Botnet Threat Detection using Machine Learning and Neural networks

Sandeep Rajakrishnan, Sudhay Senthilkumar,
Senthilkumar Thangavel[*1] and Sulakshan Vajipayajula[*2]
Department of Computer Science and Engineering
[*1]Amrita School of Engineering, Coimbatore,
Amrita Vishwa Vidyapeetham, India.
[*2]Architect-CTO Office
IBM Security, Bangalore

sandur43@gmail.com ,
sudhay2001@gmail.com ,
t_senthilkumar@cb.amrita.edu [*1],
svajipay@in.ibm.com [*2]

Abstract:

With an ever-increasing population and the complexity of interconnected devices that can communicate information among themselves both with and without user intervention, computers are being used to automate simple and advanced human jobs. This has also resulted in the transfer of critical information over the network to achieve a common goal where safeguarding such information is critical, such as in large organizations and individual setups. Because of the widespread use of IoT systems, malevolent third parties have begun to target them. To handle this challenge, intrusions or attacks must be detected and prevented. As an effort towards this, we have used the IoT Botnet dataset generated by the University of New South Wales, Canberra, Australia. We used numerous machine learning algorithms and neural networks to classify the data points into attack categories after using multiple preprocessing approaches. Finally, we fine-tuned the hyperparameters to get optimal outcomes, compared the trained models' accuracies, and arrived at the optimal model for production deployment.

Keywords: Botnet, IoT, Machine Learning, Preprocessing, Malware, Attack

Introduction

While cyber-attacks have always been on the rise, Covid-19 has accelerated it. With work from home becoming the norm world over, the heavy dependence on technology was inevitable. In addition to that the increased adoption of 5G, interconnectedness of devices, new processes and procedures, updated employee profiles, and less-controlled work environments - have all led to an increase in vulnerabilities. According to research from Nozomi Networks, assaults and threats directed at Operational Technology (OT) and Internet of Things (IoT) networks increased in the first half of 2020, particularly from IoT botnets. With so many botnets on the internet nowadays, prevention is critical—but it's not easy. Botnets are constantly changing to exploit vulnerabilities and security shortcomings. As a result, each botnet might be vastly different from the others. With this in mind, our work aims to identify such threats on networks with high efficiency, which will aid in the development of improved countermeasures.

Background

Botnet attacks are classified into 4 categories :

Distributed denial of service (DDoS)

A distributed denial-of-service (DDoS) attack occurs when a large number of compromised systems are exploited to bombard a single target, leading to a loss of service for the targeted system's users.

The torrent of incoming traffic ultimately compels the target system to shut down, denying access to authorized users. The hacker exploits a vulnerability in one computer system and makes it the DDoS master in a typical DDoS attack. The attack master (also known as the botmaster) hunts down and infects other machines with malware. The hacker steers the controlled devices to attack a certain target. Large corporations are the most common targets of DDoS attacks, but enterprises can still be targeted or become victims if another organization on their network suffers.

Denial of service (DoS)

A Denial of Service (DoS) attack is a strategy for barring legitimate users from accessing a network or internet resource. This is commonly accomplished by submitting a malicious request that causes the target resource to fail or entirely crash, or by overloading a target (often a web server) with a large volume of traffic.

Some DoS attacks are intended to disrupt a specific target's access to a network of resources, whereas others are destined to render a resource completely inaccessible. These attacks can last anywhere from a few minutes to several hours, and in rare circumstances, even days.

Reconnaissance

Reconnaissance attacks are designed to gather information about a network. An intruder selects a target, researches it, and searches for vulnerabilities or information such as operating system, services on the network that are currently operational, permissions for files, active targets, relationships of trust, information about the user's accounts.

It's also important for penetration testing. A thorough reconnaissance would provide detailed information and allow attackers to scan and attack the entire route.

Theft

Money theft, data theft, and company disruption are all manifestations of criminally motivated attackers. Similarly, those who are motivated by personal gains, such as disgruntled current or former workers, will accept funds, data, or the potential to disrupt a company's system. They are mostly looking for vengeance. Attackers with socio-political motivations want to draw attention to themselves and their causes.

