International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(REVIEW ARTICLE)

Check for updates

Botnet attack detection in IoT using machine learning models

Tewogbade Shakir Adeyemi * and Ajasa Muhammed

Researcher with CAPE economic research and consulting US.

International Journal of Science and Research Archive, 2024, 12(01), 2221–2229

Publication history: Received on 16 April 2024; revised on 26 May 2024; accepted on 29 May 2024

Article DOI: https://doi.org/10.30574/ijsra.2024.12.1.0936

Abstract

Botnet is of great concern when dealing with security of computer networks globally. Bots readily attack network infrastructure through their malicious activities. It is pertinent to mitigate and control the level of threat posit by Bot and thus the need for advanced technologies in predicting their occurrences. Machine learning offered a greater support in this regard with ability to handle voluminous data from IoT devices and the robustness in predicting the potential attack from trained data. Both supervised (DT and RF) and unsupervised learning (Deep Learning) were used to investigate prediction of attack. Results of various machine learning models were compared along the performance metrics (Confusion Matrix, ROC Curves). The outcome showed that supervised learning did better with the study dataset than the unsupervised model.

Keywords: Botnet; Machine Learning; Supervised; Unsupervised; Deep Learning

1. Introduction

Botnets have posited as one of the most dangerous threats to the security and integrity of computer networks worldwide. These sophisticated networks of compromised computers, known as bots, are designed by attackers to execute venomous actions like distributed denial-of-service (DDoS) attacks, spam circulation, fraud (data, financial). With rapid development in ICT circle, the challenge is detection and reduce attacks through botnet in the cyber landscape, as this will be requiring advanced techniques that can identify the inherent patterns and behaviors exhibited by these malicious entities. Ab inition, botnet detection has been achieved through signature-based methods where detection system checks for known patterns of malicious activity, such as specific network traffic or file signatures. However, signature-based methods are not always effective against new botnets, as attackers are constantly operating new tactics. Machine learning (ML) has demonstrated great potential in effectively identifying and countering botnet attacks in recent times. Its strength lies in its capacity to analyze vast amounts of data (big data) and reveal concealed patterns, making it an important tool in the detection botnets attack. ML algorithms can be trained to learn the patterns of normal and malicious network traffic from dataset collected from IoT devices. ML algorithms execute many functions such as analysis of network traffic, system logs, and other relevant data sources to identify anomalous patterns and behaviours associated with botnet activities. By utilizing supervised, unsupervised, or semi-supervised learning techniques, ML models can be trained to classify network traffic as botnet-related or legitimate, enabling real-time detection and response. Many ML algorithms have been used for botnet detection: decision trees, SVM, logistic regression, random forest. In this study ML approaches like deep learning with be evaluated against decision tree, random forest and logistic regression on IoT botnet dataset chosen from Kaggle (open data source). The intent of the study is to compare outcome of supervised and unsupervised learning in prediction of botnet attack.

* Corresponding author: Tewogbade Shakir Adeyemi

2. Literature Review

[1] affirmed the effectiveness of using ML approaches in detection of botnet in their work. They identified and analysed the importance features that readily separate normal traffic from botnet traffic. The usefulness of features cannot be less emphasized and this view was specifically mentioned in the work of [2], where they proposed extraction of useful features based on dominant pattern in the dataset. In research by [3], 23 out of 39 features in the UNSW-NB15 dataset were used in their modelling. XgBoost performed best among all classifiers with accuracy of 88% followed by random forest 87.89%. Investigation by [4] utilized different classifiers and clustering ML methods to study botnet attack. Decision tree yielded best accuracy of 90.27%. in a similar study by [5], SVM showed very effective prediction with accuracy of 98% and lowest error rate of 6% among all models used. Based on these studies, it can be concluded that machine learning is indeed suitable for the detection of botnets. The studies have shown suitability of machine learning techniques in identifying and mitigating botnet attacks. This particular study will contribute to body of knowledge by comparing outcome of supervised and unsupervised learning.

3. Case Study

3.1. Data Source and Description

The study dataset has been sourced from Kaggle: N-BaIoT Dataset to Detect IoT Botnet Attacks | Kaggle. The dataset was extracted from the work of [6] and also available on UCI Machine Learning repository.

