University of Wisconsin Milwaukee

UWM Digital Commons

Theses and Dissertations

December 2020

Cross Dataset Evaluation for IoT Network Intrusion Detection

Anjum Farah
University of Wisconsin-Milwaukee

CROSS - DATASET EVALUATION FOR IOT NETWORK INTRUSION DETECTION

by

Anjum Farah

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Computer Science

at

The University of Wisconsin-Milwaukee

December 2020

ABSTRACT
CROSS -DATASET EVALUATION FOR IOT NETWORK INTRUSION DETECTION

by

Anjum Farah

The University of Wisconsin -Milwaukee, 2020
Under the Supervision of Professor Rohit J. Kate

With the advent of Internet of Things (IOT) technology, the need to ensure the security of an IOT network has become important. There are several intrusion detection systems (IDS) that are available for analyzing and predicting network anomalies and threats. However, it is challenging to evaluate them to realistically estimate their performance when deployed. A lot of research has been conducted where the training and testing is done using the same simulated dataset. However, realistically, a network on which an intrusion detection model is deployed will be very different from the network on which it was trained. The aim of this research is to perform a cross-dataset evaluation using different machine learning models for IDS. This helps ensure that a model that performs well when evaluated on one dataset will also perform well when deployed. Two publicly available simulation datasets., IOTID20 and Bot-IoT datasets created to capture IOT networks for different attacks such as DoS and Scanning were used for training and testing. Machine learning models applied to these datasets were evaluated within each dataset followed by cross -dataset evaluation. A significant difference was observed between the results obtained using the two datasets. Supervised machine learning models were built and evaluated for binary classification to classify between normal and anomaly attack instances as well as for multiclass classification to also categorize the type of attack on the IoT network.

TABLE OF CONTENTS

LIST OF FIGURES VI
LIST OF TABLES VII
LIST OF ABBREVIATIONS VIII
ACKNOWLEDGEMENTS IX
1. INTRODUCTION 1
1.1. BACKGROUND 1
1.2. MOTIVATIONS AND OBJECTIVES 3
2. LITERATURE SURVEY 5
3. MATERIALS AND METHODS 16
3.1. TERMINOLOGY 16
3.1.1. Different types of attacks 16
3.1.2. Classification 17
3.2. CICFLOWMETER 18
3.3. DATASETS 18
3.3.1. Bot-IoT Dataset 19
3.3.2. IOTID20 Dataset 20
3.4. MACHINE LEARNING METHODS 23
3.4.1. Naïve Bayes 23
3.4.2. Logistic Regression 23
3.4.3. KNN 24
3.4.4. Decision Trees 24
3.4.5. Ensemble 24
3.5. EVALUATION METHODS 25
3.6. METHODOLOGY 27
3.6.1. Data Preparation 27
3.6.1.1 Extracting data 27
3.6.1.2. Labelling 28
3.6.1.3. Data Cleaning 28
3.6.2. Features 30
3.6.2.1 Feature Removal 32
3.6.2.2. Feature Ranking and Distribution 33
3.6.2.3. Scaling 36
3.6.3. Testbed and Experimental Setup 36
4. RESULTS AND DISCUSSIONS 38
4.1. COMPARISON OF RESULTS WITHIN SAME DATASETS 38

4.1.1. Binary Classification 38
4.1.2. Category Classification 40
4.2 CROSS DATASET EVALUATION RESULTS 42
4.2.1. Binary Classification 43
4.2.2. Category Classification 44
4.3. DISCUSSION 46
4.4. CHALLENGES 48
5. CONCLUSION AND FUTURE WORK 50
REFERENCES 52

LIST OF FIGURES

Figure 1: Types of Intrusion Detection Systems 2
Figure 2: Bot-IoT Testbed Architecture [25] [26] 20
Figure 3 : IoTID20 testbed environment [31] [28] 21
Figure 4 : Flow Duration distribution for Bot-IoT dataset 34
Figure 5 : Flow Duration distribution for IoTID20 dataset 34
Figure 6 : Packet Size Average distribution for Bot-IoT dataset 35
Figure 7 : Packet Size Average for IoTID20 dataset 35
Figure 8: Weighted AUC scores for Bot-IoT Binary Classification 39
Figure 9: Weighted AUC scores for IoTID20 Binary Classification 40
Figure 10 : AUC value for each category for Bot-IoT 41
Figure 11: AUC for each category class for IoTID20 42
Figure 12 : Weighted AUC for Cross-dataset Binary Classification with IoTID20 as training set and Bot-IoT as test set 43
Figure 13 :Weighted AUC for Cross-dataset Binary Classification with Bot-IoT as training set and IoTID20 as test set 44
Figure 14: AUC for Cross-dataset Category Classification with training set as IoTID20 and test set as Bot-IoT dataset 45
Figure 15: AUC for Cross-dataset Category Classification with training set as Bot-IoT and test set as IoTID20 dataset 46

LIST OF TABLES

Table 1: Literature Review Summary 14
Table 2 : IoTID20 Binary Data Distribution 22
Table 3 : IoTID20 Anomaly Data Distribution 22
Table 4 : Bot-IoT Binary Data 29
Table 5 : Bot-IoT Anomaly Data 29
Table 6 : IoTID20 Binary Data... 29
Table 7: IoTID20 Anomaly Data... 30
Table 8 : Feature Description 32
Table 9 : Deleted Features 33
Table 10 : Important features using Information Gain 33

LIST OF ABBREVIATIONS

AUC	Area under the ROC curve
DT	Decision Trees
DoS	Denial of Service
DDoS	Distributed Denial of Service
IDS	Intrusion Detection System
IoT	Internet of Things
KDD	Knowledge Data Discovery
ML	Machine Learning
NB	Naïve Bayes
RF	Random Forest
ROC	Receiver Operator Characteristics
RT	Random Tree
SVM	Support Vector Machines

ACKNOWLEDGEMENTS

First and foremost, I would like to sincerely thank my advisor Dr. Rohit J. Kate for mentoring me throughout this research. I deeply appreciate his knowledge on the research topics and his approach in offering helpful solutions. I am truly grateful for his constant guidance and patience in acknowledging and encouraging my research interests.

Additionally, I would also like to thank my thesis committee members., Dr. Zeyun Yu and Dr. Jun Zhang for their invaluable suggestions and guidance.

I wish to thank the Computer Science department at University of Wisconsin-Milwaukee for providing me the resources especially through the COVID-19 pandemic to help learn, grow, and pursue my academic aspirations.

Finally, I would like to thank my parents Lubna and Sadiq, my husband Akram and my siblings, Sabah and Is-haaq for always standing by me and encouraging me to reach my goals.

1. INTRODUCTION

1.1. Background

The Internet of Things or commonly referred to as IoT constitutes to a network of several million devices that are interconnected to each other via the internet for the purpose of accumulating, sharing, and processing data. In other words, any of the smart devices connected over wireless networks can be controlled and used for communication. The application of IoT is widely spread into several domains ranging from households, commercial, automobiles to industries. IoT technology is highly scalable and has a huge economic impact on the society. According to research conducted by McKinsey Global institute [1] in 2015, the IoT Market is estimated to be valued at $11.1 trillion by 2025 across nine different sectors. Having several devices connected over the internet is a great cause of concern for security. Not all data that is transferred over the IoT network is encrypted and can be vulnerable to different attacks or threats. A lot of research is being performed to enhance IoT security to create a reliable and secure network. Solutions provided should adhere to the three basic principles of security that is confidentiality, integrity, and authenticity. Developing various types of security measures such as intrusion detection systems for IoT networks have become quite significant and the idea of applying machine learning to IDS is gaining immense popularity.

Intrusion detection systems refers to a software or a service that can help monitor or identify anomalous activity within a network or a system. Several machine learning methods are being implemented to predict anomaly detection within IOT networks and promised successful results [2]. Intrusion detection systems as seen in Figure 1., are categorized mainly into two types – Host-based Intrusion detection systems and Network-based intrusion detection systems. Host based IDS is used to monitor and secure a single device or host, based on information of the device such as system logs. Network based IDS is used to monitor an entire network by accessing and analyzing the flows present within the network.

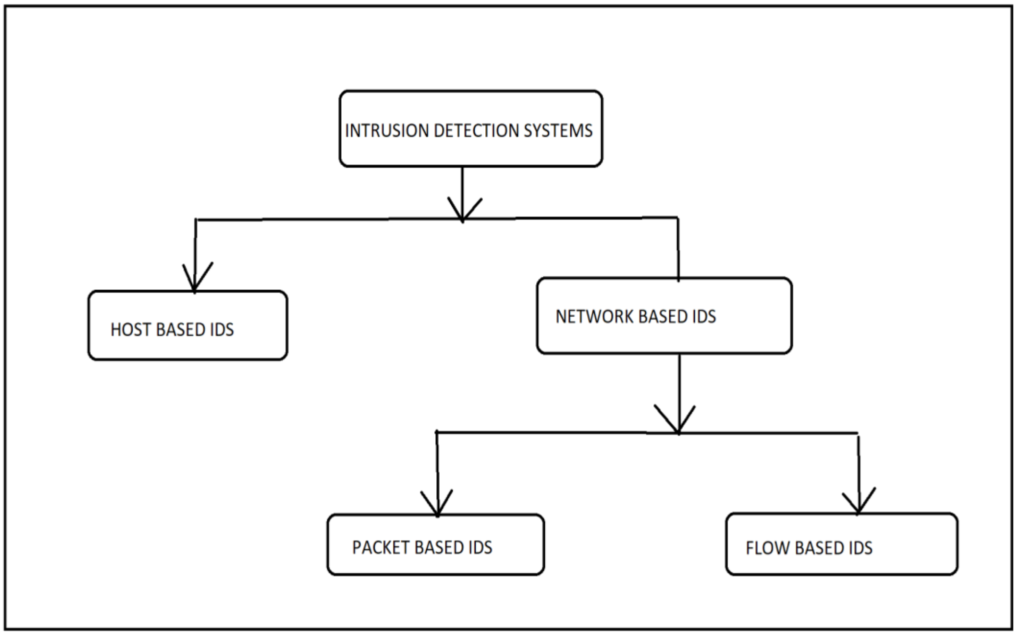

Figure 1: Types of Intrusion Detection Systems

Network based IDS are further classified into Packet based IDS and Flow based IDS. Packet based IDS used the network packet information such as payload or header information. They are also referred to as Traditional IDS [3]. Flow based IDS on the other hand uses network flow characteristics such as data rate and byte information to analyze and monitor anomalies within the network. Flow based IDS are also called as Network Behavior Analysis [3]. Several supervised or unsupervised machine learning models are being used that help classify malicious or anomalous activity within the network.

1.2. Motivations and Objectives

The challenging aspect of creating a machine learning based IDS for an IOT network is the lack of availability of datasets. Many industries are bound to data privacy and cannot grant public access to IoT network data. To overcome these issues, several researchers have simulated a test bed environment and data is gathered from it. This is later used for the purpose of network analysis, network improvement and for providing security solutions.

While several machine learning methods used to classify anomalies in the network have been studied in the past, a common trend seen across is the same simulated dataset is used for training and testing these models and then very high accuracy is reported. In an ideal life scenario, the data that is used to train a model would greatly differ when compared to the data that the model will run on when deployed in production. This can adversely impact the performance of the models when deployed. The aim of this research is to understand how the performance of the model varies when different datasets are separately used to train and test the model. We are using two publicly available simulated datasets to validate the performance of IDS built using machine learning models. The objective of this proposed research is to:

- Extract and process data with common features needed to perform comparison between the two datasets
- Compare the performance of different supervised machine learning methods on individual datasets for binary and multiclass attack classification
- Perform cross dataset evaluation using the same machine learning methods for binary and multiclass attack classification
- Analyze the challenges and performance results

The rest of the thesis is structured as follows. In Section 2, we perform literature survey to understand the datasets and methods used in previous papers. In Section 3, we overview the materials and methods required as part of this research. In Section 4, we discuss the results and challenges. In Section 5, we present the conclusions and future work.