Dataset Used:

For initial investigation of the nature of malware datasets, we have explored three datasets namely UNSW_NB15, IoT Botnet, and NSL KDD looking for features that contribute towards the identification of malware in network infrastructures. Further analysis of the features and their generation in a realistic environment helped us narrow down to using the "UNSW_2018_IoT_Botnet" dataset developed at the Research Cyber Range lab of UNSW Canberra.

Dataset Description:

Upon observation of parameters from a range of malware datasets which include UNSW_NB15, IoT Botnet and NSL KDD, we were able to find out the influencing features and common features across these datasets. Our observation also includes a multiclass classified output into various types of attacks including DDoS, DoS, Theft, Reconnaissance. As a result of these common features, we understand that when a new data point is provided and asked to classify under a type of attack, we can find its type of attack using the independent features in the chosen dataset, which will greatly assist the organization or individual in mitigating the identified type of attack. The context about each feature in the obtained dataset generated by UNSW is also mentioned below.

Table 3
Features and descriptions.

Feature	Description
pkSeqID	Row Identifier
Stime	Record start time
flgs	Flow state flags seen in transactions
flgs_number	Numerical representation of feature flags
Proto	Textual representation of transaction protocols present in network flow
proto_number	Numerical representation of feature proto
saddr	Source IP address
sport	Source port number
daddr	Destination IP address
dport	Destination port number
pkts	Total count of packets in transaction
bytes	Totan number of bytes in transaction
state	Transaction state
state_number	Numerical representation of feature state
ltime	Record last time
seq	Argus sequence number
dur	Record total duration
mean	Average duration of aggregated records
stddev	Standard deviation of aggregated records
sum	Total duration of aggregated records
min	Minimum duration of aggregated records
max	Maximum duration of aggregated records
spkts	Source-to-destination packet count
dpkts	Destination-to-source packet count
sbytes	Source-to-destination byte count
dbytes	Destination-to-source byte count
rate	Total packets per second in transaction
srate	Source-to-destination packets per second
drate	Destination-to-source packets per second
attack	Class label: 0 for Normal traffic, 1 for Attack Traffic
category	Traffic category
subcategory	Traffic subcategory

UNSW provides a fragmented subset of the broader dataset due to the size of the recorded dataset, which has 72 million data points (16.7 GB). Due to computational restrictions, we chose a suitable dataset with 733,705 data points that are further preprocessed before training the models.

Data preprocessing and analysis:

Sample of independent features before pre-processing

pkSeqID	proto	saddr	sport	daddr	dport	seq	stddev
792371	udp	192.168.100.150	48516	192.168.100.3	80	175094	0.226784
2056418	tcp	192.168.100.148	22267	192.168.100.3	80	143024	0.451998
2795650	udp	192.168.100.149	28629	192.168.100.3	80	167033	1.931553

N_IN_Conn_P_SrcIP	min	state_number	mean	N_IN_Conn_P_DstIP	drate
100	4.100436	4	4.457383	100	0.000000
100	3.439257	1	3.806172	100	0.225077
73	0.000000	4	2.731204	100	0.000000

drate	srate	max	attack	subcategory
0.000000	0.404711	4.719438	1	UDP
0.225077	0.401397	4.442930	1	TCP
0.000000	0.407287	4.138455	1	UDP

Sample of Dependent/target feature before pre-processing

category
DoS
DDoS
DDoS

After observing the dataset obtained, looking for the nature of values present in all attributes, we conclude that we have a mix of string data in the form of IP addresses and numerical data for other parameters such as port numbers. Also, it is to be noted that we have 5 target classes. We divided the dataset into groups of numerical and categorical features. To deal with string and categorical data we considered multiple approaches to perform encoding. The encoding techniques included Label Encoding, One-Hot Encoding, Mean Encoding, and Target Encoding. Of the four,

After reviewing the dataset acquired and searching for the nature of values contained in all attributes, we conclude that we have a combination of string data in the form of IP addresses and numerical data for other characteristics such as port numbers. It's also worth noting that there are five target classes in all. The data was split into numerical and categorical features. We investigated several encoding options to cope with string and category data. Label encoding, One-Hot encoding, Mean encoding, Category encoding, and Target encoding were all used as encoding techniques. Target Encoding and category encoding proved to yield the optimal results for the test and train datasets.