3.2. Data Preparation and Analysis

The dataset has nine IoT devices and comprise of both benign traffic and of a variety of malicious attacks. The machine learning application in this study is to identify cyberattacks on a Provision PT-737E Security Camera.

3.3. Data Loading

Firstly, the zipfile module is imported to extract the dataset. The path to the zipped dataset file (5.mirai.udpplain.csv.zip) is set. The path to the directory where the dataset will be extracted (/content/dataset) is set. The dataset is then extracted using the extractall method from the ZipFile class.

```
import numpy as np
import pandas as pd
```

```
import zipfile

# Set the path to the zipped dataset file
zip_path = '/content/5.mirai.udpplain.csv.zip'

# Set the path to the directory where you want to extract the dataset
extract_path = '/content/dataset'

# Extract the dataset
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)
```

```
benign=pd.read_csv('/content/dataset/5.benign.csv')
g_c=pd.read_csv('/content/dataset/5.gafgyt.combo.csv')
g_j=pd.read_csv('/content/dataset/5.gafgyt.junk.csv')
g_s=pd.read_csv('/content/dataset/5.gafgyt.scan.csv')
g_t=pd.read_csv('/content/dataset/5.gafgyt.tcp.csv')
g_u=pd.read_csv('/content/dataset/5.gafgyt.udp.csv')
m_a=pd.read_csv('/content/dataset/5.mirai.ack.csv')
m_sc=pd.read_csv('/content/dataset/5.mirai.scan.csv')
m_sy=pd.read_csv('/content/dataset/5.mirai.syn.csv')
m_u=pd.read_csv('/content/dataset/5.mirai.udp.csv')
m_u_p=pd.read_csv('/content/dataset/5.mirai.udpplain.csv')

benign=benign.sample(frac=0.25,replace=False)
g_c=g_c.sample(frac=0.25,replace=False)
g_j=g_j.sample(frac=0.5,replace=False)
g_s=g_s.sample(frac=0.5,replace=False)
g_t=g_t.sample(frac=0.15,replace=False)
g_u=g_u.sample(frac=0.15,replace=False)
m_a=m_a.sample(frac=0.25,replace=False)
m_sc=m_sc.sample(frac=0.15,replace=False)
m_sy=m_sy.sample(frac=0.25,replace=False)
m_u=m_u.sample(frac=0.1,replace=False)
m_u_p=m_u_p.sample(frac=0.27,replace=False)
```

Figure 1 Codes for Importing Data

3.4. Data Pre-processing

R-code that loads the CSV files for different types of traffic data into separate DataFrames, such as benign, gafgyt, and mirai was created. Each DataFrame is randomly sampled using the sample method to reduce the size of the dataset. A new column 'type' is added to each DataFrame to represent the type of traffic. All the DataFrames are concatenated into a single DataFrame called 'data' using the concat method. The rows of the 'data' DataFrame are shuffled using a random permutation generated by np.random.permutation. The shuffled data is reassigned to the 'data' DataFrame.

3.5. Data Transformation

The labels in the 'type' column of the 'data' DataFrame are one-hot encoded using the pd.get_dummies method. The encoded labels are stored in a new DataFrame called 'labels_full'. The 'type' column is dropped from the 'data' DataFrame.

```
#dummy encode labels, store separately
labels_full=pd.get_dummies(data['type'], prefix='type')
labels_full.head()
```

	type_benign	type_gafgyt_combo	type_gafgyt_junk	type_gafgyt_scan	type_gafgyt_tcp	type_gafgyt_udp	type_mirai_ack	type_mirai_scan	type_mirai_syn	type_mirai_udp	type_mirai_udpplain
118415	0	0	0	0	0	0	1	0	0	0	0
8889	1	0	0	0	0	0	0	0	0	0	0
163561	0	0	0	0	0	0	0	0	0	0	1
47447	0	0	1	0	0	0	0	0	0	0	0
63857	0	0	0	1	0	0	0	0	0	0	0