2. LITERATURE SURVEY

In this section, we will discuss the different types of intrusion detection systems created for the purpose of identifying anomalous behavior in networks as well as in IoT networks, the different datasets used, and the evaluation metrics adopted.

Several datasets have been simulated for the use of IDS. One of the datasets that has been used for quite long for the purpose of building a predictive IDS is the KDD CUP 99 [4] [5] that has both training and test sets available. The dataset consists of normal and attack instances. The attacks are Denial of Service (DoS), probe attacks, Remote to Local(R2L) and User-to-root(U2R). Obeidat, I. et al. [6] in their paper used the KDD CUP 99 datasets and applied different machine learning models to create an IDS. The different machine learning models used were decision trees, naïve bayes, multilayer perceptron, random forests, and random tree. The AUC for each of these classifiers was reported as 0.969, 0.969, 0.990, 0.996 and 0.953, respectively. However, the paper uses accuracy scores to evaluate the different classifiers. It does not specify the scores for each category and all the features of KDD where taken into consideration.

A newer version of KDDCUP99 was introduced named the NSL-KDD as the former suffered with issues in redundancies in the training and test sets and class imbalance in attacks [7] [8]. The NSL KDD consists of different train and test files.

Gao X. et al. [9] in his paper performs multi-class classification using an ensemble machine learning model and compares accuracy with other machine learning methods. The accuracy on the test sets for decision tree is 79.71%, for random forests is 76.64%, on is 75.51%, Logistic Regression is 73.58%, SVM is 74.09%, DNN is 81.6% and adaboost is 76.02%. Their proposed ensemble machine learning method is that of a multitree that and gives the highest accuracy score of 84.23%. This paper uses accuracy metric to draw comparisons between the different model's performances. The scores for each classification label are not mentioned.

DH Deshmukh et al. [10] in their paper further improved the accuracy of the NSL KDD dataset by performing preprocessing on the training and test datasets prior to applying the machine learning models. Correlation based feature selection and data discretization was done on the data followed by which they report an accuracy of 88.20% for Naïve Bayes, 93.40% for Hidden Naïve Bayes and the highest of 94.20% for Naïve Bayes Tree. This paper uses accuracy metric to draw comparisons, which does not help in understanding the performance based on the class distribution of the datasets.

A relatively newer dataset named as ISCX-IDS-2012 was created for the purpose of network intrusion detection by Ali A. et al. [11] [12]. This dataset consists of wide array of different attacks within a network over a range of different network protocols such as HTTP, SMTP, SSH, POP3, IMAP, and FTP. M.J. Vargas et al. [13] used Bayesian networks to perform anomaly detection in networks using the ISCX-IDS-2012 dataset. Their paper reports accuracies across uniflow, bi-flows and aggregated flows as 99.95%, 99.92% and 99.92% respectively by generating flow-based features for the model. Their Bayesian network classifier heavily relies on the flow identification features like the source and destination IP address to perform prediction. This can cause an issue when the model comes across an attack from a different IP address which was not observed in the training set. A different approach was adopted by S.N. Mighan et al. [14] in his paper to perform anomaly detection on the UNB ICSX-IDS 2012 dataset. They used deep learning in combination with support vector machines to perform binary classification which provides an accuracy of 90.2%. The paper does not comment on the class imbalance and does not report weighted accuracies for their model.

A newer dataset named as CICIDS2017 was created that consisted of more common attacks such as DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Port scan and Botnet [15] [16]. This dataset consists of a total of 80 flow-based features generated using CICFlowMeter [17] [18]. S. Ustabay et al. [19] in their paper used the CICIDS2017 dataset to create an Intrusion detection system using recursive feature elimination by random forest classifier and deep learning model. The flow identifier features were eliminated. The original dataset was reduced by 95% and this was split into 80% for training and rest 20% for testing. The model performs binary classification and reports an AUC score of 0.96.

The CICIDS2017 dataset was used for binary classification by Arif Y. et al. [20] to create an IDS with a synthetic minority oversampling technique or SMOTE to overcome the class imbalance in the dataset. The flow identifier features were eliminated, and the dataset was split into 70% training and 30% test data. The AdaBoost machine learning algorithm was applied to a total of 25 features selected using principal component analysis (PCA) and accuracy of 81.83% was achieved with an AUC score of 0.92.

A more realistic network IDS dataset was created by Nour M. et. al [21] [22] that captured a wider range of attacks named UNSW-NB15. The attacks simulated within the network are Fuzzers, Analysis, Backdoors, DoS, Exploits, Reconnaissance, shellcode, and worms. The dataset has binary labels for attack or normal instance, attack category labels as well as further categorized sub-category attack labels with a total of 49 features. Mustapha B. et.al [23] implement a RepTree algorithm on UNSW-NB15 and NSL-KDD datasets and perform both binary and multiclass classification to create an IDS. The authors split the UNSW-NB15 into training and test sets and use feature selection followed by which they apply a two-level classifier(RepTree) and report an accuracy of 88.95% on UNSW-NB15 for binary classification and 81.28% for multiclass classification. As for NSL-KDD dataset, they report an accuracy of 89.95% for binary classification and 83.59% for multi-class classification. The paper does not report AUC for the different classes and does not take class imbalance into consideration. Moving onto unsupervised machine learning methods, Liu X. et. al [24] applied a feed forward neural network learning model on the UNSW-NB15 dataset that consists of 10 hidden layers for multiclass classification to detect anomalies. They perform 10-fold cross validation on 60% of training data, used 10% of data as validation set and achieve an accuracy of 99.5% on 30% test data and AUC of 1.0. The model uses all features and does not disregard the flow identifiers.

The above datasets mentioned so far were mainly used for Intrusion detection systems within networks and do not contain any smart devices that are present within an IoT network. An approach was made to simulate an IOT network in the Bot-IoT dataset by Nicholaos K. et al. [25] [26] by connecting smart devices and sensors within the network. Different attacks such as Port Scanning, DoS, Distributed DoS, Information theft were performed within the network and the instances were labelled accordingly. The authors extracted 10 best features which do not contain the flow identifiers from the entire set of 47 features and used three different classifier machine learning methods to train and test the dataset. They used support vector machines, recurrent neural networks, and LSTM- recurrent neural networks. 5% of the entire data was used for the purpose of training and testing. Results were reported for both binary classification and multiclass classification. The confusion matrices for binary were reported for all three classifiers and confusion matrices for RNN were reported for multiclass classification. In addition, the accuracy scores were for each attack category were mentioned. The AUC scores for SVM, RNN and LSTM-RNN were 0.976 ,0.99 and 1.00, respectively.

Another supervised learning approach with a newer algorithm – Bijective soft set technique was applied on the Bot-IoT dataset to classify attacks by Muhammad S. et. al [27]. The authors used Bayes Net, Naïve Bayes, C4.5 decision trees, Random Forest and Random Tree fed to the bijective soft set technique which uses different machine learning models to train and test. The ML method that presented the best results is returned by the bijective soft set technique. Three classifiers namely DT, RF and Random Trees reported an accuracy of 99.99% while Bayes Net and Naïve Bayes reported an accuracy of 99.77% and 99.79%, respectively for binary classification. The paper does not mention weighted accuracies or area under curve scoring metrics for the classification performed.

Another IOT network intrusion detection dataset (IoTID) was publicly made available by the HCR Lab [28] . This dataset simulates an IOT network by using smart devices. The dataset consists of different attacks and normal instances in the form of raw network packet files. Liu Z. et. al [29] used the IoTID dataset [28] to perform anomaly detection by using various machine learning methods. The packet data was extracted into CSV files and split into 75% for training data and 25% for testing data. The results for binary classification for Random Forest was the highest with an accuracy of 100%, followed by KNN with an accuracy of 99%. XGBoost algorithm gave an accuracy of 97%, logistic regression was 86% and the poorest performance was that of SVM with an accuracy of 79%. The confusion matrices for each method were reported to better evaluate the result. All the machine learning models rely only on the basic flow identifiers to perform binary classification. The weighted accuracies were not reported for any models to understand the class imbalance.

The IoTID [28] dataset paved way for the creation of another Intrusion detection dataset with more features., IoTID20 [30] [31]for the purpose of anomaly detection in IoT networks. The authors of the IoTID20 [30] [31] dataset also performed binary as well as multiclass classification on the dataset and reported the accuracy scores for different classifier methods by using cross validation. The F-Scores were also reported for binary classification and category classification. The decision tree classifier gave the highest accuracy of 88%, followed by ensemble classifier with an accuracy of 87%. The accuracies for Random Forest, Naïve Bayes, Linear Discriminant Analysis, Logistic Regression and SVM are 84%, 73%, 70%, 40% and 40%, respectively. The paper specifies the F1-score for each attack category class; however, it does not report AUC scores or weighted accuracies and takes the flow identifiers into account for features. The IDS created will fail to provide better results when deployed in production.

The latest dataset for an IoT network is the Ton-IOT Telemetry dataset [32] [33] that provides heterogenous data at different layers within a IoT network, i.e., edge layer, fog layer and cloud layer. The Ton-IOT telemetry dataset makes use of different sensors and smart devices, and a wide range of attacks were simulated. The data collected individually for each of these devices were combined to form a larger dataset and different machine learning models were applied to the individual datasets as well as the combined IIOT dataset to perform binary and multiclass classification. 80% of the data was used for training and 20% was used for testing, followed by which a 4-fold cross validation was performed on the training data. The accuracy scores for each of the models and devices were mentioned for both binary and multiclass classification. CART decision trees performed the best with an accuracy of 88% for binary and 77% for multiclass classification. The paper does not mention weighted accuracy or AUC to better evaluate the class distribution within the dataset. The datasets and the highest scoring machine learning model and the performance metrics reported are summarized in the Table 1 below.

Dataset	Paper	Classification	Highest Scoring ML Model	Performance Metric	Value
KDD-CUP 99	Obeidat, I. et al. [6]	Binary	Random forests	AUC	0.996
				Accuracy	0.99
NSL-KDD	Gao X. et al. [9]	Multi-class	Ensemble (Multitree)	Accuracy	0.8423
	DH Deshmukh et al. [10]	Binary	Feature selection + data discretization + Naïve Bayes Tree	Accuracy	0.942
	Mustapha B. et.al [23]	Binary	Two-level RepTree	Accuracy	0.8995
		Multi-class			0.8359
ISCX-IDS-2012	M.J. Vargas et al. [13]	Binary	Bayesian Networks	Accuracy	0.9995
	S.N. Mighan et al. [14]	Binary	SVM	Accuracy	0.9020
CICIDS2017	S. Ustabay et al. [19]	Binary	Recursive feature elimination by random forest classifier and deep learning model	AUC	0.96
	Arif Y. et al. [20]	Binary	PCA+ SMOTE + AdaBoost	Accuracy	0.8183
				AUC	0.92
UNSW-NB15	Mustapha B. et.al [23]	Binary	Two-level RepTree	Accuracy	0.8895
		Multi-class			0.8128
	Liu X. et. al [22]	Binary	Neural Network	AUC	1
Bot-IoT	Nicholaos K. et al. [23] [24]	Binary	LSTM-RNN	AUC	1
		Multi-class			

	Muhammad S. et. al [25]	Binary	Bijective soft set - Decision trees, Random Forest, and Random Tree	Accuracy	0.9999
IOTID	Liu Z. et. al [27]	binary	Random Forest	Accuracy	1
IoTID20	I. Ullah and Q. H. Mahmoud [28]	Binary	Decision Tree and Random Forest	F-Score	1
		Category Classification	Decision Tree	F-Score	1
		Sub-category classification	Decision Tree	Accuracy	0.87
Ton-IOT	A. Alsaedi. Et. al [30] [31]	Binary	CART	Accuracy	0.88
		Multi-class			0.77

Table 1: Literature Review Summary

As it can be seen from the above table, the accuracy reported for several models and datasets is extremely high. Several models take the basic flow identifiers like the IP addresses and timestamps into consideration while developing the IDS. Using such features to develop an IDS lead to achieving 100% accuracy for the models. Alternatively, the training and testing sets contain the same simulated attacks for a given dataset, which can present misleading performance metrics, since attacks simulated in a test environment can differ to ones the IDS will face when deployed. This gives rise to the need of cross-dataset evaluation to analyze the performance of a model. Several other domains have adopted the idea of evaluating the performance of machine-learning methods using a cross-dataset. In the field of neuroscience, M. Lorbach et. al [34] performed cross-dataset evaluation to recognize social behavior in rats. The authors use an interaction classifier and report the F1-score to evaluate their model. The f1-score when trained and tested with RatSI and Validation datasets respectively is 0.54 and when reversed gives an f1-score of 0.72. Similarly, Y.Chen et.al [35] perform cross-dataset evaluation for the purpose of activity recognition in humans using transfer learning. The accuracy is reported for after cross-evaluating with four different datasets.

