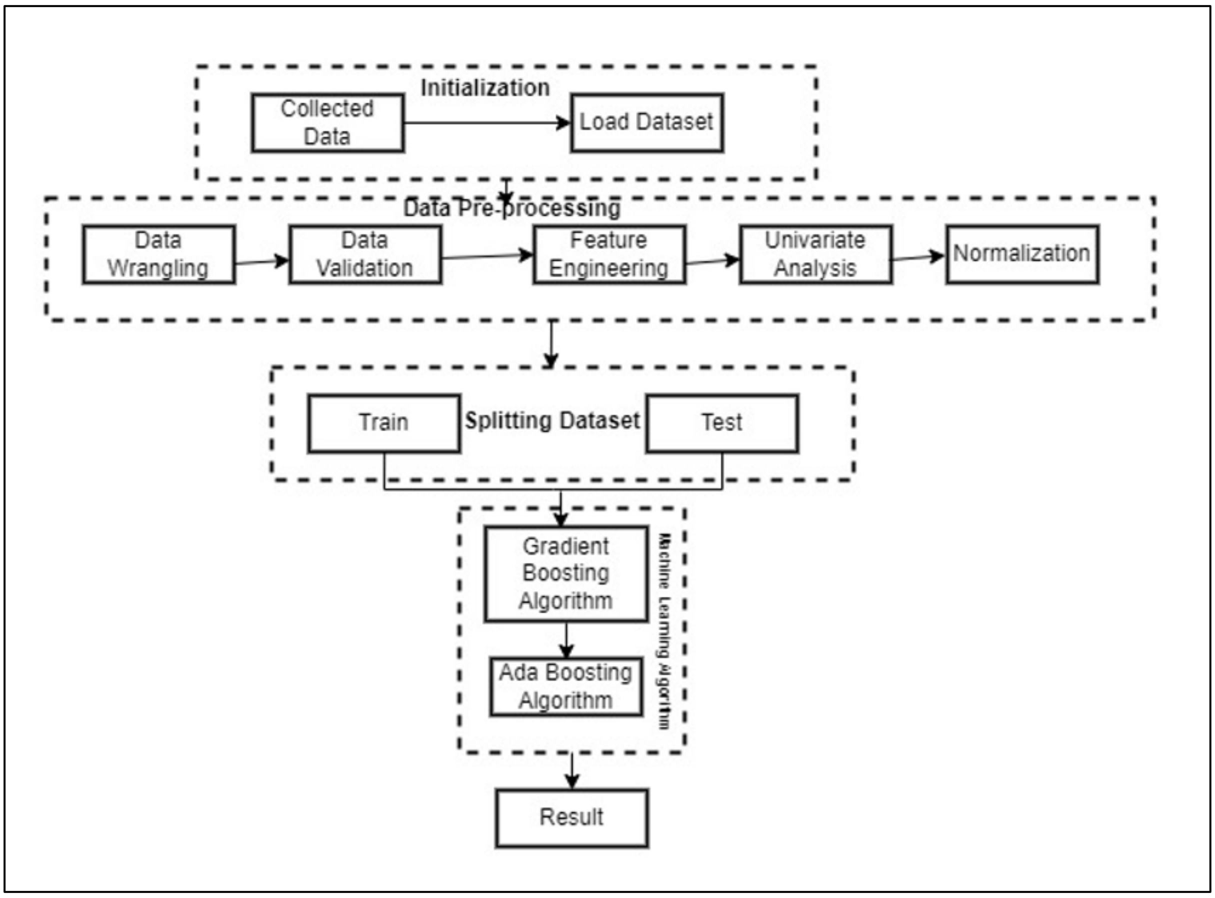


Fig 3: Design Specification

i. ***Initialization:***

Initialization is the process of collection or downloading the public dataset from the Internet and loading into our programming code and making arrangements for the next step.

ii. ***Data Pre-processing:***

The first step in pre processing is removing duplicate records and data from the dataset, as doing so simplifies the analysis and shortens the processing time this is called as Data wrangling after the data validation process is carried out , following that feature engineering process takes place after that univariate analysis is taken place where it gives the statistical analysis of the dataset of each category, then normalization takes place where to make the uneven dataset into even dataset.

iii. ***Splitting the dataset:***

During this step, the dataset is often subdivided for use in experiments. As we divide, for example, to conduct separate training and evaluation sessions, we find that we need to break into two groups. Perhaps we can split into training and testing groups of 50%.
After this is complete, the testing dataset is forwarded to the machine learning algorithm stage, and the results are evaluated in the next step.

iv. ***Machine Learning algorithms:***

Machine learning algorithm used here is Gradient boosting and Ada boosting algorithm where - it works by sequentially adding predictors to an ensemble, each one correcting its predecessor. However, instead of tweaking the instance weights at every iteration and in AdaBoost method tries to fit the new predictor to the residual errors made by the previous predictor.

v. ***Evaluation:***

The obtained results are made in accuracy ,precision ,recall ,F-measure.

Accuracy:

The term "accuracy" refers to the proportion of right forecasts to total forecasts.

$$Accuracy = \frac{TP+TN}{TP + FN + FP + TN}$$

Precision:

Precision is the fraction of true positives that are actually calculated.

$$Precision = \frac{True\ positives}{True\ positives + False\ Positives}$$

Recall:

A subject's recall is a calculation of the number of true positives she or he was able to identify.

$$\mathit{Recall} = \frac{\mathit{True\ positives}}{\mathit{True\ positives} + \mathit{False\ Negative}}$$

F-measure:

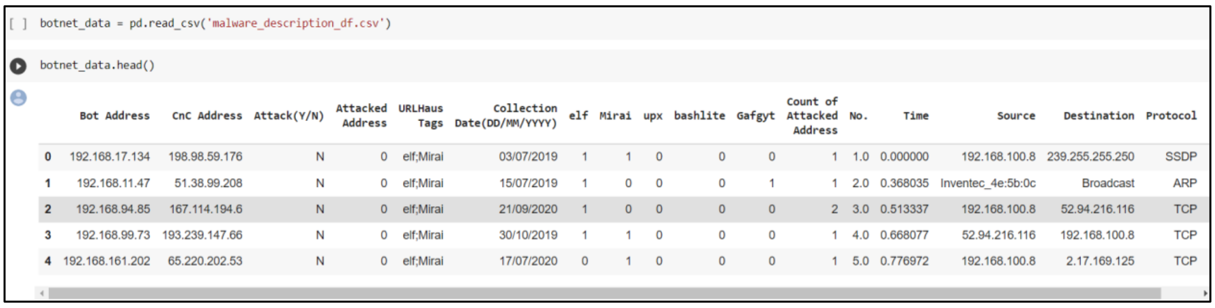
This metric represents a ratio of how well a test was completed. The measuring sticks for this examination are known as "test precision" and "test recall."

$$\mathit{F\text{-}\ measure} = 2 * \frac{\mathit{Precision} * \mathit{Recall}}{\mathit{Precision} + \mathit{Recall}}$$

5 Implementation:

The implementation process is which it consist of the step by step process of the botnet detection .