Figure 2 Coding extract of Data Transformation

3.6. Data Standardization

The numerical columns of the 'data' DataFrame are standardized using the 'standardize' function.The 'data' DataFrame is copied to a new DataFrame called 'data_st'. The 'standardize' function subtracts the mean and divides by the standard deviation for each column.

```
#standardize numerical columns
def standardize(df,col):
    df[col]= (df[col]-df[col].mean())/df[col].std()

data_st=data.copy()
for i in (data_st.iloc[:,:-1].columns):
    standardize (data_st,i)

data_st.head()
```

	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L3_mean	MI_dir_L3_variance	MI_dir_L1_weight	MI_dir_L1_mean	MI_dir_L1_variance	MI_dir_L0.
118415	0.597048	2.206778	1.577993	0.461342	2.207620	1.526803	0.296726	1.912227	1.650875	
8889	-0.897924	0.140774	1.568879	-0.931813	0.164550	1.434118	-0.958750	0.221712	1.343135	
163561	-0.433365	2.020274	1.613521	-0.378263	2.065883	1.494750	-0.159475	1.677983	1.606929	
47447	1.265175	-0.507037	-0.603336	1.421826	-0.531828	-0.625369	1.114203	-0.560878	-0.642217	
63857	-0.009546	-0.507065	-0.603340	-0.277758	-0.531662	-0.625350	-0.661088	-0.559114	-0.641679	

Figure 3 Data Standardization

3.7. Data Exploration

The code performs a grouping on the 'type' column of the 'data' DataFrame and counts the number of instances of each class. The result is displayed, showing the count of instances for each type of traffic.

```
#how many instances of each class
data.groupby('type')['type'].count()
```

```
type
benign           15538
gafgyt_combo     15345
gafgyt_junk      15449
gafgyt_scan      14648
gafgyt_tcp       15676
gafgyt_udp       15602
mirai_ack        15138
mirai_scan       14517
mirai_syn        16436
mirai_udp        15625
mirai_udpplain   15304
Name: type, dtype: int64
```

Figure 4 Validation of Data Classes

3.8. Research Question

How can machine learning algorithms be effectively applied to detect and classify botnet activities?

This research question focuses on exploring the optimal ML approaches, such as decision trees, random forest and deep learning models.

4. Results and Discussions

The code combines data pre-processing, transformation, and training of a deep neural network using the Keras library. The training results show the progress of the model's loss during training.