F. Sha et.al [36] perform cross dataset evaluation for the purpose of answering visual questions. The models are trained using one dataset and evaluated using other datasets. Binary classification is performed using deep learning. The accuracy results reported within the dataset versus the results evaluated using other datasets report great differences. For instance, the VQA dataset reports an accuracy of 65.7% within the dataset and when evaluated with Visual7W dataset reports an accuracy of 53.4%.

Cross dataset evaluation thereby helps in understanding if an IDS created using a single dataset is likely to perform well when deployed.

3. MATERIALS AND METHODS

In this section, we describe the datasets used, preprocessing performed, features used, and the machine learning methods applied to the dataset. In addition, we also mention the classification performed and metrics used to evaluate the classifiers.

3.1. Terminology

3.1.1. Different types of attacks

An IoT network is vulnerable to different types of attacks and is prone to several weaknesses. Hardware, software, and network challenge can hinder the performance of an IoT network, additionally devices present within a smart home are vulnerable to Distributed DoS, DoS, malware, and impersonation attacks [37].

As part of this research, we are considering two of the attacks that are commonly observed in an IoT network and are described below

- Denial of Service (DoS): A denial of service or DoS attack occurs when legitimate users are unable to access resources, such as network, information system or devices due to the presence of a malicious code. This attack is established by inundating the network with traffic or flooding the information systems with requests so that it crashes or fails to respond to the legitimate user.

- Scanning: Scanning attacks occur when attackers can scan devices within a given network to gather information either by scanning ports, IP address or OS and version. These attacks help the adversaries to obtain personal information that can be later used to launch other attacks.

3.1.2. Classification

Classification refers to the task of assigning labels to examples with the help of machine learning algorithms. The value to be predicted is called as class label or target. It can also be viewed as a predictive modeling problem that can help distinguish one instance from another using the input variables and applying a function to obtain the target class.

- Binary classification: Binary classification is the task of classifying the output variables into two target variables or classes. The IoT network instances are mainly classified into two types.
    - Normal - This represents no attack on the network
    - Anomaly – This represents an attack on the network

- Multiclass or Category classification: Multiclass classification is the task of mapping machine learning algorithms on input variables to classify more than 2 targets variables or classes. The class labels for the attacks on the IoT network can be classified as follows.
    - Normal – This indicates no attack on the network
    - Scan – This indicates the type of attack on the IoT network is a scanning attack.
    - DoS – This indicates the attack on the IoT network is Denial of Service

3.2. CICFlowmeter

CICFlowmeter is a network traffic generator and analyzer software. It is used to understand flows present in the network packets. It can determine bi-directional flows based on the protocols – TCP and UDP. In addition, it generates a CSV file from raw network packet (pcap) files with several time-based features that can be used for network analysis. It is predominantly used for intrusion detection systems [17] [38] [18].

3.3. Datasets

To perform cross-dataset evaluation, this research uses two publicly available IoT network intrusion detection datasets i.e., Bot-IoT dataset [17] and the IoTID20[20] datasets. Both these datasets simulated the attacks caused on an IOT network.

3.3.1. Bot-IoT Dataset

The Bot-IoT [25] [26] dataset set up a testbed with simulated IoT devices, normal and attacking virtual machines and network devices. The Node-red tool [39] was used to simulate 5 IoT devices i.e., the weather station, smart fridge, motion activated lights, smart garage door, and a smart thermostat. In addition, a firewall and two network interface cards were configured into the testbed environment. The IoT simulated devices were connected to the Ubuntu server. The normal traffic data was generated using Ostinato network monitoring and testing tool [40] that was used to extract data from the target machines which were ubuntu mobile, windows machines and ubuntu server. Kali Linux machines were used to simulate the attacks. The attacks simulated were DDoS, DoS, Scanning and information theft. The data collected was then labelled based on the hacking machines IP addresses and features were extracted. [25] [26]. The testbed architecture is presented in the Figure 2 below.

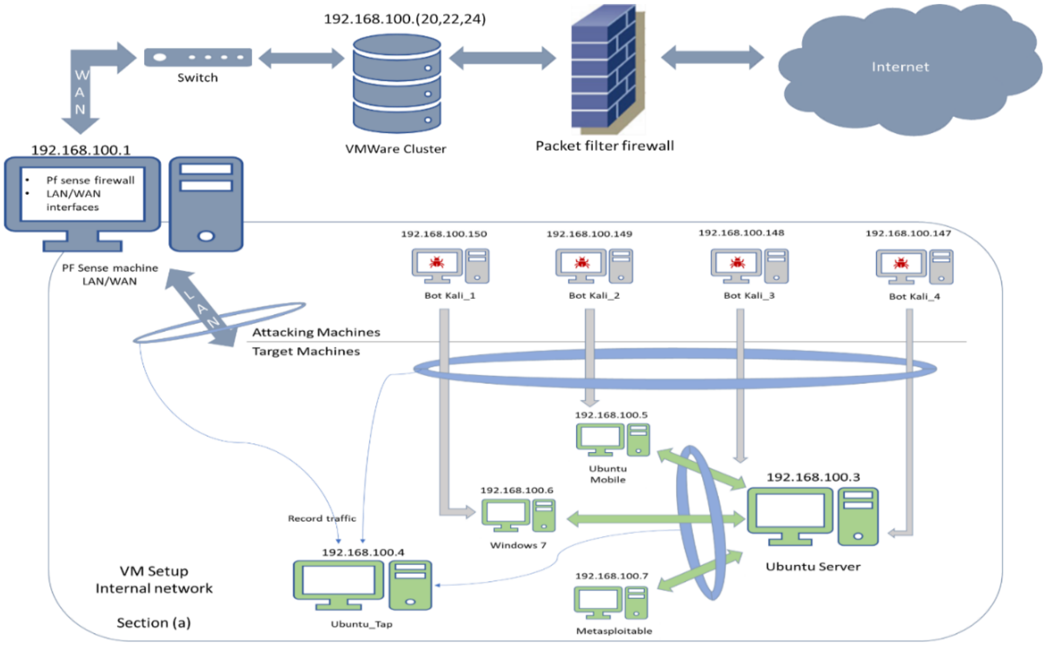

Figure 2: Bot-IoT Testbed Architecture [25] [26]

The dataset consisted of 69.3 GB of raw network packet files. The formatted csv files extracted amounted to 16.7GB. A smaller set was extracted for the purpose of training and testing that contains over 3 million records. The scanning attacks were further classified into OS scan and Port Scan. DDoS and DoS attacks were categorized further based on the protocols i.e., TCP, HTTP and UDP. The scanning attacks were simulated using the Nmap and Hping3 software and the DoS attacks were simulated using the Hping3 tool.

3.3.2. IOTID20 Dataset

The IoTID20 [30] used the raw network packet files created by the IoTID [28] dataset. For the purpose of creating an IoT network, two smart home devices – SKT

NUGU(NU 100) which is an AI-based speaker and EZVIZ Wi-Fi Camera (C2C Mini O Plus 1080P) along with different smart phones and laptops were connected to the same wireless network using a smart home Wi-Fi router. The network packet files were then captured using the wireless adaptor's monitor mode. Attacks like DoS, Scanning and Man in the middle were simulated using Nmap tools. The attacks packets for Mirai botnet were generated separately using a laptop and were later changed to simulate its origination from the IoT devices [28]. The testbed architecture is presented in the Figure 3 below.

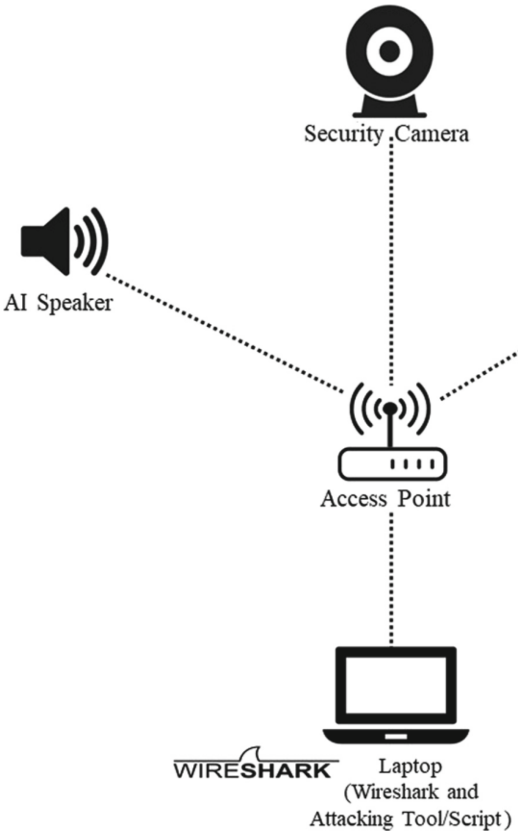

Figure 3 : IoTID20 testbed environment [31] [28]

The IoTID20 dataset used these packet files and created CSV files using the CICFlowMeter [18] [17]. The CSV files generated were then labelled accordingly for anomaly and types of attacks based on the IP addresses. The distribution of the dataset is mentioned in Table 2 and Table 3 below.

Binary Label	**Instances**
Anomaly	585710
Normal	40073

Table 2 : IoTID20 Binary Data Distribution

Category	**Instances**		
DoS	59391		
Mirai	415677		
MITM	35377		
Scan	75265	PortOS	53073
		HostPort	22192

Table 3 : IoTID20 Anomaly Data Distribution

3.4. Machine Learning Methods

Different machine learning based classifiers are used for the purpose of predicting binary and multiclass labels. The following machine learning methods described have been used as part of this research

3.4.1. Naïve Bayes

The Naïve Bayes classifier uses the Bayes theorem to perform classification. This model assumes the conditional independence of the features given the target. X represents the features of the dataset and Y represent the target class or variable. The likelihood of an instance belonging to a certain class can be determined using the formula below. This can be extended to perform both binary and multiclass classification.

$$P(X_1 \dots X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$$

3.4.2. Logistic Regression

Logistic Regression machine method models on “sigmoid function” or “logistic function” to perform predictive analysis. It helps predict the likelihood of a target occurring for a given instance. [41].

3.4.3. KNN

K-Nearest Neighbor or k-NN machine learning method is also called as a lazy learning algorithm. It is used for the purpose of predictive analysis and performs classification of a given instance based on the Euclidean distance from its “k” nearest neighbors. It then labels the instances based on the majority class value of its neighbors.