The first process consists of downloading the dataset and reading it in our programming code.



```
[ ] botnet_data = pd.read_csv('malware_description_df.csv')
```

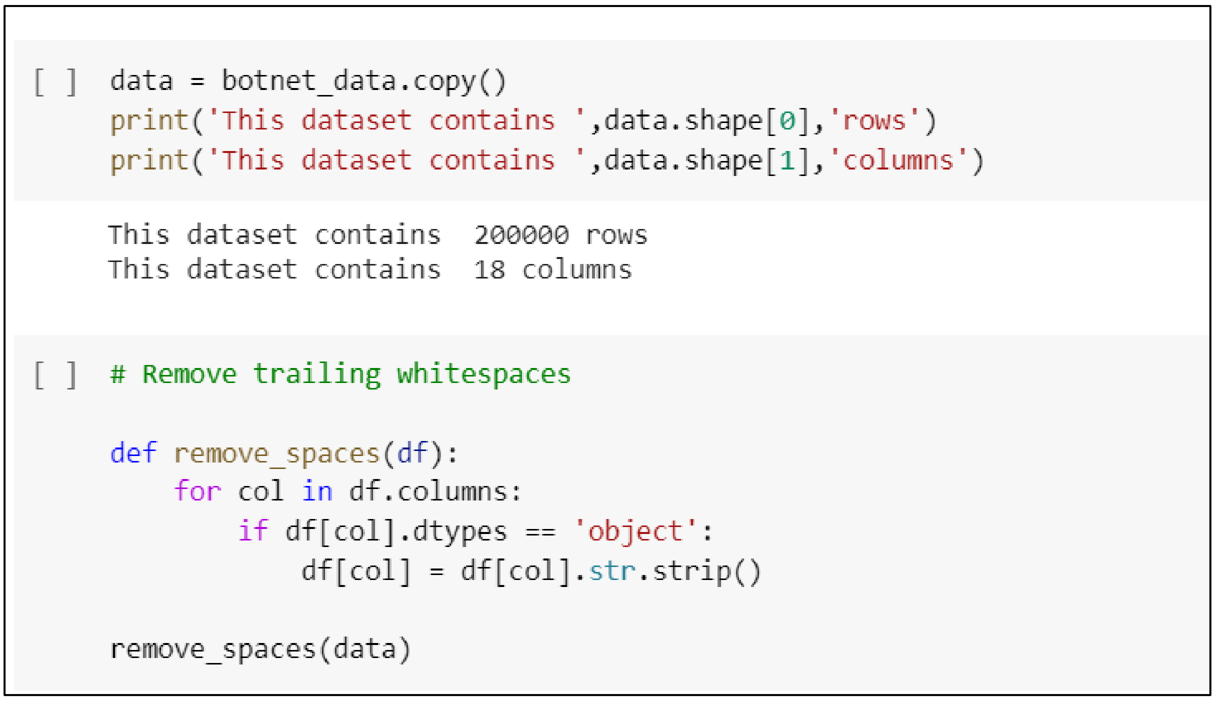
```
botnet_data.head()
```

	Bot Address	CnC Address	Attack(Y/N)	Attacked Address	URLHaus Tags	Collection Date(DD/MM/YYYY)	elf	Mirai	upx	bashlite	Gafgyt	Count of Attacked Address	No.	Time	Source	Destination	Protocol
0	192.168.17.134	198.98.59.176	N	0	elf;Mirai	03/07/2019	1	1	0	0	0	1	1.0	0.000000	192.168.100.8	239.255.255.250	SSDP
1	192.168.11.47	51.38.99.208	N	0	elf;Mirai	15/07/2019	1	0	0	0	1	1	2.0	0.368035	Inventec_4e:5b:0c	Broadcast	ARP
2	192.168.94.85	167.114.194.6	N	0	elf;Mirai	21/09/2020	1	0	0	0	0	2	3.0	0.513337	192.168.100.8	52.94.216.116	TCP
3	192.168.99.73	193.239.147.66	N	0	elf;Mirai	30/10/2019	1	1	0	0	0	1	4.0	0.668077	52.94.216.116	192.168.100.8	TCP
4	192.168.161.202	65.220.202.53	N	0	elf;Mirai	17/07/2020	0	1	0	0	0	1	5.0	0.776972	192.168.100.8	2.17.169.125	TCP

Fig 4: Uploading the dataset

In the above figure shows the collection of dataset and it has been read through the programming code.

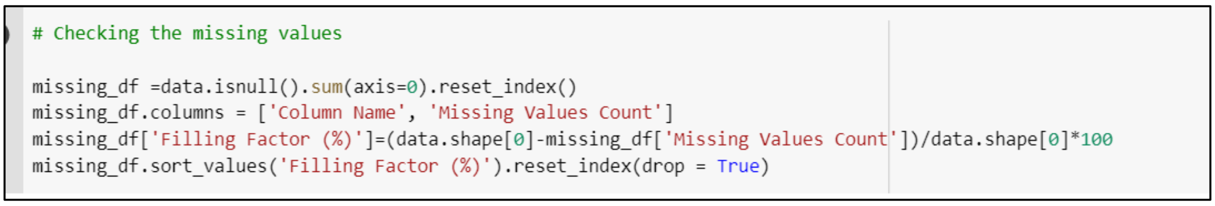
Now before pre-processing is done where it consists of data wrangling where it finds for loading the data, check for cleanliness and trimming process.



```
[ ]  data = botnet_data.copy()
     print('This dataset contains ',data.shape[0],'rows')
     print('This dataset contains ',data.shape[1],'columns')

     This dataset contains  200000 rows
     This dataset contains  18 columns

[ ]  # Remove trailing whitespaces

     def remove_spaces(df):
         for col in df.columns:
             if df[col].dtypes == 'object':
                 df[col] = df[col].str.strip()

     remove_spaces(data)
```

Fig 5: Code for removing unwanted space

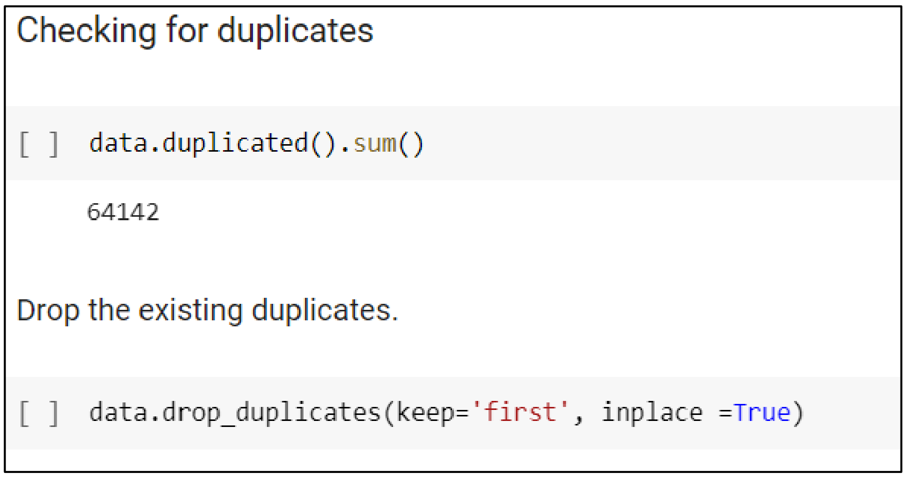
Removing the unwanted spaces in the dataset while creating



```
# Checking the missing values

missing_df =data.isnull().sum(axis=0).reset_index()
missing_df.columns = ['Column Name', 'Missing Values Count']
missing_df['Filling Factor (%)']=(data.shape[0]-missing_df['Missing Values Count'])/data.shape[0]*100
missing_df.sort_values('Filling Factor (%)').reset_index(drop = True)
```