For the reasons stated, we did not use the following encoding methods:

- The problem of dimensionality stems from one hot encoding.
- Data is not ordinal, thus label encoding is not used.
- Mean encoding cannot be used as the dependent variable is not binary and the same mean appears for more than 1 unique value, it is a multiclass classification problem

Value Count for Target Variable Categories

```
DDoS              385309
DoS               330112
Reconnaissance     18163
Normal               107
Theft                 14
```

Statistical Description of Numerical Features

	seq	stddev	N_IN_Conn_P_SrcIP	min	state_number	mean	N_IN_Conn_P_DstIP	drate	srate	max
count	733705.000000	733705.000000	733705.000000	733705.000000	733705.000000	733705.000000	733705.000000	733705.000000	733705.000000	733705.000000
mean	121412.819892	0.887894	82.492551	1.018868	3.135073	2.233429	92.427763	0.506298	2.262398	3.023000
std	75823.398840	0.804013	24.426145	1.484235	1.186427	1.517572	18.216076	74.330175	403.408092	1.860725
min	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	54993.000000	0.030132	69.000000	0.000000	3.000000	0.182193	100.000000	0.000000	0.156231	0.281688
50%	117896.000000	0.795481	100.000000	0.000000	4.000000	2.691715	100.000000	0.000000	0.283784	4.011386
75%	185157.000000	1.745595	100.000000	2.163444	4.000000	3.566569	100.000000	0.000000	0.488849	4.296505
max	262212.000000	2.496758	100.000000	4.980470	11.000000	4.981785	100.000000	58823.527344	333333.312500	4.999999

Normalization

Normalization/standardization always has the same primary objective. Variables reported at different scales may not contribute equally to the model fitting and learning function, which might lead to bias. As a result, feature-wise normalization is commonly utilized before model fitting to address this possible issue.

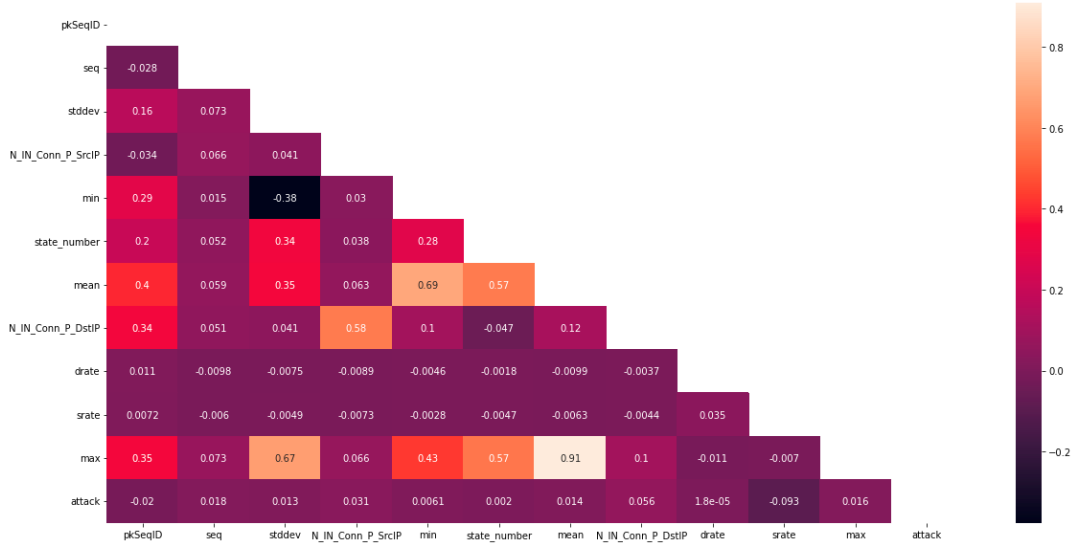
Standard scalar removes the mean and scaled to unit variance to standardize features. A sample x's standard score is calculated as follows:

$$z = (x - u) / s$$

where u is the training samples' mean, or zero if with mean=False, and s is the training samples' standard deviation, or one if with std=False.