3.4.4. Decision Trees

A decision tree machine learning model is a simple and widely used supervised machine learning method that can perform both binary and multiclass classification. This method models on decision trees where the internal nodes are features. The path can be viewed as a classification rule and leaf node represents label or class.

3.4.5. Ensemble

Ensembles makes use of different machine learning methods to perform predictive analysis to classify a given task at hand. Random Forest is an ensemble of decision trees that can be used to provide better performance and results as they are less prone to overfitting when compared to decision trees.

3.5. Evaluation Methods

It is crucial to use the right metric to evaluate a machine learning model's performance. If the correct evaluation measure is not used, the machine learning model may perform poorly when it is deployed in real life. Some standard evaluation metrics are described below.

- Confusion Matrix: A confusion matrix is a N x N table that can help determine the performance of a machine learning model. From the confusion matrix one can infer the following
    - True Positive (TP): The actual value is true, and model predicts true.
    - False Positive (FP): This is also called as Type I Error; the actual value is false, and the model predicts true.
    - True Negative (TN): The actual value is false, and model predicts false
    - False Negative (FN): This is also called as Type II Error, the actual value is true, and model predicts false.
- ROC-AUC: ROC stands for Receiver Operating Characteristics. ROC is a curve mapped with model's sensitivity versus the model's false positive rate. These values can be calculated using the below formulae.

- Sensitivity (True Positive Rate): This is the fraction of positive instances that the model classifies as positive correctly. Sensitivity can be calculated as

$$TPR = \frac{TP}{FN + TP}$$

- Specificity (True Negative Rate): This is the fraction of negative instances that the model classifies as negative correctly. Specificity can be calculated as

$$TNR = \frac{TN}{TN + FP}$$

- False positive Rate: This is the fraction of negative instances that the model misclassifies as positive. FPR can also be viewed as 1-specificity and can be calculated as

$$FPR = \frac{FP}{TN + FP}$$

ROC curve of a model is not affected by imbalanced classes. AUC or Area under the curve is a single value that calculates the area under the ROC curve. AUC values range from 0 to 1.0 which can help determine the performance of the model. The higher AUC value, the better the model performs. So the AUC of 1.0 indicates that the models classifies all the instances correctly. AUC of 0.5 can be obtained using the baseline of random classification.

3.6. Methodology

In this section, the methods used for data extraction, labelling and data cleaning are mentioned followed by the features and testbed used to perform machine learning experiments.

3.6.1. Data Preparation

The Bot-IoT dataset and the IoTID20 dataset can be accessed publicly from [25] and [31] respectively. The training and testing files presented in the Bot-IoT dataset cannot be directly used for cross-dataset evaluation, as the features are not in sync with the IOTID20 dataset. For this research, only scanning and DoS attack instances are considered, since the simulation of the Mirai Botnet or DDoS attacks vary highly between the two datasets. The process of extracting newer features from the raw packet files and labelling the records is described in detail.

3.6.1.1 Extracting data

The Bot-IoT dataset [26] [25] consisted of 69.9 GB of raw network packet files. We employed the scheme used by IoTID20[20] to extract similar features. The raw network packet files were downloaded. As this research intends to classify between

DoS, Scanning and normal features only, the respective raw packet files were used and flow-based features were extracted using the CICFlowmeter [38] [17] [18]. A total of 83 features were extracted.

3.6.1.2. Labelling

The generated CSV file from above was then labelled using the labelled CSV files presented in the Bot-IoT dataset [25] [26]. The files were loaded to a MySQL table and the flow identifiers such as the source ip address , destination ip address , port numbers and protocols were used to add binary labels and category attack labels which are DoS, Scan and Normal.

3.6.1.3. Data Cleaning

The IoTID20 and the Bot-IoT files were checked for duplicates and the duplicated instances were deleted. A total of 229029 instances were deleted from the IotID20 dataset. The data extracted from the Bot-IoT dataset consisted of huge files, to be able to draw better comparisons, the instances for DoS, Scan and normal were extracted using python scripts. The table below mentions the number of instances used for each dataset. The subcategory column within the IoTID20 dataset was also removed. The data distribution for these two datasets is mentioned below in Table 4 and Table 5 for bot-IoT dataset and Table 6 and Table 7 for IoTID20.

Bot-IoT dataset

Binary Label	Instances
Anomaly	77573
Normal	4938

Table 4 : Bot-IoT Binary Data

Bot-IoT Anomaly Distribution

Category	Instances
DoS	61164
Scan	16409

Table 5 : Bot-IoT Anomaly Data

IoTID20 dataset

Binary Label	Instances
Anomaly	77157
Normal	38598

Table 6 : IoTID20 Binary Data

IoTID20 Anomaly Distribution

Category	Instances
DoS	59390
Scan	17767

Table 7: IoTID20 Anomaly Data

3.6.2. Features

The time and flow based features extracted using CICFlowmeter [38] [17] [18] and the description for each of these features is mentioned in Table 8 below.

Feature Name	Feature Description
Flow ID	Flow Identifier
Src IP	Source IP Address
Src Port	Source Port Number
Dst IP	Destination IP Address
Dst Port	Destination Port Number
Protocol	Internet Protocol used
Timestamp	Timestamp of the packet
Flow duration	Duration of the flow in Microsecond
total Fwd Packet	Total packets in the forward direction
total Bwd packets	Total packets in the backward direction
total Length of Fwd Packet	Total size of packet in forward direction
total Length of Bwd Packet	Total size of packet in backward direction
Fwd Packet Length Min	Minimum size of packet in forward direction
Fwd Packet Length Max	Maximum size of packet in forward direction
Fwd Packet Length Mean	Mean size of packet in forward direction
Fwd Packet Length Std	Standard deviation size of packet in forward direction
Bwd Packet Length Min	Minimum size of packet in backward direction
Bwd Packet Length Max	Maximum size of packet in backward direction
Bwd Packet Length Mean	Mean size of packet in backward direction
Bwd Packet Length Std	Standard deviation size of packet in backward direction

Flow Bytes/s	Number of flow bytes per second
Flow Packets/s	Number of flow packets per second
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow
Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Total	Total time between two packets sent in the forward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Fwd PSH flags	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Length	Total bytes used for headers in the forward direction
Bwd Header Length	Total bytes used for headers in the backward direction
FWD Packets/s	Number of forward packets per second
Bwd Packets/s	Number of backward packets per second
Packet Length Min	Minimum length of a packet
Packet Length Max	Maximum length of a packet
Packet Length Mean	Mean length of a packet
Packet Length Std	Standard deviation length of a packet
Packet Length Variance	Variance length of a packet
FIN Flag Count	Number of packets with FIN
SYN Flag Count	Number of packets with SYN
RST Flag Count	Number of packets with RST
PSH Flag Count	Number of packets with PUSH
ACK Flag Count	Number of packets with ACK
URG Flag Count	Number of packets with URG
CWR Flag Count	Number of packets with CWR
ECE Flag Count	Number of packets with ECE
down/Up Ratio	Download and upload ratio
Average Packet Size	Average size of packet
Fwd Segment Size Avg	Average size observed in the forward direction
Bwd Segment Size Avg	Average number of bytes bulk rate in the backward direction
Fwd Bytes/Bulk Avg	Average number of bytes bulk rate in the forward direction
Fwd Packet/Bulk Avg	Average number of packets bulk rate in the forward direction
Fwd Bulk Rate Avg	Average number of bulk rate in the forward direction
Bwd Bytes/Bulk Avg	Average number of bytes bulk rate in the backward direction
Bwd Packet/Bulk Avg	Average number of packets bulk rate in the backward direction
Bwd Bulk Rate Avg	Average number of bulk rate in the backward direction
Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction

Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
Fwd Init Win bytes	The total number of bytes sent in initial window in the forward direction
Bwd Init Win bytes	The total number of bytes sent in initial window in the backward direction
Fwd Act Data Pkts	Count of packets with at least 1 byte of TCP data payload in the forward direction
Fwd Seg Size Min	Minimum segment size observed in the forward direction
Active Min	Minimum time a flow was active before becoming idle
Active Mean	Mean time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Idle Min	Minimum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active
Label	Anomaly or Normal
Cat	Category of attack or Normal

Table 8 : Feature Description

3.6.2.1 Feature Removal

To ensure that the model performs well, it was important to delete the flow identifiers of a network packet like the source and destination IP address, the port numbers, and the timestamp. If the model is trained using these features, it will fail to generalize well when deployed as attackers can use different IP addresses and times to launch attacks on the network. In addition to this, 10 more features were deleted as they consisted of a single value after the data was extracted using the CICFlowmeter [38] [17] [18] across the dataset and would not add any value to the machine learning models. The features deleted are mentioned in the Table 9 below. A total of 67 features were then used to train the models.

Features Deleted	Flow ID, Src IP, Dst IP, Src Port, Timestamp ,Dst Port, Fwd PSH Flags, Fwd URG Flags, Fwd Byts/b Avg, Fwd Pkts/b Avg, Fwd Blk Rate Avg, Bwd Byts/b Avg, Bwd Pkts/b Avg, Bwd Blk Rate Avg, Init Fwd Win Byts, Fwd Seg Size Min

Table 9 : Deleted Features

3.6.2.2. Feature Ranking and Distribution

Information gain is a measure that helps determine how informative a feature is achieved by decreasing the uncertainty or entropy of the dataset [42]. Higher information gain has lower entropy. In order to understand how the features are distributed and the information gain for features, we used WEKA [43] software to calculate the information gain for each feature using the ranker method. The Table 10 below mention the top 10 features for Bot-IoT dataset and IoTID20 dataset.

Bot-IoT Dataset	**IoTID20 Dataset**
Pkt Size Avg	Flow Duration
Pkt Len Mean	Flow Pkts/s
Tot Len Bwd Pkts	Idle Mean
Subflow Bwd Byts	Idle Max
Bwd Pkt Len Max	Flow IAT Max
Bwd Seg Size Avg	Flow IAT Mean
Bwd Pkt Len Mean	Init Bwd Win Byts
Pkt Len Max	Bwd Pkts/s
Flow Byts/s	Bwd IAT Tot
Flow Duration	Idle Min

Table 10 : Important features using Information Gain

To better understand the how features are distributed across both models, the bell curve for both the datasets is visualized using the RapidMiner Studio [44].