Fig 6: Code for finding the missing values

This code is applied for finding the missing values in the dataset



Checking for duplicates

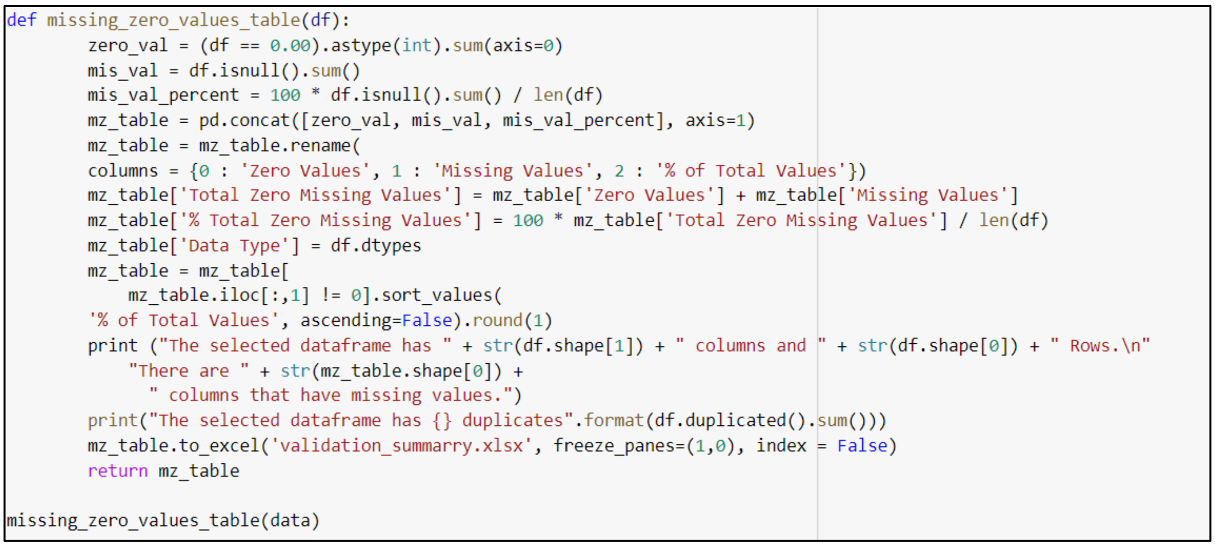
```
[ ]  data.duplicated().sum()

     64142
```

Drop the existing duplicates.

```
[ ]  data.drop_duplicates(keep='first', inplace =True)
```

Fig 7: Code for checking Duplicates

This code is written for the purpose of checking the duplicate value and deleting the duplicate values in the code

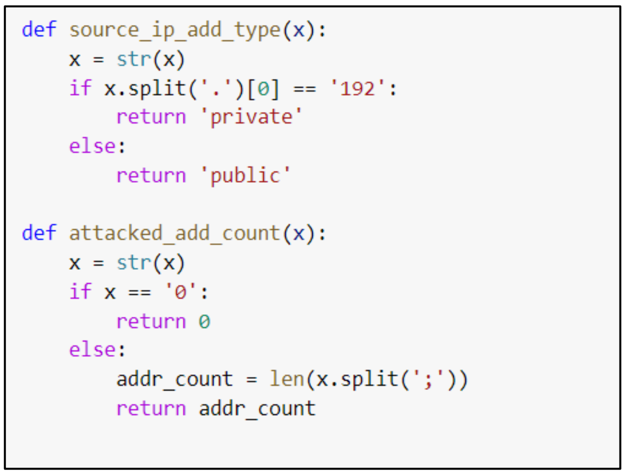


```python
def missing_zero_values_table(df):
        zero_val = (df == 0.00).astype(int).sum(axis=0)
        mis_val = df.isnull().sum()
        mis_val_percent = 100 * df.isnull().sum() / len(df)
        mz_table = pd.concat([zero_val, mis_val, mis_val_percent], axis=1)
        mz_table = mz_table.rename(
        columns = {0 : 'Zero Values', 1 : 'Missing Values', 2 : '% of Total Values'})
        mz_table['Total Zero Missing Values'] = mz_table['Zero Values'] + mz_table['Missing Values']
        mz_table['% Total Zero Missing Values'] = 100 * mz_table['Total Zero Missing Values'] / len(df)
        mz_table['Data Type'] = df.dtypes
        mz_table = mz_table[
            mz_table.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)
        print ("The selected dataframe has " + str(df.shape[1]) + " columns and " + str(df.shape[0]) + " Rows.\n"
            "There are " + str(mz_table.shape[0]) +
              " columns that have missing values.")
        print("The selected dataframe has {} duplicates".format(df.duplicated().sum()))
        mz_table.to_excel('validation_summarry.xlsx', freeze_panes=(1,0), index = False)
        return mz_table

missing_zero_values_table(data)
```

Fig 8: Code for Data validation

Data validation have been done with the process of this above mentioned code.

Feature engineering process is carried out as for the addition, manipulation and deletion which is used to increase the accuracy and performance of the model.