4.1. Decision Tree

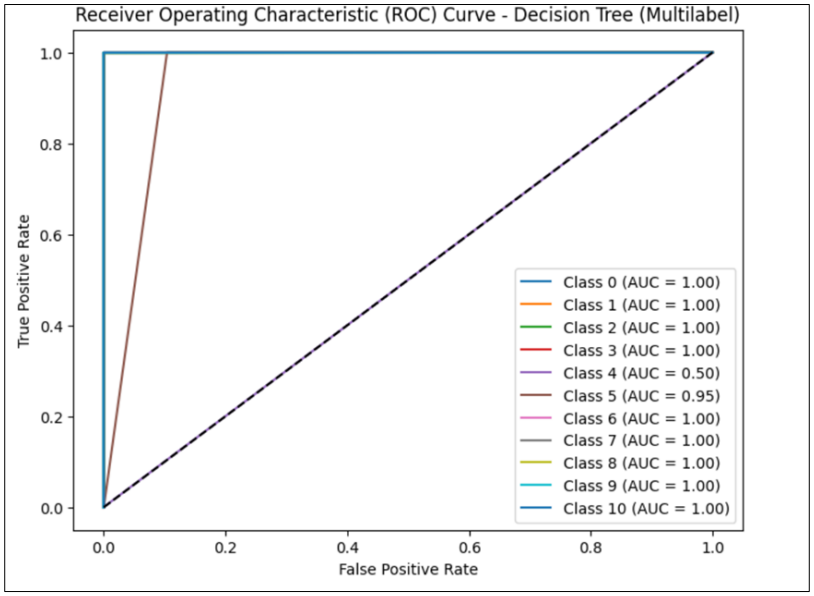

Figure 5 ROC curves for Decision Tree classifier

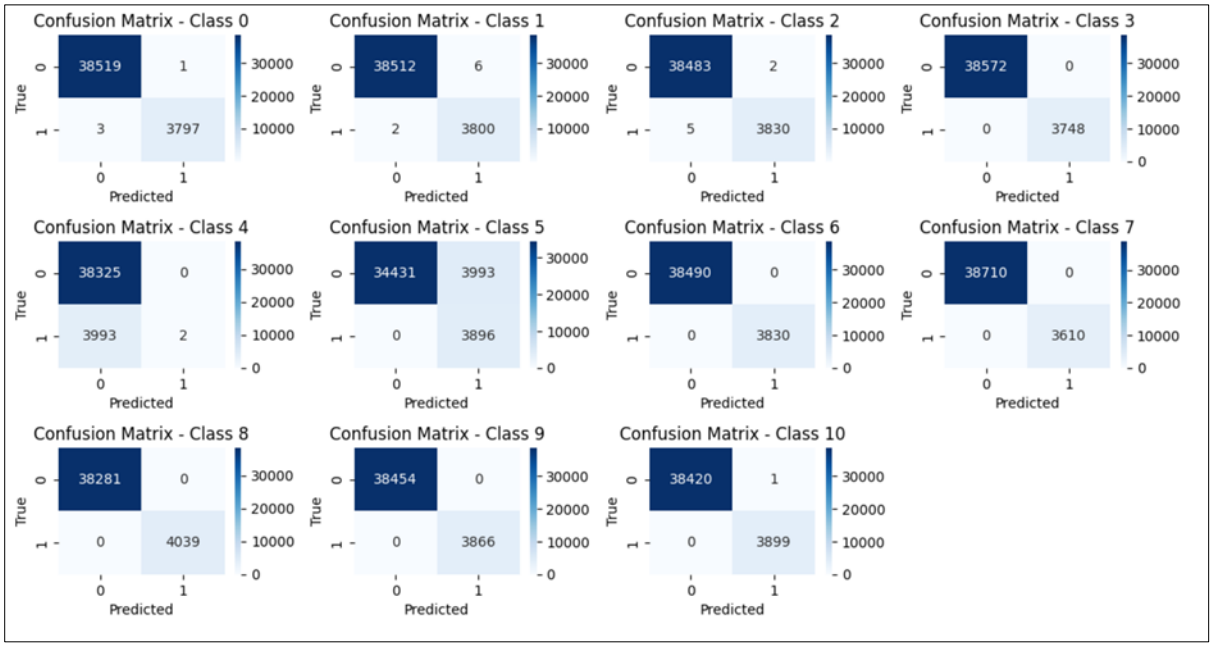

Figure 6 Confusion Matrix for Decision Tree Classifier outcome

Decision tree is one of the commonly used classifiers with repeated partitioning where the tree is formed from nodes (roots). The internal nodes (test nodes) branched out into instance space that represent attributes of the input variables.

Eventually, the tree is formed by splitting the data based on features that best separate different classes. The internal node stands for a feature, branches stand for decision rule and the leaf nodes produce output (class label).

4.2. Random Forest

Random Forest is arrived at when the tree predictors are combined in a manner that make each tree to depend on the magnitude of a random vector selected independently where all the tree in the forest exist in the same distribution. Random forest produces a set of decision trees by picking random subsets of the features and training set. The RF prediction is achieved by mixing the predictions of all individual trees.