Correlation Heatmap and Coolwarm map

The correlation between the variables on each axis is shown in each square. Correlation might be anywhere between -1 and +1. Closer to 0 indicates that there is no linear relationship between the two variables. The closer the correlation is to one, the more positively associated they are; that is, when one rises, so does the other, and the closer to one they are, the stronger the association. When the correlation is near -1, instead of both variables increasing, one will drop as the other increases. The stronger the connection between the two variables, the larger the number and the darker the color.



	pkSeqID	seq	stddev	N_IN_Conn_P_SrcIP	min	state_number	mean	N_IN_Conn_P_DstIP	drate	srate	max	attack
pkSeqID	1.000000	-0.027968	0.163074	-0.034002	0.285580	0.203275	0.398975	0.344957	0.011276	0.007222	0.350195	-0.019877
seq	-0.027968	1.000000	0.073480	0.065563	0.015416	0.052405	0.058639	0.050539	-0.009782	-0.006045	0.072952	0.017947
stddev	0.163074	0.073480	1.000000	0.040908	-0.376608	0.339678	0.350176	0.040694	-0.007464	-0.004896	0.667964	0.012640
N_IN_Conn_P_SrcIP	-0.034002	0.065563	0.040908	1.000000	0.030092	0.037561	0.062560	0.576600	-0.008948	-0.007255	0.065883	0.030598
min	0.285580	0.015416	-0.376608	0.030092	1.000000	0.276375	0.693706	0.103622	-0.004557	-0.002796	0.426768	0.006102
state_number	0.203275	0.052405	0.339678	0.037561	0.276375	1.000000	0.572631	-0.047446	-0.001837	-0.004750	0.566267	0.002041
mean	0.398975	0.058639	0.350176	0.062560	0.693706	0.572631	1.000000	0.120018	-0.009868	-0.006285	0.908540	0.013647
N_IN_Conn_P_DstIP	0.344957	0.050539	0.040694	0.576600	0.103622	-0.047446	0.120018	1.000000	-0.003651	-0.004362	0.101255	0.055616
drate	0.011276	-0.009782	-0.007464	-0.008948	-0.004557	-0.001837	-0.009868	-0.003651	1.000000	0.035355	-0.010914	0.000018
srate	0.007222	-0.006045	-0.004896	-0.007255	-0.002796	-0.004750	-0.006285	-0.004362	0.035355	1.000000	-0.007026	-0.093351
max	0.350195	0.072952	0.667964	0.065883	0.426768	0.566267	0.908540	0.101255	-0.010914	-0.007026	1.000000	0.015957
attack	-0.019877	0.017947	0.012640	0.030598	0.006102	0.002041	0.013647	0.055616	0.000018	-0.093351	0.015957	1.000000

From the above correlation heatmap and the correlation cool warm map below we see that mean and max columns are highly correlated, hence we can remove one of them

Train test split

We separated the dataset into train and test after preprocessing to apply several machine learning algorithms. The split was made in an 80:20 ratio, with 80% of the dataset being utilized for training and the remaining 20% being used for testing. The random state was also configured so that if the train test split was repeated, the same results would be obtained. If the random state is not set, we will get a different set of train and test datasets each time we split, which is inconvenient for later troubleshooting.

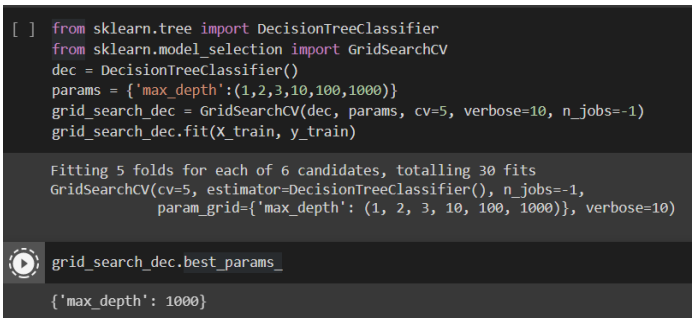
Implementation and Results:

GridSearchCV

GridSearchCV is a module present within the scikit-learn package that assists us in arriving at the most appropriate hyperparameters for a chosen machine learning algorithm by applying all possible combinations and filtering out the best one. The prerequisites are the instance of the machine learning algorithm to be fit, the dictionary of all parameter values to be tried out, and a cross-validation value.