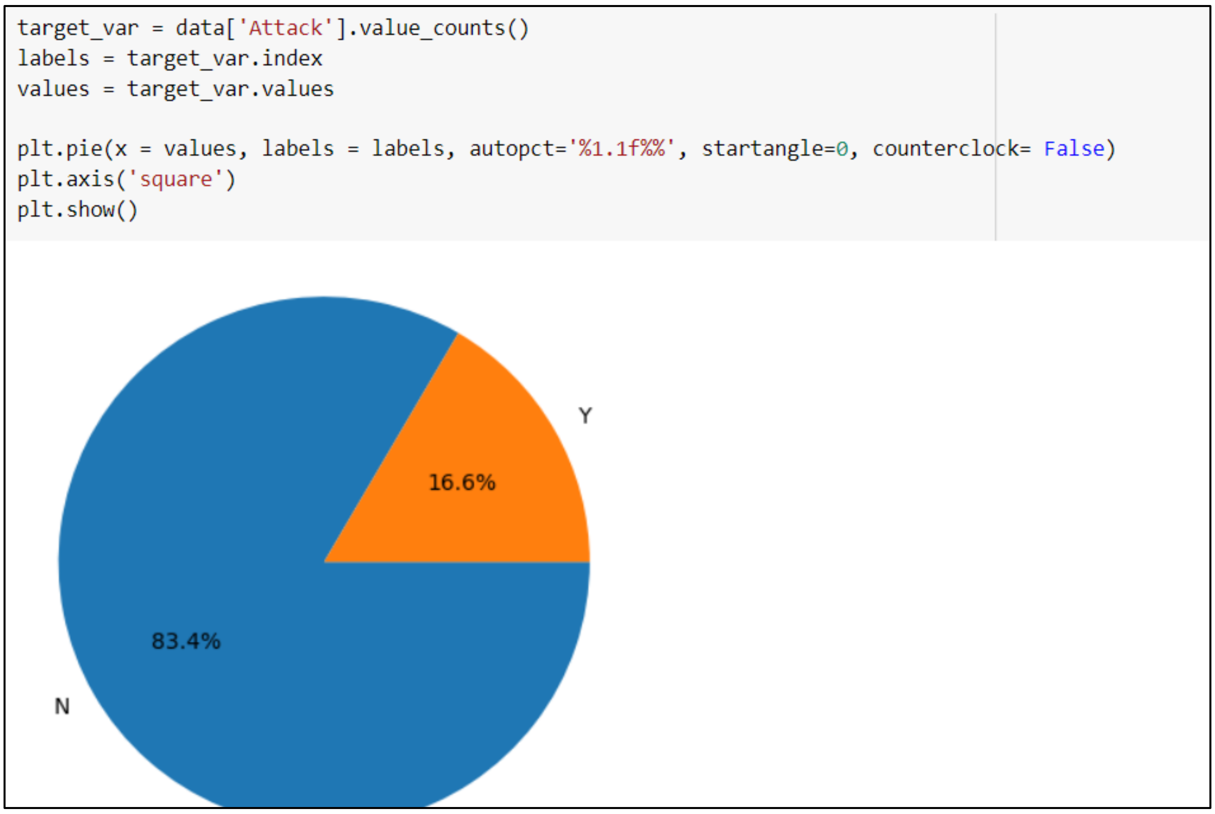


```python
def source_ip_add_type(x):
    x = str(x)
    if x.split('.')[0] == '192':
        return 'private'
    else:
        return 'public'

def attacked_add_count(x):
    x = str(x)
    if x == '0':
        return 0
    else:
        addr_count = len(x.split(';'))
        return addr_count
```

Fig 9: Code for Feature Engineering

Feature engineering is done with the above-mentioned code.

Now we are going to use the univariate analysis so that it can be used to show the statistical form of the dataset according to the attack, protocol, CNC, source ip address, below mentioned graph are the following analysis from the dataset.



```
target_var = data['Attack'].value_counts()
labels = target_var.index
values = target_var.values

plt.pie(x = values, labels = labels, autopct='%1.1f%%', startangle=0, counterclock= False)
plt.axis('square')
plt.show()
```