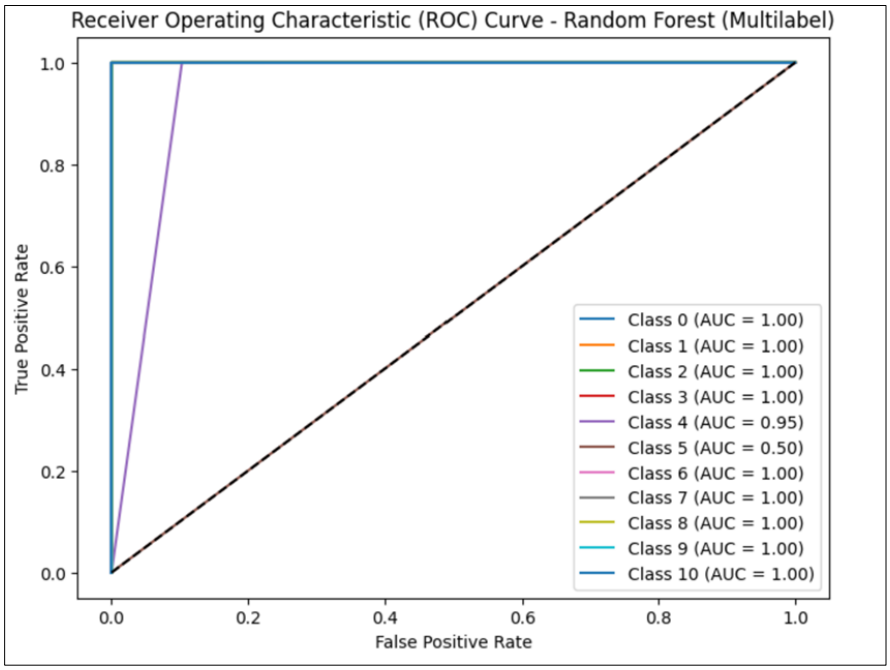

Figure 7 ROC curves for Random Forest classifier

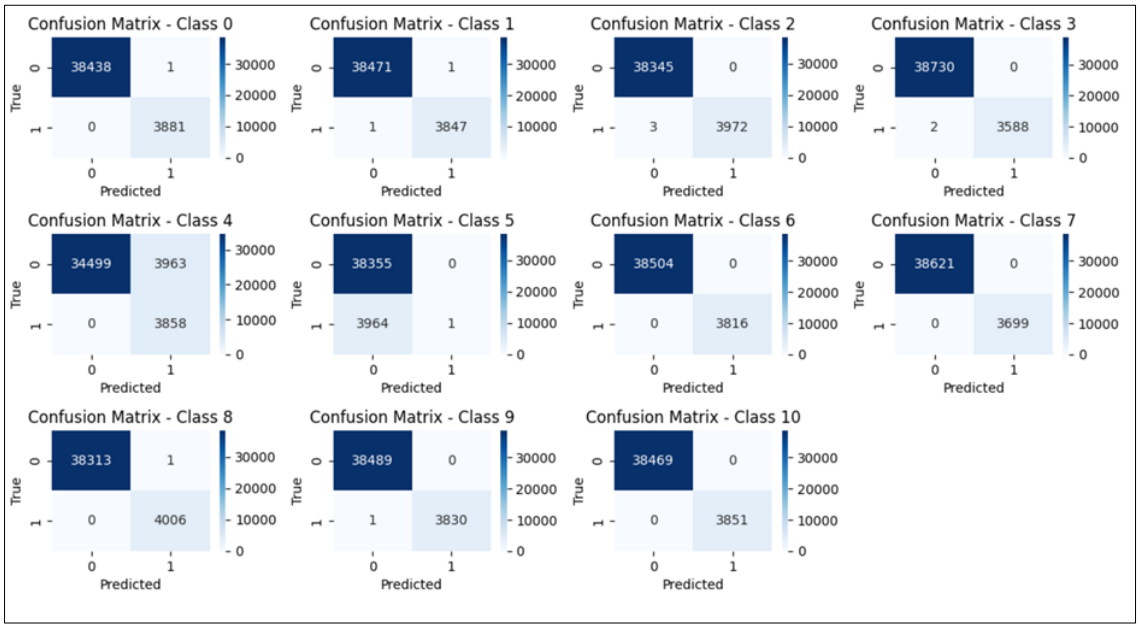

Figure 8 Confusion Matrix for Random Forest Classifier outcome

Both decision trees and random forests can be evaluated using various performance metrics. Accuracy is one such metric, which measures the proportion of correctly classified instances. The reported accuracies are 0.9054111531190926 for decision trees and 0.9054584120982987 for random forests, this indicate that both models have high accuracy.