Using GridSearchCV we were able to choose:

- *{max_depth:1000}* for DecisionTreeClassifier



- *{max_depth=10, n_estimators=10}* for RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
randForest = RandomForestClassifier()
params = {'max_depth':(1,2,9,10),'n_estimators':(10,15,30)}
grid_search = GridSearchCV(randForest, params, cv=5, verbose=10)
grid_search.fit(X_train, y_train)
```

```
y_pred = grid_search.predict(X_test)
```

```
grid_search.best_params_
```

```
{'max_depth': 9, 'n_estimators': 10}
```

Multinomial Logistic Regression

To model nominal result variables, multinomial logistic regression is utilized, in which the log chances of the outcomes are modeled as a linear mixture of the predictor variables.

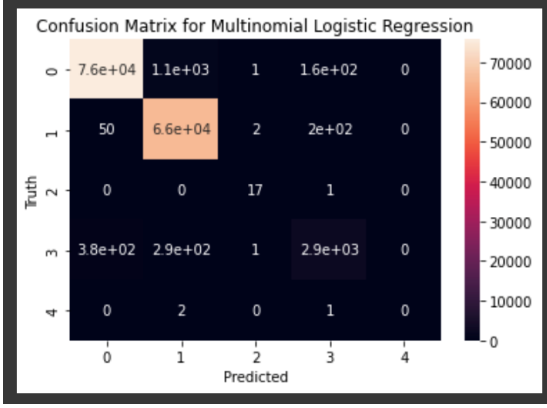
It's a natural extension of the binomial model with a logit link function for scenarios in which the target variable has three or more possible possibilities.

Following are the parameters for the LogisticRegression function:

*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='multinomial', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*

Classification report for Multinomial Logistic Regression

	precision	recall	f1-score	support
DDoS	0.994369	0.984061	0.989188	76983
DoS	0.979881	0.996145	0.987946	66152
Normal	0.809524	0.944444	0.871795	18
Reconnaissance	0.888584	0.814226	0.849782	3585
Theft	0.000000	0.000000	0.000000	3
accuracy			0.985335	146741
macro avg	0.734472	0.747775	0.739742	146741
weighted avg	0.985210	0.985335	0.985188	146741



We have obtained a test accuracy of 0.985335

Decision Tree Classifier

A Decision Tree is a supervised machine learning algorithm. This approach may be used to solve both regression and classification problems, however, it is most commonly employed to solve classification problems. A decision tree visualizes data and sorts it into categories based on a series of if-else rules.

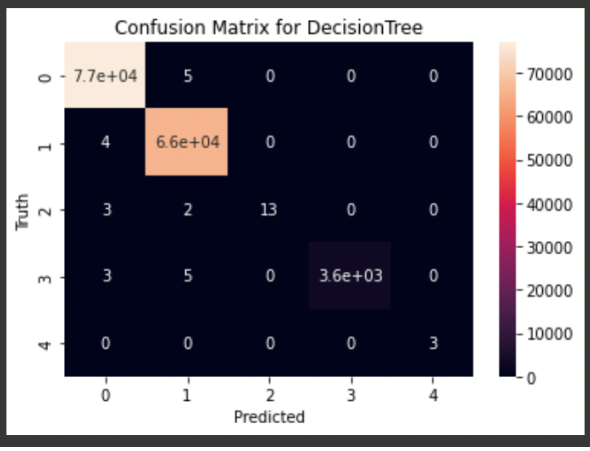
The goal of employing a Decision Tree is to develop a training model that can be used to predict the class or value of target variables by learning decision rules based on past data trained.

Using GridSearchCV, we have set the following parameter to get the optimal results:

'max_depth': 1000

Classification report for DecisionTree

	precision	recall	f1-score	support
DDoS	0.999870	0.999935	0.999903	76983
DoS	0.999819	0.999940	0.999879	66152
Normal	1.000000	0.722222	0.838710	18
Reconnaissance	1.000000	0.997768	0.998883	3585
Theft	1.000000	1.000000	1.000000	3
accuracy			0.999850	146741
macro avg	0.999938	0.943973	0.967475	146741
weighted avg	0.999850	0.999850	0.999847	146741



We have obtained a test accuracy of 0.999850.

RandomForestClassifier