Fig 10: Code and Representation of Statistical analysis of the attack

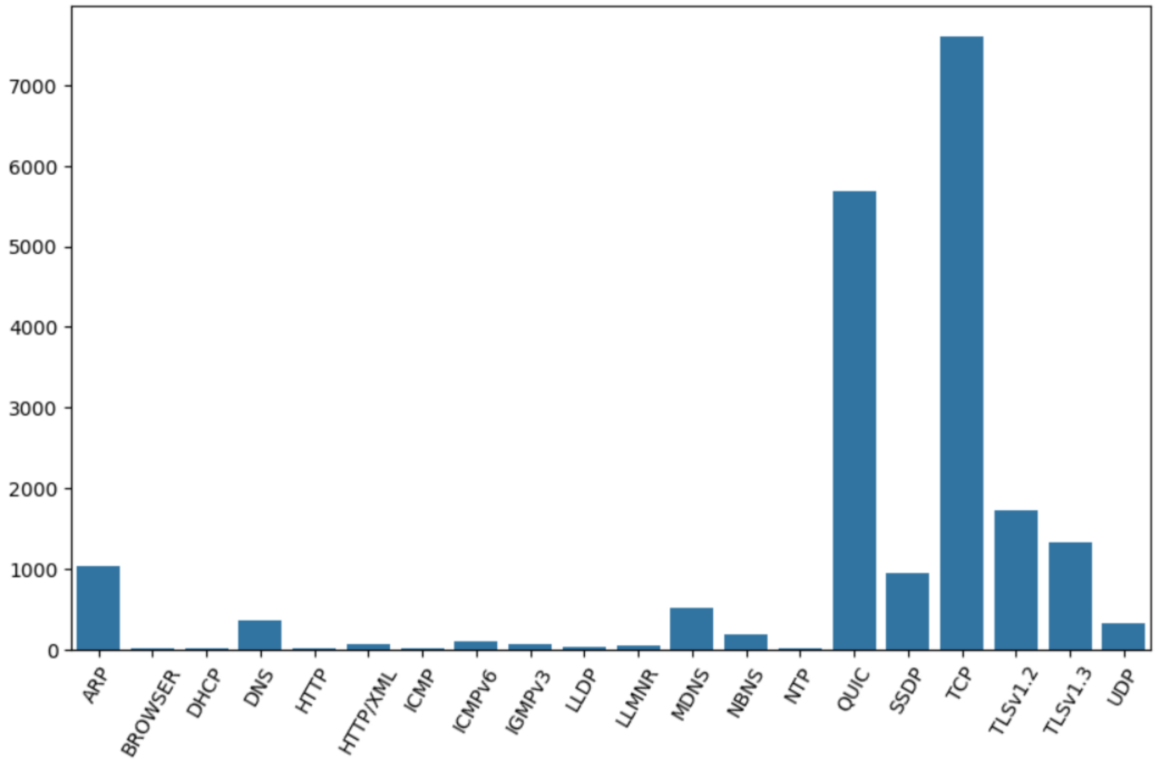


Fig 11: Representation of Statistical analysis of the protocols

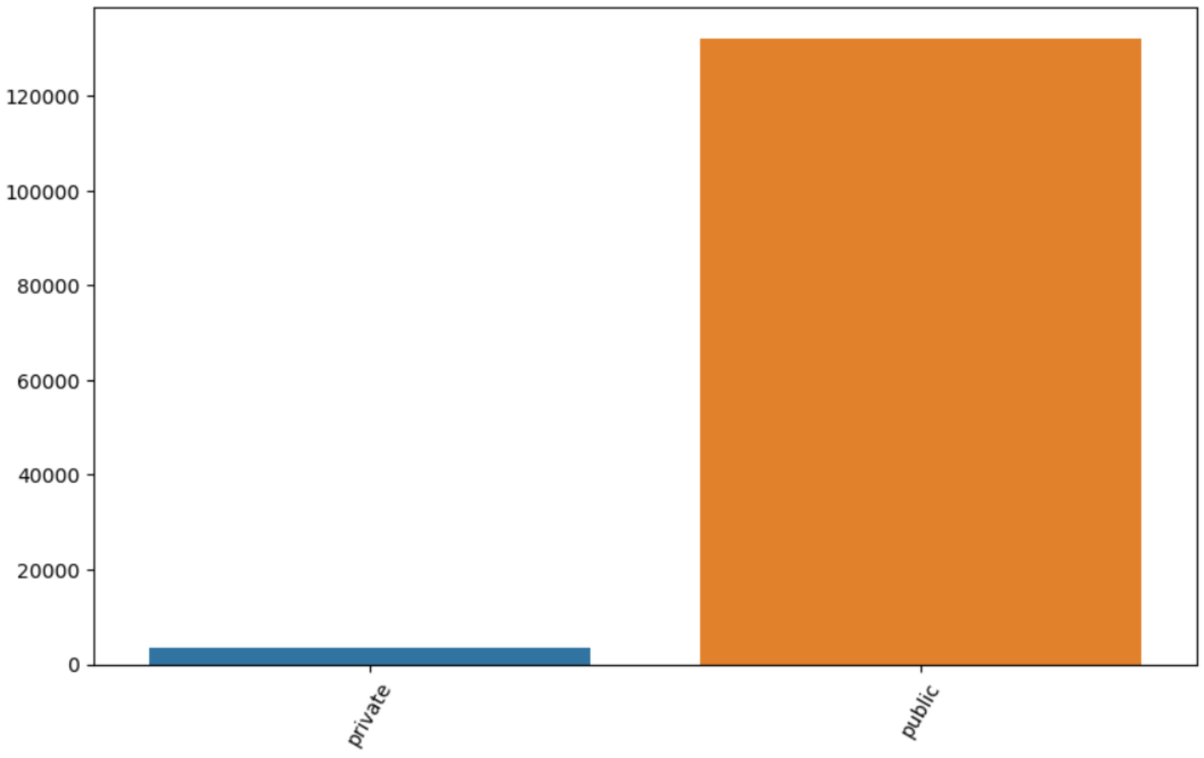
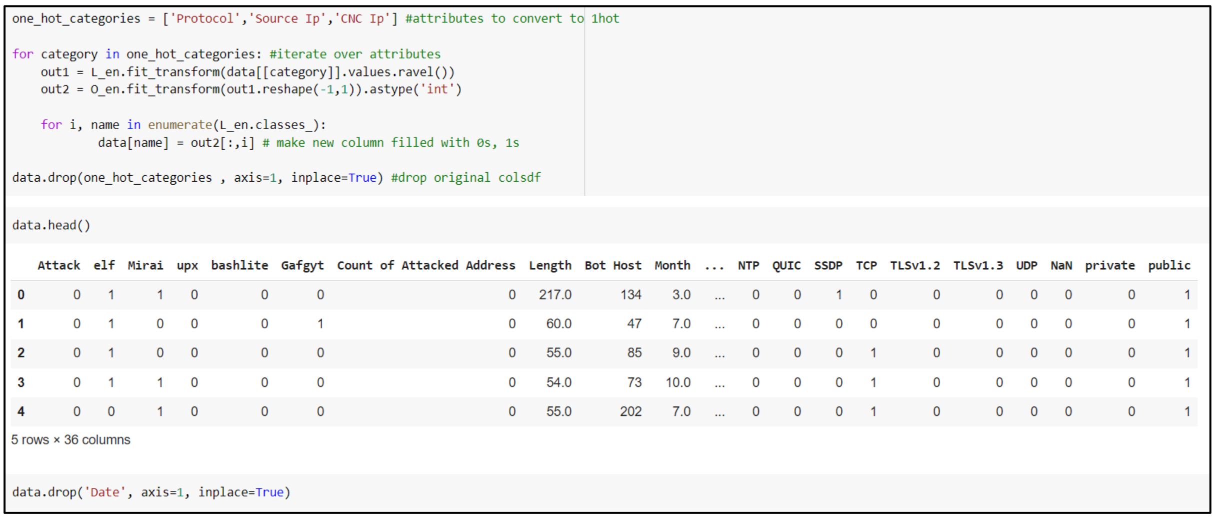


Fig 12: Representation of statistical analysis of the CNC address from the datasets.

Now on coming the actual pre-processing of the dataset where we convert all the target variable into the binary



```
one_hot_categories = ['Protocol','Source Ip','CNC Ip'] #attributes to convert to 1hot

for category in one_hot_categories: #iterate over attributes
    out1 = L_en.fit_transform(data[[category]].values.ravel())
    out2 = O_en.fit_transform(out1.reshape(-1,1)).astype('int')

    for i, name in enumerate(L_en.classes_):
            data[name] = out2[:,i] # make new column filled with 0s, 1s

data.drop(one_hot_categories , axis=1, inplace=True) #drop original colsdf
```

```
data.head()
```

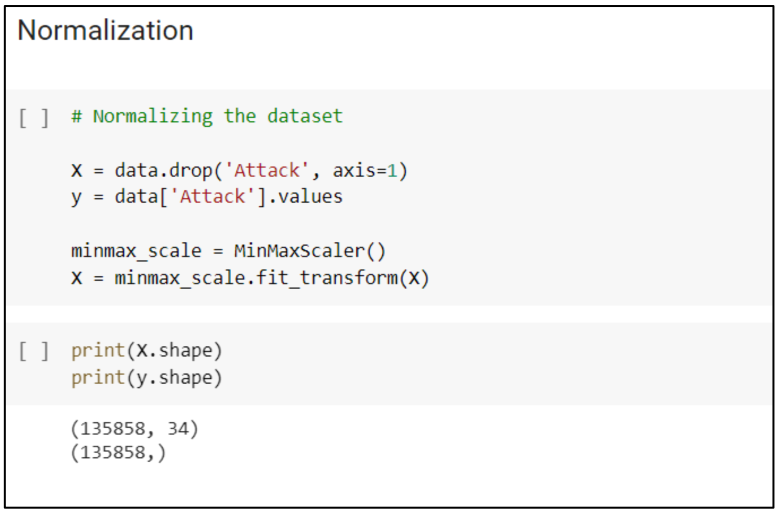
	Attack	elf	Mirai	upx	bashlite	Gafgyt	Count of Attacked Address	Length	Bot Host	Month	...	NTP	QUIC	SSDP	TCP	TLSv1.2	TLSv1.3	UDP	NaN	private	public
0	0	1	1	0	0	0	0	217.0	134	3.0	...	0	0	1	0	0	0	0	0	0	1
1	0	1	0	0	0	1	0	60.0	47	7.0	...	0	0	0	0	0	0	0	0	0	1
2	0	1	0	0	0	0	0	55.0	85	9.0	...	0	0	0	1	0	0	0	0	0	1
3	0	1	1	0	0	0	0	54.0	73	10.0	...	0	0	0	1	0	0	0	0	0	1
4	0	0	1	0	0	0	0	55.0	202	7.0	...	0	0	0	1	0	0	0	0	0	1

5 rows × 36 columns

```
data.drop('Date', axis=1, inplace=True)
```

Fig 13: Code for Pre-processing

This code shows the pre processing of the dataset that converting the variable into integer. After the pre-processing then the normalisation is done.