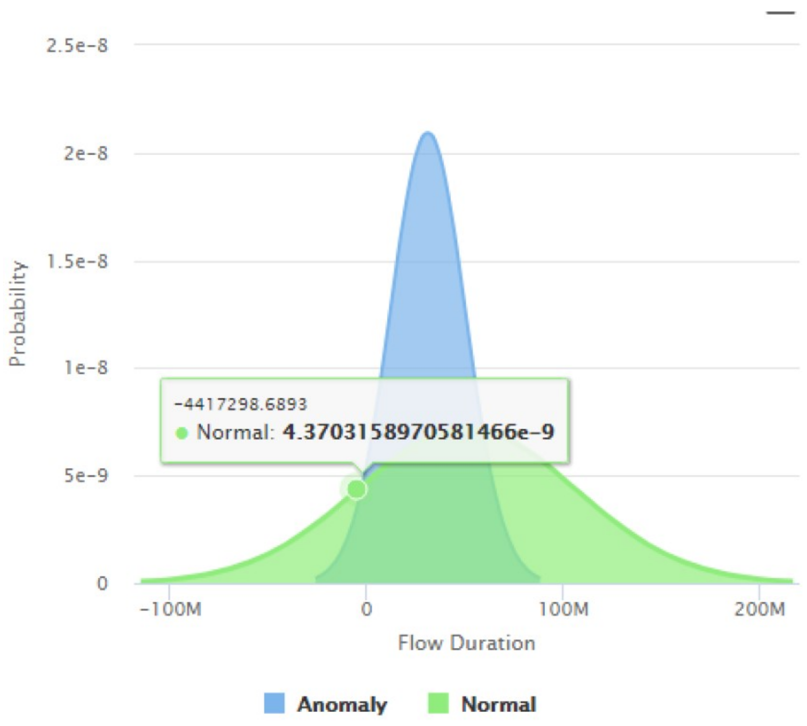

Figure 4 : Flow Duration distribution for Bot-IoT dataset

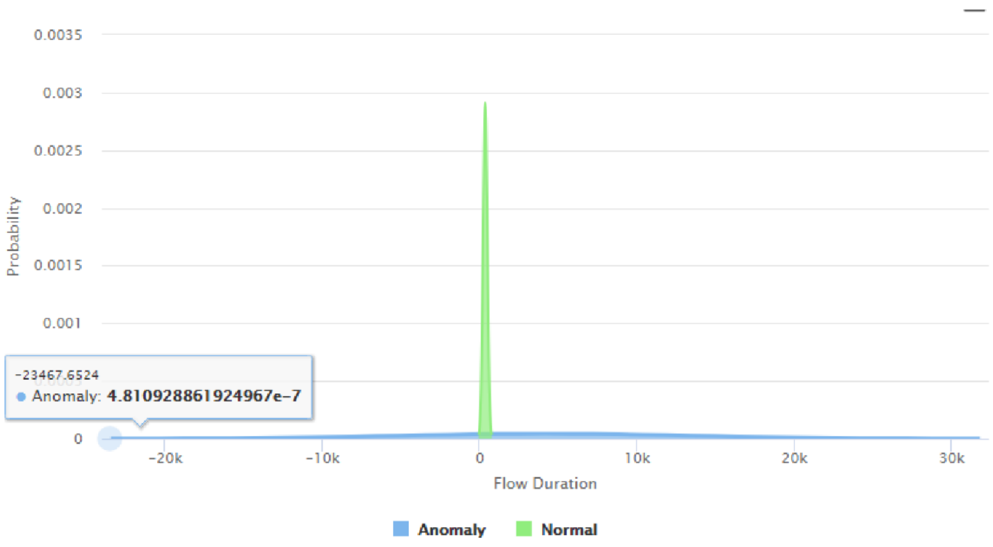

Figure 5 : Flow Duration distribution for IoTID20 dataset

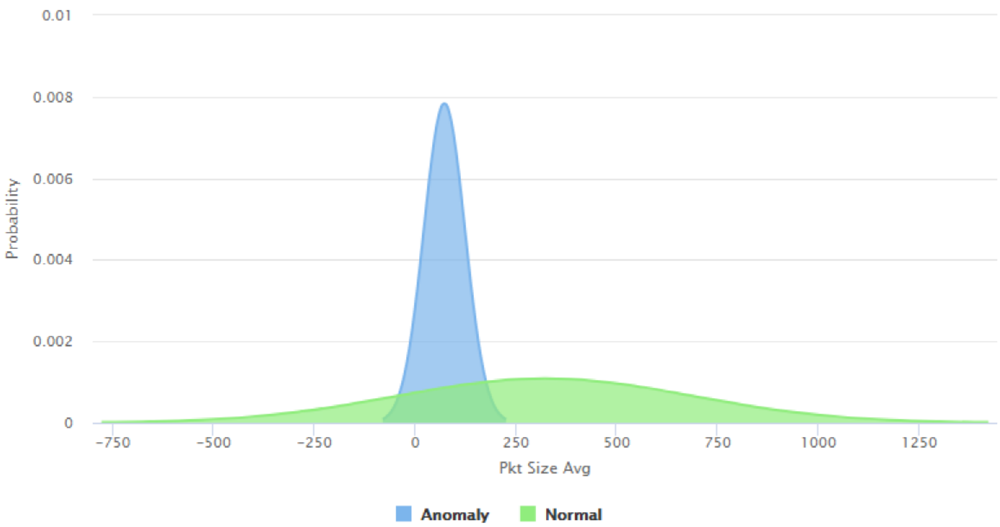

Figure 6 : Packet Size Average distribution for Bot-IoT dataset

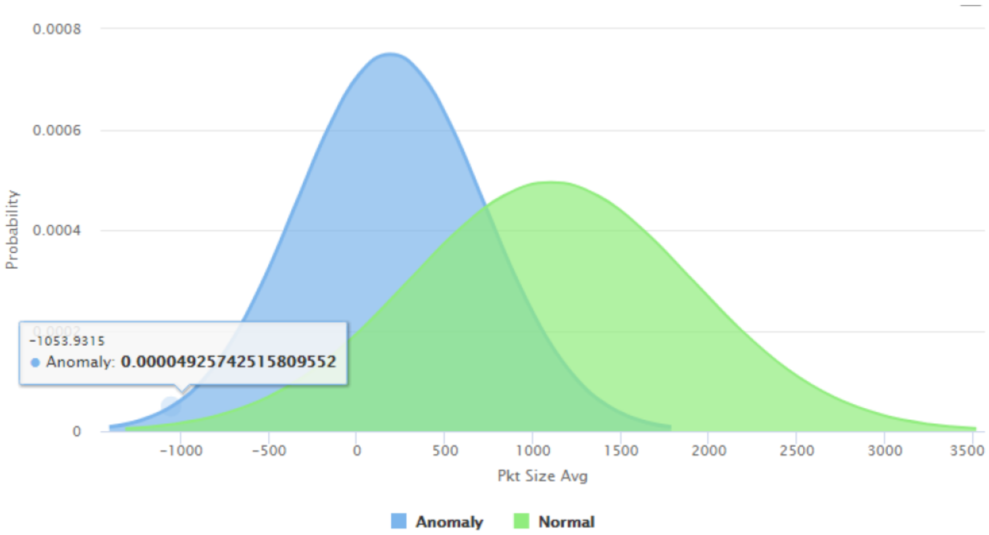

Figure 7 : Packet Size Average for IoTID20 dataset

As it can be inferred from the above images the features are not evenly distributed across both the datasets. The distribution varies because of the different testbed used for data simulation. A given network can have packets with varying length as well as the flow duration for each packet can vary differently. An effective IDS should be able to reflect these different distributions.

3.6.2.3. Scaling

Scaling is the technique ensures that different values present for a given feature can be fit into a common range so that the outliers or larger values present for that feature do not dominate the performance of a model. Different techniques are present to scale the data like standard scaling, min-max scaling, and robust scaling. Robust scaling is obtained by subtracting the median of the values from the value itself divided by the interquartile range of the values present in the feature [45]. The formula to perform robust scaling for values of a feature is described below.

$$Scaled_{value} = \frac{Current_{value} - median_{allvalues}}{Interquartilerange_{allvalues}}$$

Since the data was not normally distributed across different features, a robust scaling was performed to help enhance the performance of machine learning algorithms. Robust scaling was opted as it is less prone to being misled by the outliers when compared to standard scaling or min-max scaling. [46]

3.6.3. Testbed and Experimental Setup

For all experimental results presented in this section, we used 64-bit Windows 8 operating system on a PC with 1.80 GHz Intel core i7 CPU, 4MB cache and 8GB of RAM. The programming language used was python and data formatting were done using Pandas [47] which is a data analysis library. The machine learning models were implemented using the Scikit-Learn library [46].

4. RESULTS AND DISCUSSIONS

In this section, we discuss the results for different supervised machine learning models. Section 4.1 reports the model’s performance using the same dataset by splitting the data into 80% training and 20% testing. Section 4.2 reports the model’s performance with cross-dataset.

4.1. Comparison of results within same datasets

4.1.1. Binary Classification

The AUC weighted average was calculated for each model on the Bot-IoT dataset and the values are plotted in the Figure 8 .

A 10-fold cross evaluation was performed on the training data for parameter selection for the IoTID20 binary classification. The parameters selected for the decision tree classifier involve increasing the minimum number of samples needed to split a node from 2 to 8, increasing the minimum number of samples needed at the leaf from 1 to 10 and the maximum features to look when splitting a node is the square root of all features. This model was used for both binary and category classification. The parameters chosen for k-nearest neighbor method was setting the k value to 31, algorithm used to compute the nearest neighbors was ball tree and the weights function was set to distance (i.e., neighbors closer to the instance to be classified have higher influence compared to the neighbors far away) [46]. The ensemble used was a bagging classifier which used decision trees as the base estimator and the number of estimators used were 150. This model was used for both the datasets. As for the Bot-IoT dataset, all the other models performed very well with the default parameters in the scikit-learn library [46].

The AUC weighted average for IoTID20 dataset and is presented in the Figure 9.

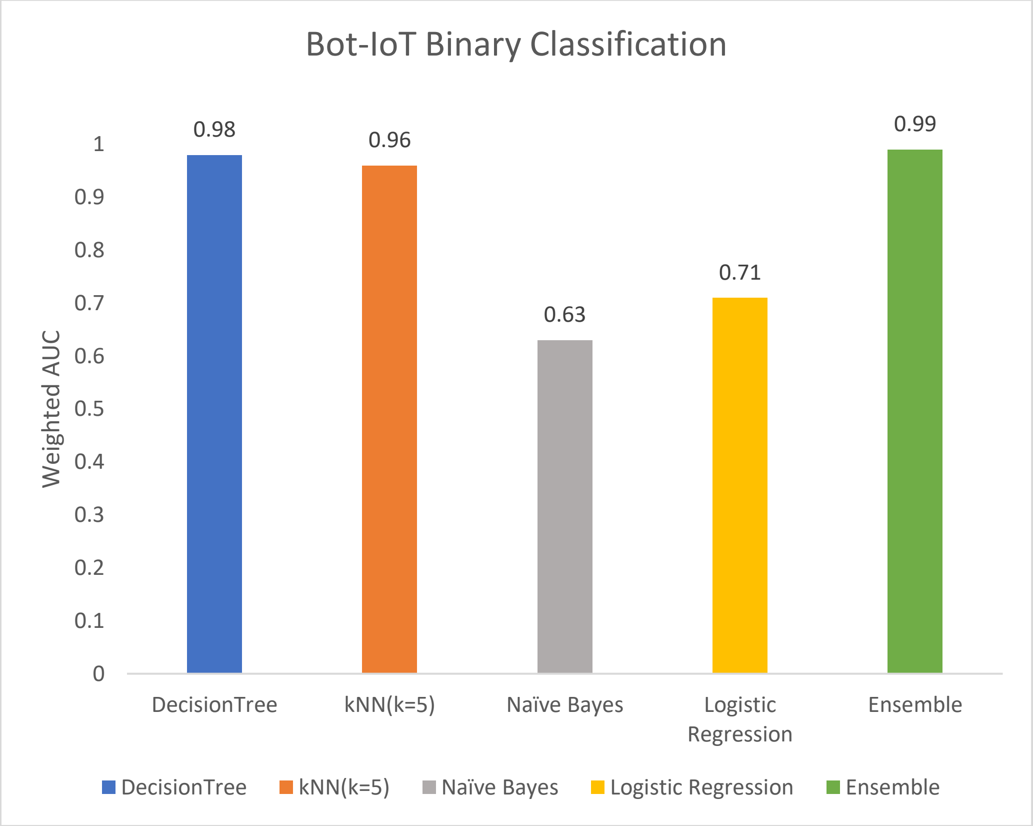

Figure 8: Weighted AUC scores for Bot-IoT Binary Classification

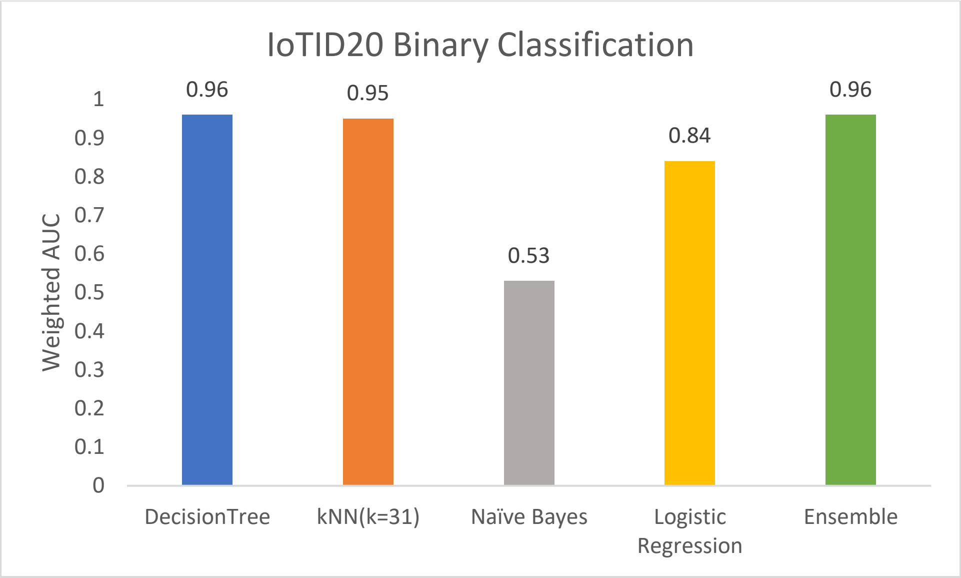

Figure 9: Weighted AUC scores for IoTID20 Binary Classification

As observed from the above two figures, the models perform very well and can detect anomalies within the dataset. The scores reported for decision tree, k-NN and ensemble are above 0.95 for both the datasets.