In addition to accuracy, the ROC AUC (Receiver Operating Characteristic Area Under the Curve) curve is another commonly used metric to evaluate classification models. The ROC curve plots the true positive rate (sensitivity) against the false positive rate (1 - specificity) for different classification thresholds. An AUC score of 1.0 indicates a perfect classifier. Both decision trees and random forests have an ROC AUC accuracy of 1.00 for each multilabel class. An AUC score of 1.0 suggests that the models have excellent discrimination ability and can perfectly separate the positive and negative instances for each class. It is key to note that accuracy and AUC are just two metrics used to evaluate models, and there may be other metrics that could provide a more comprehensive evaluation. Additionally, it's essential to consider the specific dataset, its characteristics, and the problem at hand when interpreting the performance of machine learning models.

4.3. Deep Learning

Deep Learning is a subset of machine learning where models process multilayers and non-linear values. The ability to model multiple set of representation allow modelling of complex relationship that may exist in study dataset. The algorithm learns from multiple levels which have diverse status of abstraction for easier representation and improved learning. In the short term, high-level attributes or concepts are idealized at lower-level and thus the technique is termed as “deep”.

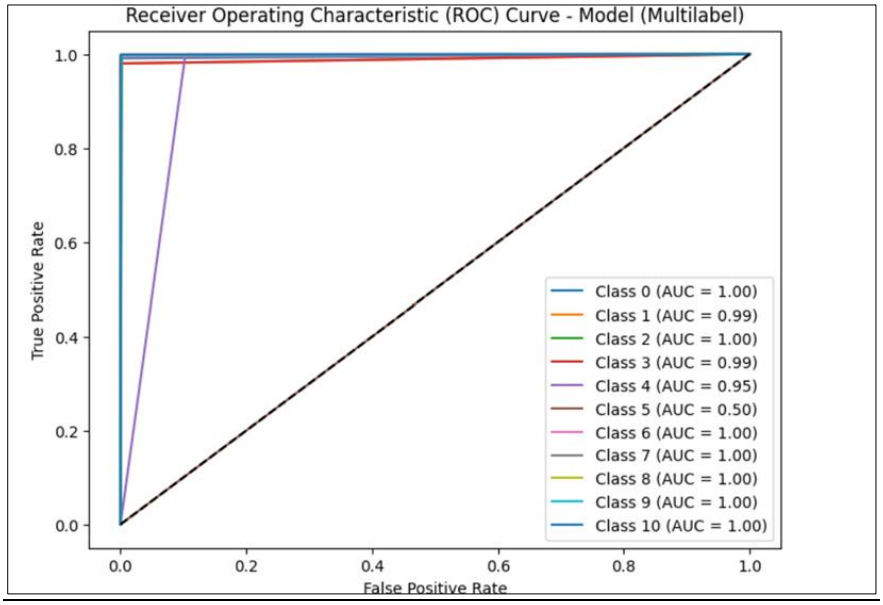

Figure 9 ROC curves for Random Forest classifier

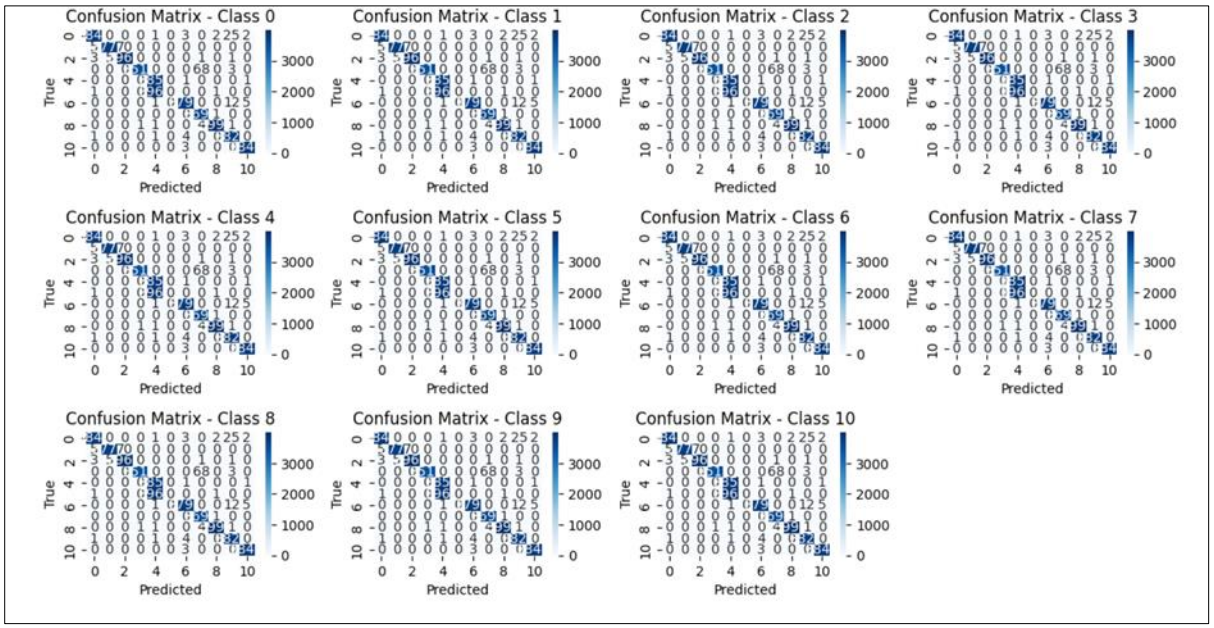

Figure 10 Confusion Matrix for Random Forest Classifier outcome

5. Conclusions

In conclusion, this study aimed to predict botnet attacks using ML approach. The results demonstrated high accuracy rates for all three models, with Random Forest achieving an accuracy of 91%, followed closely by Decision Tree at 91%, and Deep Learning achieving an accuracy of 90%.

These findings display the predictive power of machine learning algorithms in detecting botnet attacks. The high accuracy rates obtained by all three models suggest that they can be valuable tools for cybersecurity professionals in identifying and mitigating botnet threats. Random Forest and Decision Tree algorithms, which are based on ensemble learning techniques, performed particularly well in this study. Their ability to combine multiple decision trees and incorporate feature importance analysis likely contributed to their high accuracy rates. On the other hand, Deep Learning, a neural network-based approach, also showed promising results, slightly trailing behind the ensemble models. Based on the perspective of this study, the following are recommended for future work:

- Incorporate more features: Expand the feature set used in the models by including additional network-level, host-level, or behavior-based attributes. This can provide a more comprehensive understanding of botnet activities and improve the accuracy of the predictions.
- Explore hybrid models: Investigate the potential of combining multiple machine learning algorithms or ensemble methods to develop hybrid models. By leveraging the strengths of different algorithms, it may be possible to achieve even higher accuracy rates in predicting botnet attacks.
- Consider time series analysis: Explore time series analysis techniques to capture temporal patterns and dependencies in botnet activities. This can provide a more dynamic understanding of botnet behavior and enhance the accuracy of predictive models.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] X. Dong, J. Hu, and Y. Cui, Overview of Botnet Detection Based on Machine Learning, in International Conference on Mechanical, Control and Computer Engineering, Huhhot, pp. 476-479, 2018.

[2] M. Lefoane, I. Ghafir, S. Kabir, and I.U. Awan, Machine Learning for Botnet Detection: An Optimized Feature Selection Approach, in The 5th International Conference on Future Networks & Distributed Systems (ICFNDS 2021), December 15–16, 2021, Dubai, United Arab Emirates, ACM, New York, pp. 6.

[3] A. Husain, A. Salem, C. Jim, and G. Dimitoglou, Development of an efficient network intrusion detection model using extreme gradient boosting (XGBoost) on the UNSW-NB15 dataset, in Proceedings of the 2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Ajman, UAE, December 2019, pp. 1–7.

[4] S. Haq and Y. Singh, Botnet Detection using Machine Learning, in International Conference on Parallel, Distributed and Grid Computing (PDGC), India, 2018, pp. 240-245.

[5] S. Saad et al., Detecting P2P botnets through network behavior analysis and machine learning, in International Conference on Privacy, Security and Trust, Montreal, QC, 2011, pp. 174-180.

[6] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, N-baiot—network-based detection of IoT botnet attacks using deep autoencoders, IEEE Pervasive Computing, vol. 17, no. 3, pp. 12-22, 2018.