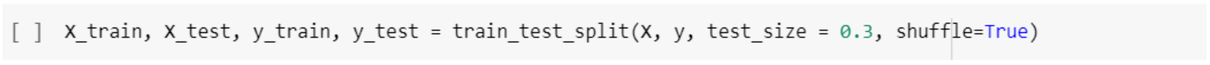
Normalization

```
[ ] # Normalizing the dataset

    X = data.drop('Attack', axis=1)
    y = data['Attack'].values

    minmax_scale = MinMaxScaler()
    X = minmax_scale.fit_transform(X)
```

```
[ ] print(X.shape)
    print(y.shape)

    (135858, 34)
    (135858,)
```

Fig 14: Code for Normalization

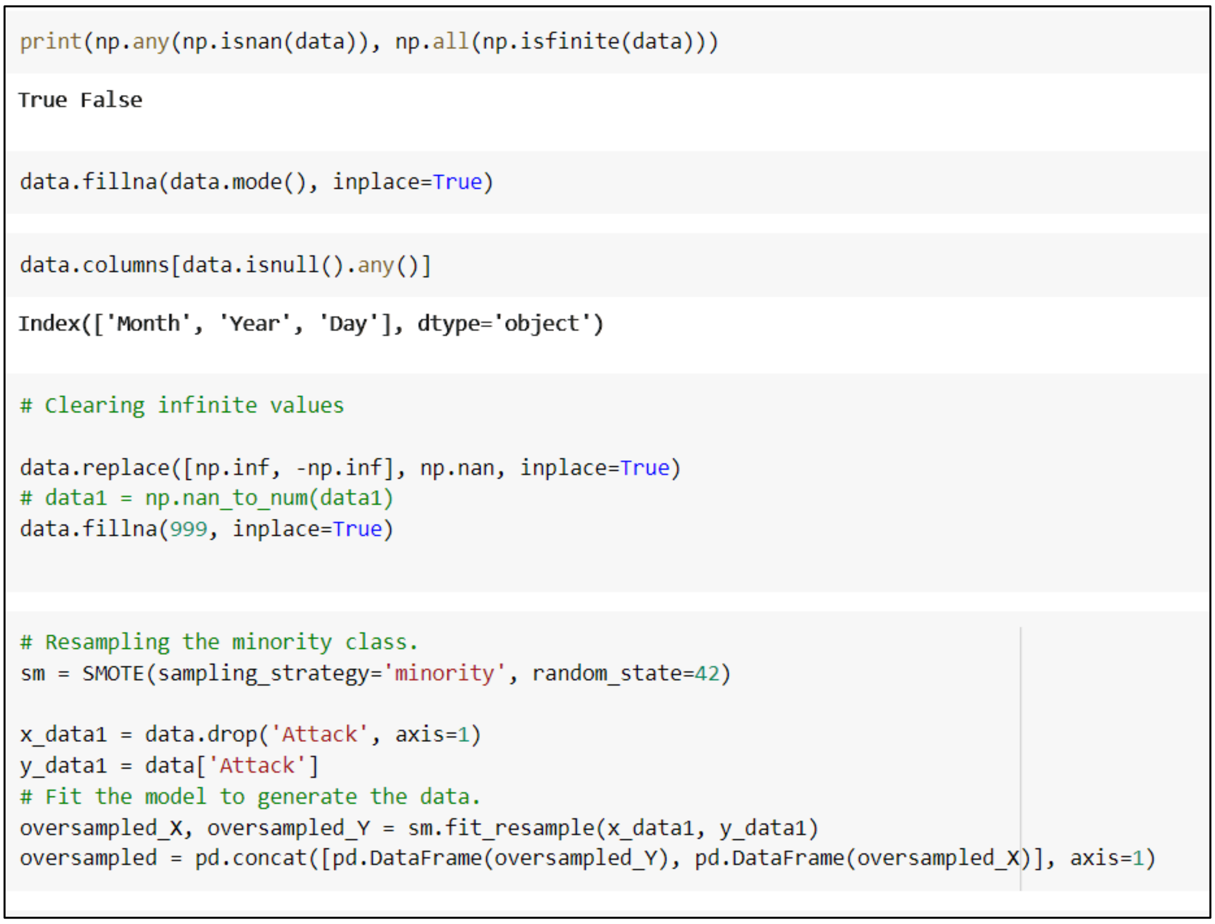
Now splitting the datasets into two for training and splitting:



```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, shuffle=True)
```

Fig 15: Code for Splitting Dataset

Using above code splitting the dataset into two for train and test.



```
print(np.any(np.isnan(data)), np.all(np.isfinite(data)))

True False
```

```
data.fillna(data.mode(), inplace=True)
```

```
data.columns[data.isnull().any()]

Index(['Month', 'Year', 'Day'], dtype='object')
```

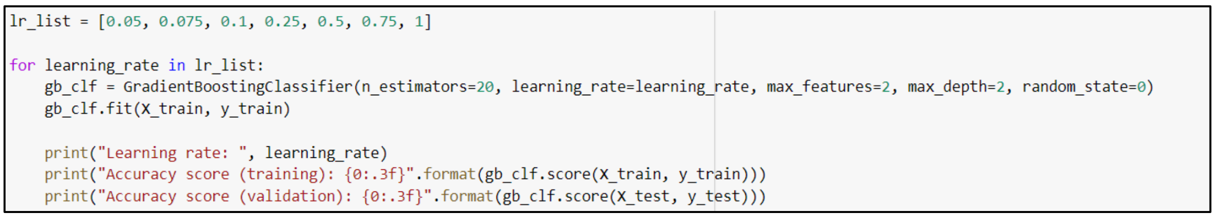
```
# Clearing infinite values

data.replace([np.inf, -np.inf], np.nan, inplace=True)
# data1 = np.nan_to_num(data1)
data.fillna(999, inplace=True)
```

```
# Resampling the minority class.
sm = SMOTE(sampling_strategy='minority', random_state=42)

x_data1 = data.drop('Attack', axis=1)
y_data1 = data['Attack']
# Fit the model to generate the data.
oversampled_X, oversampled_Y = sm.fit_resample(x_data1, y_data1)
oversampled = pd.concat([pd.DataFrame(oversampled_Y), pd.DataFrame(oversampled_X)], axis=1)
```

Fig 16: Code to imbalance target variable

The above code is used to handle the imbalance of the target variable

At last now putting our dataset into the our models to train and test with the following code.

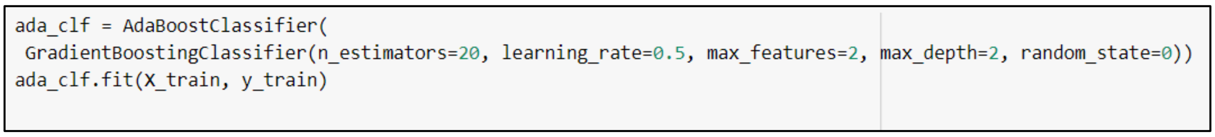


```
lr_list = [0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1]

for learning_rate in lr_list:
    gb_clf = GradientBoostingClassifier(n_estimators=20, learning_rate=learning_rate, max_features=2, max_depth=2, random_state=0)
    gb_clf.fit(X_train, y_train)

    print("Learning rate: ", learning_rate)
    print("Accuracy score (training): {0:.3f}".format(gb_clf.score(X_train, y_train)))
    print("Accuracy score (validation): {0:.3f}".format(gb_clf.score(X_test, y_test)))
```