4.1.2. Category Classification

A 10-fold cross evaluation was performed on the training data for parameter selection for the Bot-IoT category classification. The parameters selected for the decision tree classifier involve increasing the minimum number of samples needed to split a node from 2 to 8, increasing the minimum number of samples needed at the leaf from 1 to 50 and the maximum features to look when splitting a node is the square root of all features. The parameters chosen for k-nearest neighbor method was setting the k value to 11. As for the IoTID20 dataset, all the other models performed very well with the default parameters in the scikit-learn library [46].

The AUC value for each attack class category and the AUC weighted average for each model is plotted for both the Bot-IoT and IoTID20 datasets and presented in the figures below.

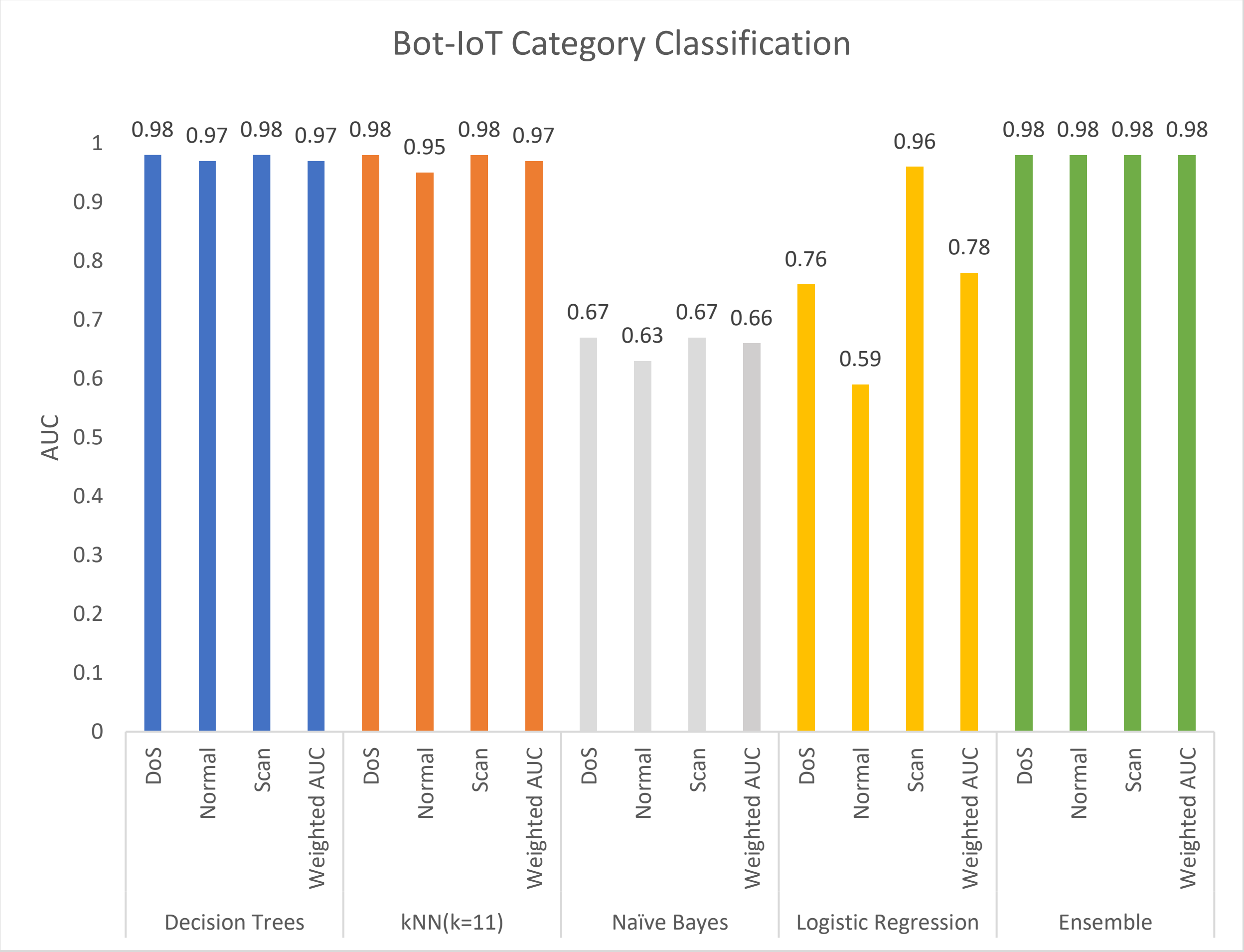

Figure 10 : AUC value for each category for Bot-IoT

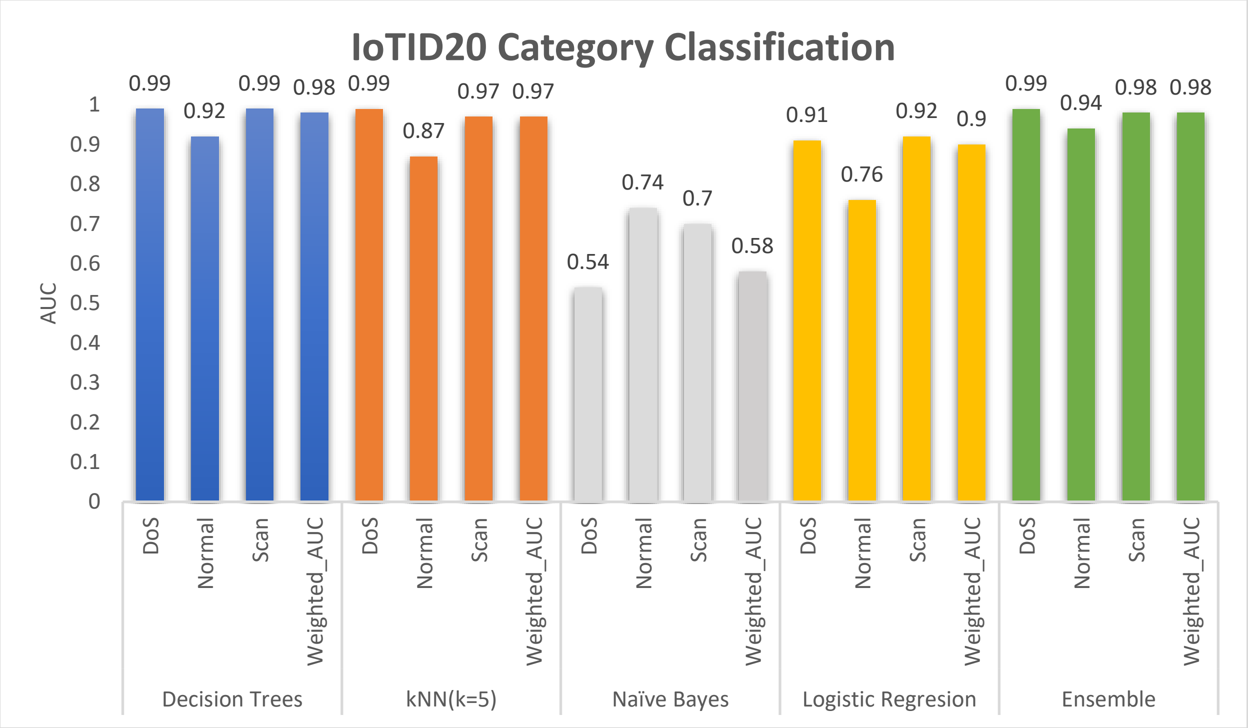

Figure 11: AUC for each category class for IoTID20

As observed from the above two figures, the models also perform and can categorize the different attacks and normal instances for both the datasets. The scores reported for decision tree, k-NN and ensemble are above 0.95 for both the datasets.

4.2 Cross dataset evaluation results

The above models where then tested with cross-dataset for both binary and category classification and the results are described below.

4.2.1. Binary Classification

Binary classification was performed using the same models and the AUC weighted average was calculated with cross-datasets.

The weighted AUC for binary classification with IoTID20 as train set and Bot-IoT as test set and vice-versa are plotted in the below figures, Figure 12 and Figure 13 respectively.

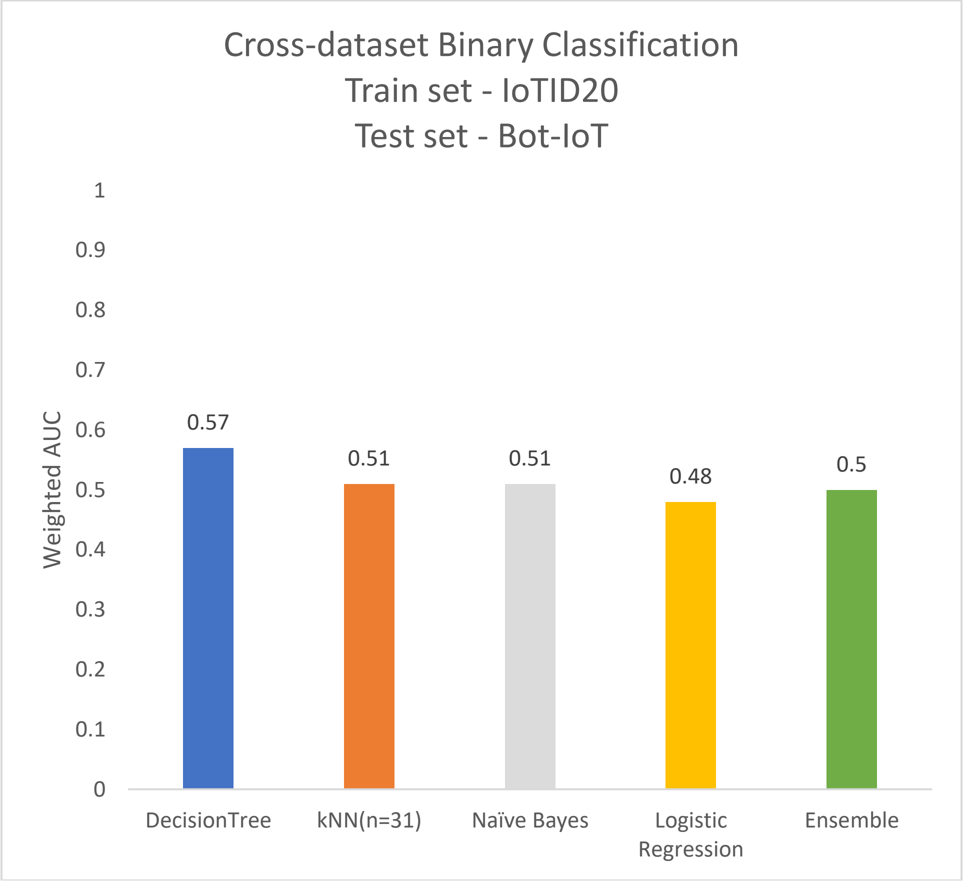

Figure 12 : Weighted AUC for Cross-dataset Binary Classification with IoTID20 as training set and Bot-IoT as test set