Random forest is an ensemble model with numerous decision trees that can produce more accurate results because there isn't just one tree contributing to the final outcome. It employs strategies such as "Majority Voting Strategy". This implies that each "estimator" in the ensemble comes to its conclusion, and the final aggregate is determined by the target class with the most votes. Another strategy is "soft voting," in which each estimator delivers a probability for each class instead of a final result, and the final aggregate is determined based on

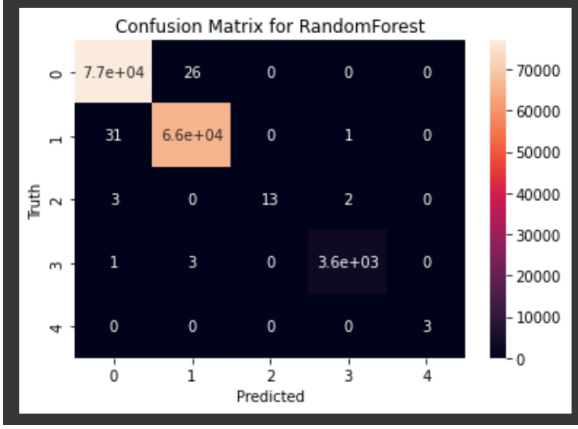
the class with the greatest sum of weighted probabilities.

Using GridSearchCV, we have set the following parameters to get the optimal results:

'max_depth': 10, 'n_estimators': 10

Classification report for RandomForest

	precision	recall	f1-score	support
DDoS	0.999545	0.999662	0.999604	76983
DoS	0.999562	0.999516	0.999539	66152
Normal	1.000000	0.722222	0.838710	18
Reconnaissance	0.999163	0.998884	0.999024	3585
Theft	1.000000	1.000000	1.000000	3
accuracy			0.999543	146741
macro avg	0.999654	0.944057	0.967375	146741
weighted avg	0.999543	0.999543	0.999541	146741



For RandomForest, we have obtained an optimal accuracy of 0.999543

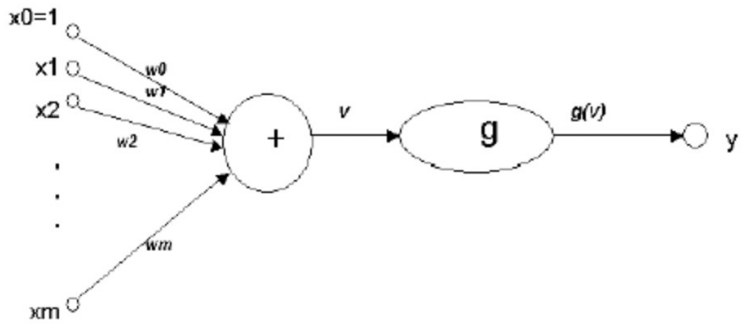
Artificial Neural Networks

Artificial neural networks (ANNs) are rudimentary networks of neurons based on the brain's neural structure. This process records one at a time, learning by comparing their prediction of the record with the known true classification of the record. The errors from the first record's initial categorization are fed back into the network and utilized to tweak the network's algorithm for subsequent rounds by adjusting the weights.

In an artificial neural network, a neuron is:

1. A set of input values (xi) and weights (wi).

2. A function (g) that adds up the weights and maps them to a result (y).



Keras Sequential model

In Tensorflow's Keras, there's a KerasClassifier class that may be used as an Estimator in scikit-learn, which is the library's base model type. A function's name is passed as an argument to the KerasClassifier. The generated neural network model, ready for training, must be returned by this function.

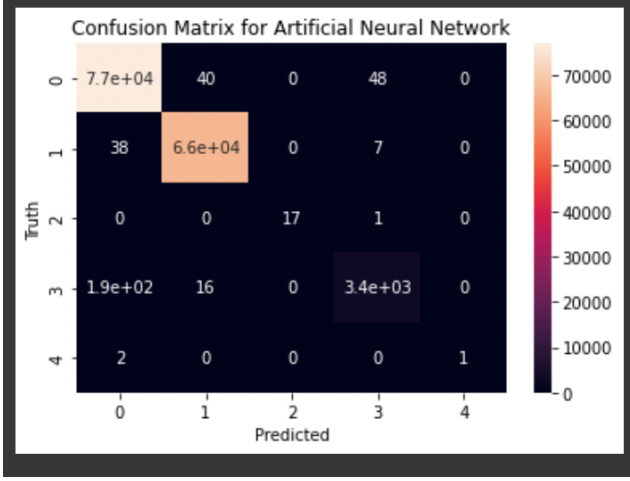
In our scenario, there are 16 neurons in the input layer, 12 neurons in the hidden layer, and 5 neurons in the final output layer, which represent the five target categories.

For the input and hidden layer, we have used the "relu" activation function. In the output layer, we utilize a "softmax" activation function. This ensures that the output values are between 0 and 1 and that they can be used as expected probabilities.