Fig 17: Code for Gradient boosting algorithm

Gradient Boosting code is used to find the evaluation process

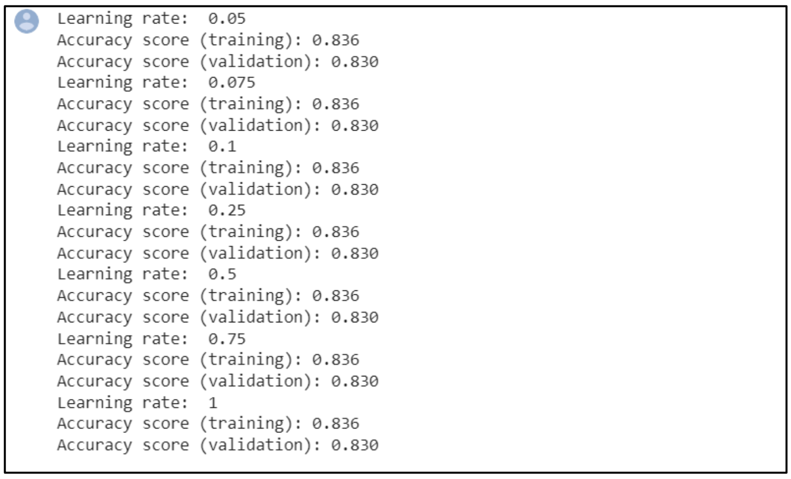
Now separately doing for the Ada boost algorithm with the result of gradient boosting



```
ada_clf = AdaBoostClassifier(
 GradientBoostingClassifier(n_estimators=20, learning_rate=0.5, max_features=2, max_depth=2, random_state=0))
ada_clf.fit(X_train, y_train)
```

Fig 18: Code for Ada boosting algorithm

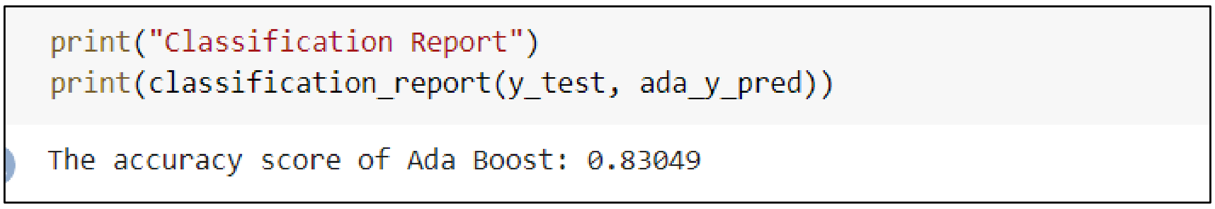
Hence the dataset is successfully loaded into the model to give the accuracy of the dataset



```
Learning rate:  0.05
Accuracy score (training): 0.836
Accuracy score (validation): 0.830
Learning rate:  0.075
Accuracy score (training): 0.836
Accuracy score (validation): 0.830
Learning rate:  0.1
Accuracy score (training): 0.836
Accuracy score (validation): 0.830
Learning rate:  0.25
Accuracy score (training): 0.836
Accuracy score (validation): 0.830
Learning rate:  0.5
Accuracy score (training): 0.836
Accuracy score (validation): 0.830
Learning rate:  0.75
Accuracy score (training): 0.836
Accuracy score (validation): 0.830
Learning rate:  1
Accuracy score (training): 0.836
Accuracy score (validation): 0.830
```

Fig 19: Accuracy of Gradient boosting

The above-mentioned accuracy value is for the gradient boosting algorithm



```
print("Classification Report")
print(classification_report(y_test, ada_y_pred))
```

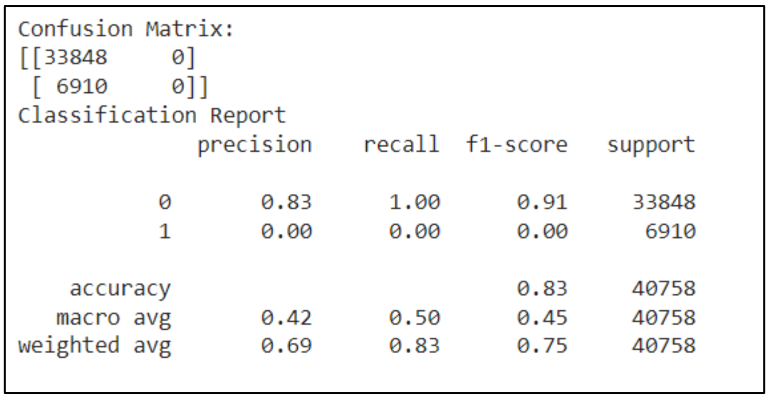
```
The accuracy score of Ada Boost: 0.83049
```

Fig 20: Accuracy of Ada Boosting algorithm

Hence the accuracy of the Ada boost is shown.

6 Evaluation:

The accuracy, precision, recall and F1 score is evaluated by training and testing the model.

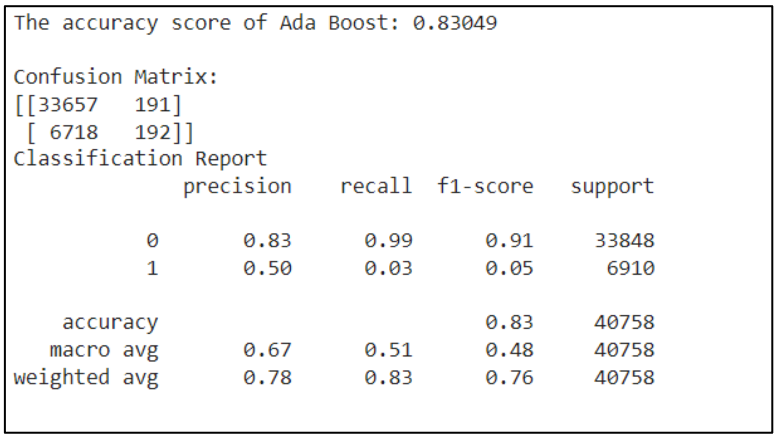


```
Confusion Matrix:
[[33848     0]
 [ 6910     0]]
Classification Report
              precision    recall  f1-score   support

           0       0.83      1.00      0.91     33848
           1       0.00      0.00      0.00      6910

    accuracy                           0.83     40758
   macro avg       0.42      0.50      0.45     40758
weighted avg       0.69      0.83      0.75     40758
```

Fig 21: confusion matrix and result of Gradient boosting algorithm

From the above diagram we come to know the gradient boosting algorithm have the accuracy of 0.83 or 83%, and the F1 score is 0.91, recall value is 1.00, and precision is 83% shown

After this again the model evaluates with the Ada boosting algorithm and the confusion matrix of that is given below.



```
The accuracy score of Ada Boost: 0.83049

Confusion Matrix:
[[33657   191]
 [ 6718   192]]
Classification Report
              precision    recall  f1-score   support

           0       0.83      0.99      0.91     33848
           1       0.50      0.03      0.05      6910

    accuracy                           0.83     40758
   macro avg       0.67      0.51      0.48     40758
weighted avg       0.78      0.83      0.76     40758
```

Fig 22: confusion matrix and result of Ada boosting algorithm

After evaluating the gradient boosting result with the Ada boosting algorithm gives the result of accuracy is 83%, F1 score of 91%, recall value of 99% and precision value of 83%. This gives the clear note that combing the two-machine algorithm gives the higher evaluation metrics.