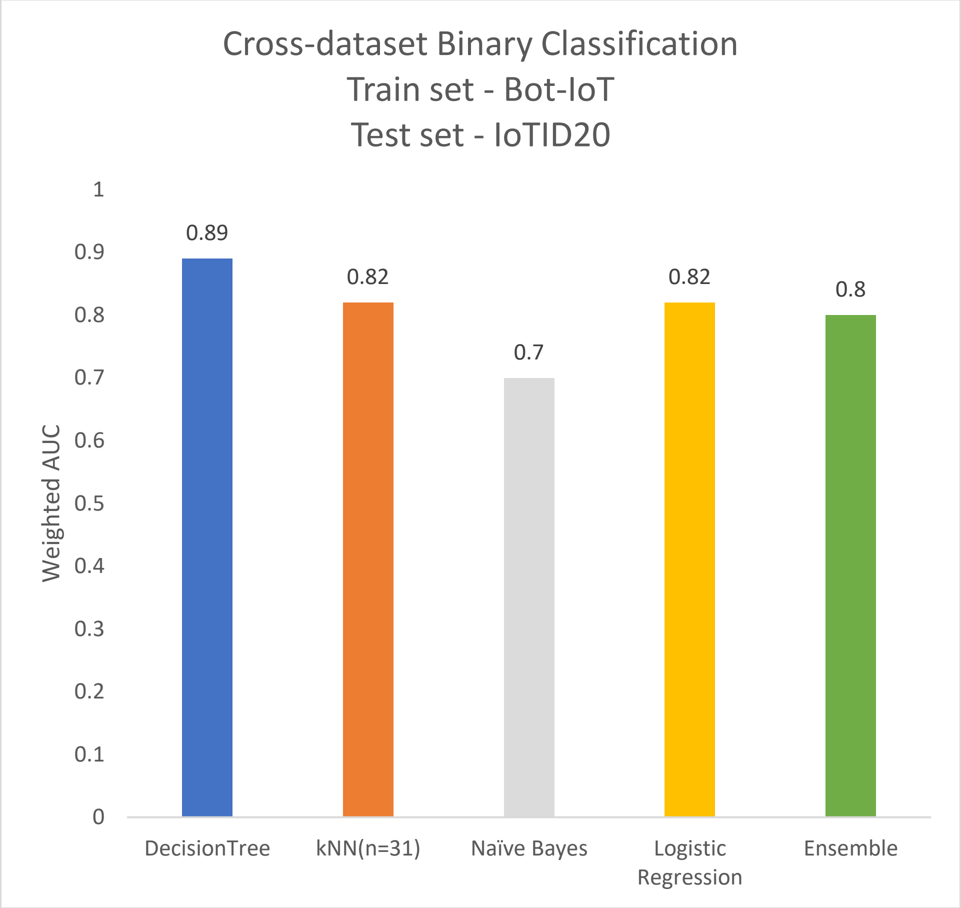

Figure 13 :Weighted AUC for Cross-dataset Binary Classification with Bot-IoT as training set and IoTID20 as test set

4.2.2. Category Classification

Category classification was performed using the same models and the AUC weighted average was calculated for each category class. The evaluation was performed with cross-datasets.

The AUC for each category attack and the weighted AUC with IoTID20 as train set and Bot-IoT as test set and vice-versa are plotted in the below figures, respectively.

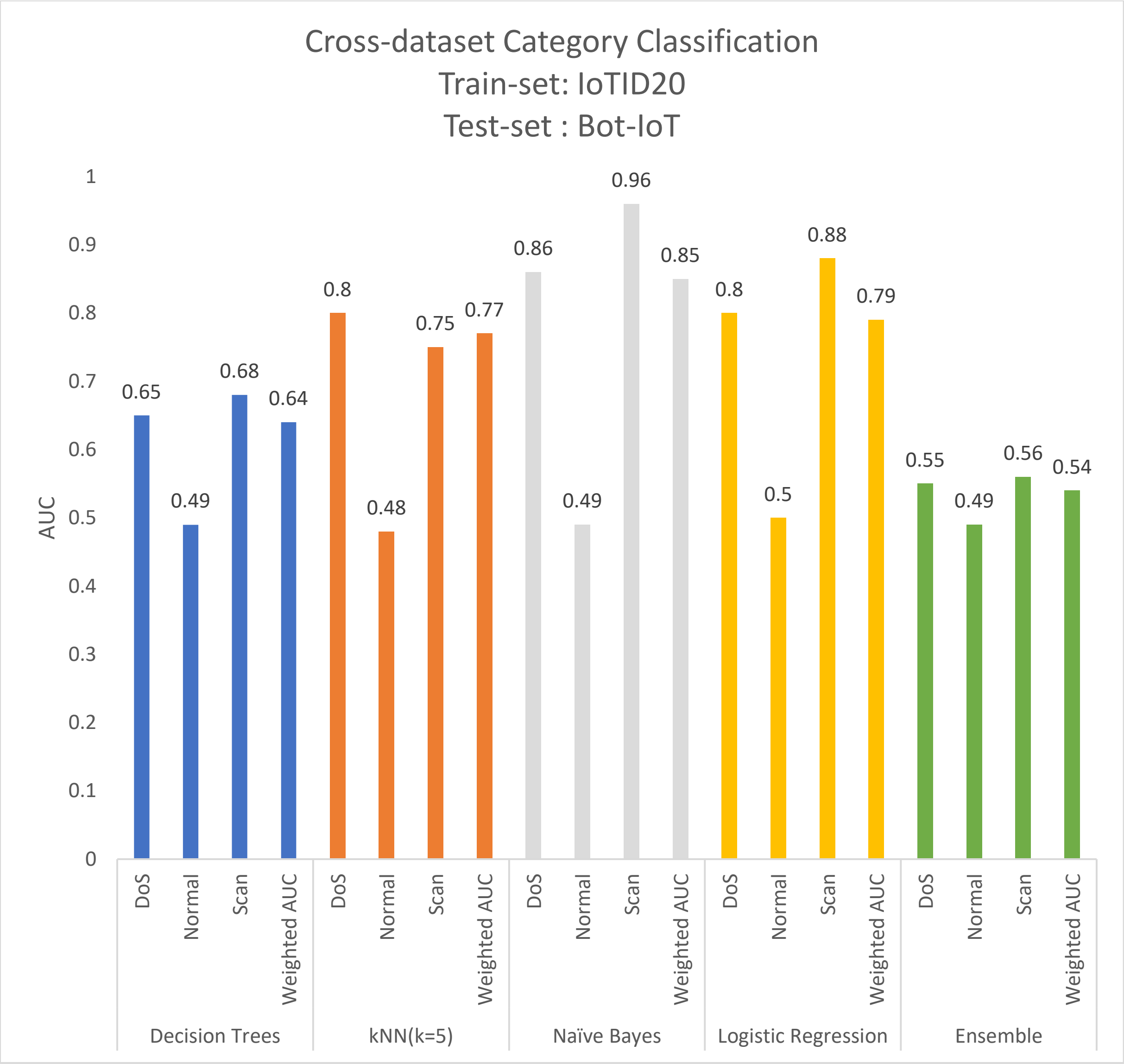

Figure 14: AUC for Cross-dataset Category Classification with training set as IoTID20 and test set as Bot-IoT dataset

As it can be inferred from the above figure, the performance of the model is poor compared to when the testing is performed within dataset.

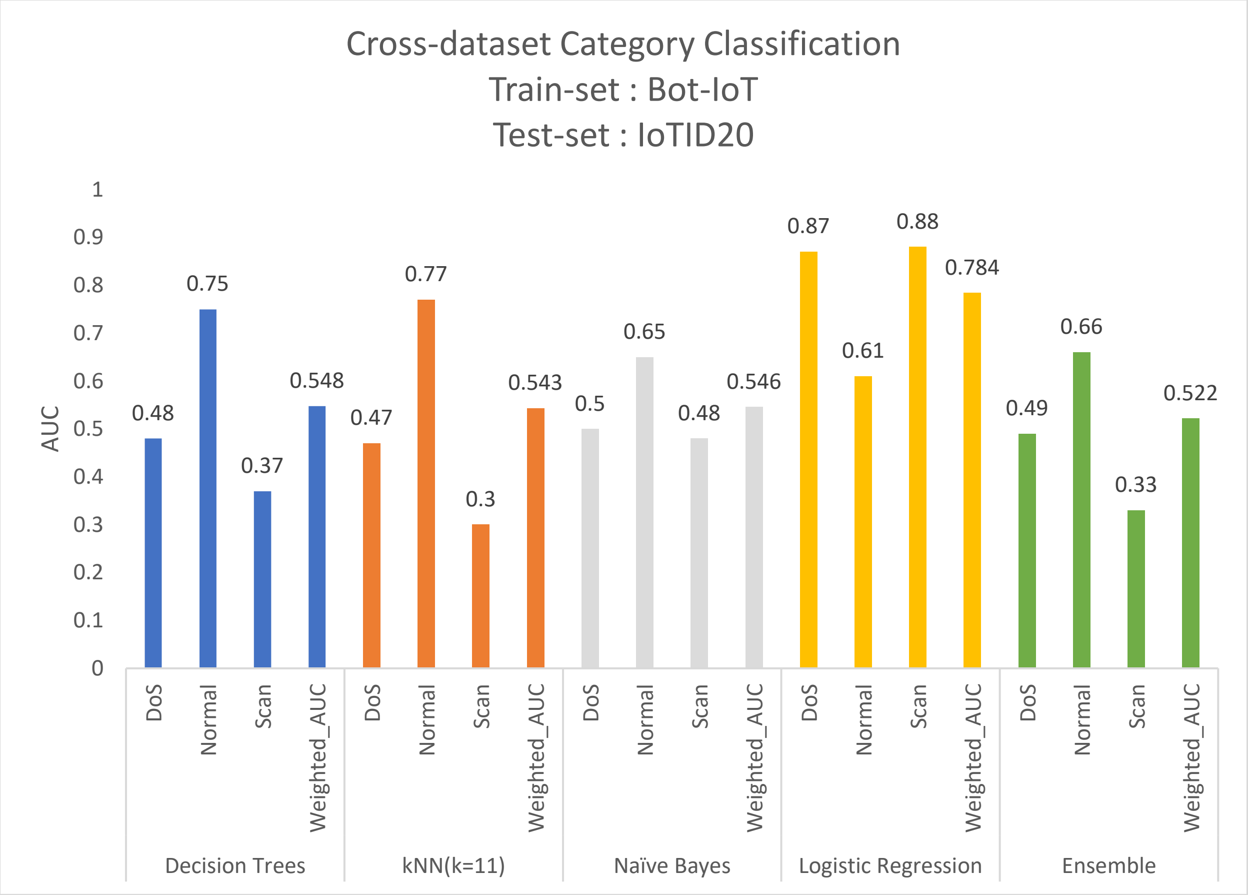

Figure 15: AUC for Cross-dataset Category Classification with training set as Bot-IoT and test set as IoTID20 dataset

As it can be inferred from the above figure, the performance of the model is very poor for normal instances as well as attack categories compared to when the testing is performed within dataset.