Finally, the network employs Keras' "categorical cross-entropy" method, which utilizes the Adam gradient descent optimization approach with a logarithmic loss function.

Classification report for Artificial Neural Network

	precision	recall	f1-score	support
0	0.997070	0.998857	0.997962	76983
1	0.999154	0.999320	0.999237	66152
2	1.000000	0.944444	0.971429	18
3	0.983716	0.943654	0.963269	3585
4	1.000000	0.333333	0.500000	3
accuracy			0.997697	146741
macro avg	0.995988	0.843922	0.886379	146741
weighted avg	0.997683	0.997697	0.997676	146741



We have obtained an accuracy of 0.997697

Inference:

Algorithm	Accuracy
Artificial Neural Networks	0.997697
RandomForestClassifier	0.999543
Multinomial Logistic Regression	0.985335
Decision Tree Classifier	0.999850

From the observations, we can conclude that, given the current train test split and the hyperparameters chosen by GridSearchCV, the Decision Tree Classifier is the best performing model.

Conclusion:

In the undertaken research activity, we have used various machine learning algorithms namely Artificial Neural Networks, RandomForest Classifier, Decision Tree Classifier, Multinomial Logistic Regression, further we improved the results by fine-tuning the hyperparameters with GridSearchCV. We found that the Decision tree classifier was the best performing model for the considered test train split, since the findings are so close, we can also deduce that one of the other competent models may do better on the entire dataset or a larger subset, albeit by a small margin. Overall results will hover around the higher side of accuracy, allowing us to implement mitigating steps against the identified threats with confidence. This helps us to build a robust machine learning model around the IoT Botnet dataset. In the future, our results can be used as a benchmark while developing or optimizing machine learning algorithms for malware/botnet detection in IoT devices.

References:

- Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset
- https://cloudstor.aarnet.edu.au/plus/s/umT99TnxvbpkkoE?path=%2FCSV%2FTraning%20and%20Testing%20Tets%20(5%25%20of%20the%20entier%20dataset)%2FAll%20features - Dataset Source
- https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657
- DDoS: https://www.njamha.org/it/resources/XAHIVEFactsheetCyberAttacksDDOS.pdf

- IXIA PerfectStorm tool - use in Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) - towards the generation of "UNSW_NB15_dataset.csv"
- https://www.oreilly.com/library/view/machine-learning-for/9781783980284/47c32d8b-7b01-4696-8043-3f8472e3a447.xhtml
- ANN: https://www.solver.com/xlminer/help/neural-networks-classification-intro
- Github link: https://github.com/IBM-ML-PROJECT/Group-10_BotNet_Malware_Analysis