Hence It proves our research question that botnet detection in iot devices using the machine algorithm such as gradient and Ada boosting algorithms combined.

7 Conclusion and Future work:

The combination of Gradient boosting and Ada boosting algorithm helps in boosting the failed models and train them to give the clear values, hence this study helps in the prediction of botnet in Iot devices network traffic can be efficiently detected with the greater accuracy value. For the future work we can able to build a model that which helps in indicating the changes in network traffic will be found and alert the user or user enterprises.

References:

[1] Nguyen, H., Ngo, Q. and Le, V., 2019. A novel graph-based approach for IoT botnet detection. *International Journal of Information Security*, 19(5), pp.567-577.

[2] Wang, W., Shang, Y., He, Y., Li, Y. and Liu, J., 2020. BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors. *Information Sciences*, 511, pp.284-296.

[3] Yassin, W., Abdullah, R., Abdollah, M., Mas'ud, Z. and Bakhari, F., 2019. An IoT Botnet Prediction Model Using Frequency based Dependency Graph. *Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City*,.

[4] Almutairi, S., Mahfoudh, S., Almutairi, S. and Alowibdi, J., 2020. Hybrid Botnet Detection Based on Host and Network Analysis. *Journal of Computer Networks and Communications*, 2020, pp.1-16.

[5] Blaise, A., Bouet, M., Conan, V. and Secci, S., 2020. Botnet Fingerprinting: A Frequency Distributions Scheme for Lightweight Bot Detection. *IEEE Transactions on Network and Service Management*, 17(3), pp.1701-1714.

[6] Soe, Y., Feng, Y., Santosa, P., Hartanto, R. and Sakurai, K., 2020. Machine Learning-Based IoT-Botnet Attack Detection with Sequential Architecture. *Sensors*, 20(16), p.4372.

[7] Nugraha, B., Nambiar, A. and Bauschert, T., 2020. Performance Evaluation of Botnet Detection using Deep Learning Techniques. *2020 11th International Conference on Network of the Future (NoF)*,.

[8] De La Torre Parra, G., Rad, P., Choo, K. and Beebe, N., 2020. Detecting Internet of Things attacks using distributed deep learning. *Journal of Network and Computer Applications*, 163, p.102662.

[9] Ahmed, A., Jabbar, W., Sadiq, A. and Patel, H., 2020. Deep learning-based classification model for botnet attack detection. *Journal of Ambient Intelligence and Humanized Computing*, 13(7), pp.3457-3466.

[10] Maeda, S., Kanai, A., Tanimoto, S., Hatashima, T. and Ohkubo, K., 2019. A Botnet Detection Method on SDN using Deep Learning. *2019 IEEE International Conference on Consumer Electronics (ICCE)*,.

[11] Jeelani, F., Rai, D., Maithani, A. and Gupta, S., 2022. The Detection of IoT Botnet using Machine Learning on IoT-23 Dataset. *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*,.

[12] Garg, U., Kaushik, V., Panwar, A. and Gupta, N., 2021. Analysis of Machine Learning Algorithms for IoT Botnet. *2021 2nd International Conference for Emerging Technology (INCET)*,.

[13] Desai, M., Shi, Y. and Suo, K., 2021. A Hybrid Approach for IoT Botnet Attack Detection. *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*,.

[14] Memos, V. and Psannis, K., 2020. AI-Powered Honeypots for Enhanced IoT Botnet Detection. *2020 3rd World Symposium on Communication Engineering (WSCE)*,.

[15] Prokofiev, A., Smirnova, Y. and Surov, V., 2018. A method to detect Internet of Things botnets. *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*,.

[16] Malik, K., Rehman, F., Maqsood, T., Mustafa, S., Khalid, O. and Akhunzada, A., 2022. Lightweight Internet of Things Botnet Detection Using One-Class Classification. *Sensors*, 22(10), p.3646.

[17] Hezam, A., Mostafa, S., Baharum, Z., Alanda, A. and Salikon, M., 2021. Combining Deep Learning Models for Enhancing the Detection of Botnet Attacks in Multiple Sensors Internet of Things Networks. *JOIV : International Journal on Informatics Visualization*, 5(4), p.380.

[18] Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D. and Elovici, Y., 2018. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Computing*, 17(3), pp.12-22.

[19] Siboni, S. and Cohen, A., 2020. Anomaly Detection for Individual Sequences with Applications in Identifying Malicious Tools. *Entropy*, 22(6), p.649.

[20] Francois, J., Wang, S., Bronzi, W., State, R. and Engel, T., 2011. BotCloud: Detecting botnets using MapReduce. *2011 IEEE International Workshop on Information Forensics and Security*,.

[21] Borges, J., Medeiros, J., Barbosa, L., Ramos, H. and Loureiro, A., 2022. IoT Botnet Detection based on Anomalies of Multiscale Time Series Dynamics. *IEEE Transactions on Knowledge and Data Engineering*, pp.1-1.

[22] 2022. [online] Available at: <https://www.mygreatlearning.com/blog/adaboost-algorithm/> [Accessed 30 July 2022].

[23] Analytics Vidhya. 2022. *Gradient Boosting Algorithm: A Complete Guide for Beginners*. [online] Available at: <https://www.analyticsvidhya.com/blog/2021/09/gradient-boosting-algorithm-a-complete-guide-for-beginners/#h2_2> [Accessed 30 July 2022].

[24] IEEE DataPort. 2022. *UNSW_NB15 dataset*. [online] Available at: <https://ieee-dataport.org/documents/unswnb15-dataset> [Accessed 30 July 2022].

[25] Arxiv.org. 2022. [online] Available at: <https://arxiv.org/ftp/arxiv/papers/2101/2101.05067.pdf> [Accessed 30 July 2022].

[26] Jatit.org. 2022. [online] Available at: <http://www.jatit.org/volumes/Vol96No15/28Vol96No15.pdf> [Accessed 30 July 2022].

[27]Hamid, Y., Ranganathan, B., Journaux, L. and Muthukumarasamy, S., 2022. [online] resarchgate.net. Available at: <https://www.researchgate.net/publication/324601933_Benchmark_Datasets_for_Network_Intrusion_Detection_A_Review> [Accessed 30 July 2022].

[28] Modi, A., 2022. *Botnet Attacks: How IoT Devices become Part/Victim of such Attacks*. [online] Einfochips.com. Available at: <https://www.einfochips.com/blog/botnet-attacks-how-iot-devices-become-part-victim-of-such-attacks/> [Accessed 1 August 2022].

[29] 1, C., Security, F., Security, C., Automation, T., SPE, S. and efficiency, R., 2022. *An overview of the IoT Security Market Report 2017-2022 – IIoT World*. [online] IIoT World. Available at: <https://www.iiot-world.com/news/reports/an-overview-of-the-iot-security-market-report-2017-2022/> [Accessed 1 August 2022].

[30] Hussain, F., Abbas, S., Pires, I., Tanveer, S., Fayyaz, U., Garcia, N., Shah, G. and Shahzad, F., 2021. A Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks. *IEEE Access*, 9, pp.163412-163430.