4.3. Discussion

From the above results, it can be observed that binary and category classification on the Bot-IoT dataset achieves high scores for all the methods except Naïve Bayes and Logistic Regression. This is because attacks can be identified based on rules that can be easily learnt by decision trees, but it is not easy for probabilistic models to learn. Similarly, binary and category classification on the IoTID20 dataset achieves high scores for all methods except for Naïve Bayes. When the models are trained with the IoTID20 dataset and tested with the Bot-IoT dataset, they perform very poorly and cannot identify the normal instances. This is because normal data generated by the dataset was done for using the network adaptor's monitor mode. This mode is limited to a single wireless channel and is prone to corrupted packets as error detection is not performed, whereas the normal data in the Bot-IoT data was generated using the Ostinato tool [40]. This tool can generate traffic via multiple streams and capture traffic for different protocols and test errors. Hence the IoTID20 was unable to generalize well for Bot-IoT. However, the models can detect anomalies very well. An IDS built with these models would cause issues and not allow normal or safe traffic to pass within the IoT network. In contrast, all the models except for Naïve Bayes trained with the Bot-IoT dataset and tested with the IoTID20 dataset perform relatively well but the AUC scores obtained are lower compared to their performance tested with the same dataset. The decision trees perform the best and can classify the normal and anomaly instances. Therefore, cross-dataset evaluation for binary classification on both the datasets provide lower AUC score compared to within dataset.

Although the primary aim of an IDS is to perform anomaly detection, to better understand the behavior of the models on the datasets, category classification is performed. Logistic Regression performs better compared to the other methods when trained with Bot-IoT and tested with IoTID20 and can capture the signal needed to classify DoS and Scan attacks. The other models built using the Bot-IoT dataset do not perform well with attack category classification for DoS and Scan. The attacks simulation method on the Bot-IoT dataset differs from that of the IoTID20 dataset and default packet sizes were used to simulate these attacks. On the other hand, the models trained with IotID20 can categorize the attacks well but fail with the normal instances. This is again because the dataset does not generalize well with normal instances. This indicates that IDS created using the same training and test data is likely to perform well but perform poorly when deployed because the traffic for a given network when deployed can greatly differ from the simulated data.

4.4. Challenges

A lot of shortcomings were observed as part of this research. We only found two IoT intrusion detection network datasets that performed similar attacks and could be used for comparison. In addition, most datasets simulated are done for different attack categories. The test-bed setup, simulation methods and tools adopted for the generation of the dataset vary which can greatly alter model’s performance and make it difficult to evaluate an IDS using cross-dataset. The data available cannot be directly used for comparison because the features generated for different datasets differ highly. Generating newer features from the raw packet files and labelling them is a challenging task.

5. CONCLUSION AND FUTURE WORK

It is crucial to use the right way to evaluate the performance of a machine learning model used to build an intrusion detection system. An IDS trained and tested using a single dataset may not perform well when deployed as the data observed in real life can vary a lot from the simulated data. The Bot-IoT dataset and IOTID20 datasets perform exceptionally well when trained and tested within the same dataset because the train and test sets data distribution is similar. The results obtained mislead the performance of the IDS as the model is bound to face different data when deployed. When the models were trained with one dataset and tested with another, the performance of the model declines because the methods used for simulation are different across both the datasets. Performing cross-dataset evaluation helps understand that the IDS generated with the same simulated data does not generalize well when tested with attacks simulated differently that is adversaries may intrude into the network using novel methods which the simulated data does not reflect.

To be able to develop a strong IDS using simulated data, it is important to train the model with different types of data and simulation, so that the IDS can generalize well when deployed. Future work includes to generate a dataset that contains similar attacks simulated using different methods to help generalize well, train a single model with different datasets to reflect all possible attack simulations. One can also perform cross-dataset evaluations with a greater number of datasets to ensure that the IDS built is strong and perform well when its deployed in production.

REFERENCES

[1] McKinsey Global Institute, "The internet of the things : Mapping the value beyond the hype," 2015.

[2] H. Liu and B. Lang, "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey," *Applied Sciences,* vol. 9, no. 20, p. 4396, October 2019.

[3] H. Alaidaros, M. Mahmuddin and A. A. mazari, "An Overview of Flow-based and Packet-based Intrusion Detection Performance in High-speed Networks," *The International Arab Conference on Information Technology,* 2011.

[4] "KDD Cup 1999 Data," The UCI KDD Archive, 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. [Accessed 2020].

[5] S. Hettich and D. Bay S, "The UCI KDD Archive," University of California, Department of Information and Computer Science, Irvine, CA, 1999.

[6] I. M. Obeidat, N. Hamadneh, M. Alkasassbeh, M. Almseidin and M. I. AlZubi, "Intensive Pre-Processing of KDD Cup 99 for Network Intrusion Classification Using Machine Learning Techniques," *Internation Journal of Interactive Mobile Technologies,* vol. 13, pp. 70-84, 2019.

[7] "NSL-KDD dataset," University of New Brunswick, [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html. [Accessed 2020].

[8] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, 2009.

[9] X. Gao, C. Shan, C. Hu, Z. Niu and Z. Liu, "An Adaptive Ensemble Machine Learning Model for Intrusion Detection," *IEEE Access,* vol. 7, pp. 82512-82521, 2019.

[10] D. H. Deshmukh, T. Ghorpade and P. Padiya, "Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset," in *International Conference on Communication, Information & Computing Technology (ICCICT)*, Mumbai,India, 2015.

[11] "Intrusion detection evaluation dataset (ICSXIDS2012)," University of Brunswick, 2012. [Online]. Available: https://www.unb.ca/cic/datasets/ids.html. [Accessed 2020].

[12] A. Shiravi, H. Shiravi, M. Tavallaee and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security,* vol. 31, no. 3, pp. 357-374, May 2012.

[13] M. J. Vargas-Muñoz, R. Martínez-Peláez, P. Velarde-Alvarado, E. Moreno-García, D. L. Torres-Roman and J. J. Ceballos-Mejía, "Classification of network anomalies in flow level network traffic using Bayesian networks," in *International*

Conference on Electronics, Communications and Computers (CONIELECOMP), Cholula, Mexico, 2018.

[14] S. N. Mighan and M. Kahani, "Deep Learning Based Latent Feature Extraction for Intrusion Detection," in *Iranian Conference on Electrical Engineering (ICEE)*, Mashhad, Iran, 2018.

[15] "Intrusion Detection Evaluation Dataset (CIC-IDS2017)," University of New Brunswick , Canada, [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html. [Accessed 2020].

[16] I. Sharafaldin, A. Lashkari and A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *International Conference on Information Systems Security and Privacy (ICISSP 2018)*, Portugal, 2018.

[17] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun and A. A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features," in *In the proceeding of the 3rd International Conference on Information System Security and Privacy*, Portugal, 2017.

[18] G. D. Gil, A. H. Lashkari, M. Mamun and G. A. Ali, "Characterization of Encrypted and VPN Traffic Using Time-Related Features," in *In Proceedings of the*

2nd International Conference on Information Systems Security and Privacy(ICISSP 2016), Rome, 2016.

[19] S. Ustebay, Z. Turgut and M. A. Aydin, "Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier," in *International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, Ankara, 2018.

[20] A. Yulianto, P. Sukarno and N. A. Suwastika, "Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset," in *Journal of Physics Conf. Ser. 1192 012018*, 2019.

[21] N. Moustafa and J. Slay, "The UNSW-NB15 Dataset Description," 2015. [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/. [Accessed 2020].

[22] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, 2015.

[23] M. Belouch, S. E. Hadaj and M. Idhammad, "A Two-Stage Classifier Approach using RepTree Algorithm for Network Intrusion Detection," in *(IJACSA) International Journal of Advanced Computer Science and Applications*, 2017.

[24] L. Zhiqiang, G. Mohi-Ud-Din, L. Bing, L. Jianchao, Z. Ye and L. Zhijun, "Modeling Network Intrusion Detection System Using Feed-Forward Neural Network Using UNSW-NB15 Dataset," in *IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE)*, Oshawa , Canada, 2019.

[25] N. Koroniotis, N. Moustafa, E. Sitnikova and B. Turnbull, "The BoT-IoT Dataset," [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php. [Accessed 2020].

[26] N. Koroniotis, N. Moustafa, E. Sitnikova and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," in *Future Generation Computer Systems*, 2019.

[27] M. Shafiq, Z. Tian, Y. Sun, X. Du and M. Guizani, "Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city," *Future Generation Computer Systems,* vol. 107, pp. 443-442, 2020.

[28] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park and H. K. Kim, "IoT Network Intrusion Dataset," IEEE DataPort, 2019. [Online]. Available: https://ieee-dataport.org/open-access/iot-network-intrusion-dataset. [Accessed 2020].

[29] Z. Liu, N. Thapa, A. Shaver, K. Roy, X. Yuan and S. Khorsandroo, "Anomaly Detection on IoT Network Intrusion Using Machine Learning," *International*

Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD), 2020.

[30] I. Ullah and Q. H. Mahmoud, "A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks," *In: Goutte C., Zhu X. (eds) Advances in Artificial Intelligence. Canadian AI 2020. Lecture Notes in Computer Science,* vol. 12109, pp. 508-520, 2020.

[31] I. Ullah and Q. H. Mahmoud, "IoT Intrusion Dataset 2020," [Online]. Available: https://sites.google.com/view/iot-network-intrusion-dataset/home. [Accessed 2020].

[32] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood and A. Anwar, "TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems," *IEEE Access,* vol. 8, pp. 165130 - 165150, 2020.

[33] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood and A. Anwar, "TON_IOT DATASETS," IEEE DataPort, 2020. [Online]. Available: https://ieee-dataport.org/documents/toniot-datasets. [Accessed 2020].

[34] M. Lorbach, E. I.Kyriakou, R. Poppe, E. A. Dam, L. P.J.J.Noldus and R. C.Veltkamp, "Learning to recognize rat social behavior: Novel dataset and cross-dataset application," *Journal of Neuroscience Methods,* vol. 300, pp. 166-172, 2018.

[35] X. Qin, Y. Chen, J. Wang and C. Yu, "Cross-Dataset Activity Recognition via Adaptive Spatial-Temporal Transfer Learning," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies,* vol. 3, no. 4, 2019.

[36] F. Sha, H. Hu and W.-L. Chao, "Cross-Dataset Adaptation for Visual Question Answering," *IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 5716-5725, 2018.

[37] Z. Mohammad, T. A. Qattam and K. Saleh, "Security Weaknesses and Attacks on the Internet of Things Applications," *IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT),* 2019.

[38] A. H. Lashkari, G. Draper-Gil and M. S. I. Mamun, *CICFlowMeter.*

[39] OpenJS Foundation , "Node-Red," [Online]. Available: https://nodered.org/. [Accessed 2020].

[40] S. P, "OSTINATO," [Online]. Available: https://ostinato.org/. [Accessed 2020].

[41] D. W. Hosmer and S. Lemeshow, Applied logistic Regression, Hoboken,NJ: John Wiley & Sons, 2004.

[42] T. M. Mitchell, Machine Learning, McGraw Hill , 1997.

[43] E. Frank, M. A. Hall and I. H. Witten, Data Mining: Practical Machine Learning Tools and Techniques, Fourth ed., Morgan Kaufmann, 2016.

[44] RapidMiner, *RapidMiner Studio Software.*

[45] J. Browlee, "How to Scale Data With Outliers for Machine Learning," Machine Learning Mystery, 2020 May 27. [Online]. Available: https://machinelearningmastery.com/robust-scaler-transforms-for-machine-learning/. [Accessed 2020].

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and Duchesnay, "Scikit-learn: Machine Learning in {P}ython," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[47] T. p. d. team, *pandas-dev/pandas: Pandas,* Zenodo, 2020.