

An Ensemble Machine Learning Approach For Network Intrusion
Detection Using Packet Flow Statistics

MAJOR PROJECT REPORT

*Submitted in partial fulfillment of the requirements for the
award of the degree*

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

N.Ramanujam
01751202717

Paras Prakash
02051202717

Shubham Dhingra
20251202717

Lakshay Arora
41551202717

Guided by

Mr Vishal Sharma

Assistant Prof



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
BHARATI VIDYAPEETH'S COLLEGE OF ENGINEERING
(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)
DELHI – 110063
JUNE 2021

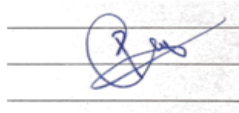
CANDIDATE'S DECLARATION

It is hereby certified that the work which is being presented in the B. Tech Minor Project Report entitled "**An Ensemble Machine Learning Approach For Network Intrusion Detection Using Packet Flow Statistics**" in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the **Department of Computer Science & Engineering of BHARATI VIDYAPEETH'S COLLEGE OF ENGINEERING, New Delhi (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)** is an authentic record of our own work carried out during a period from **February 2021 to June 2021** under the guidance of **Mr Vishal Sharma, Assistant Professor**.

The matter presented in the B. Tech Minor Project Report has not been submitted by us for the award of any other degree of this or any other Institute.



N.Ramanujam
01751202717



Paras Prakash
02051202717



Shubham Dhingra
20251202717



Lakshay Arora
41551202717

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. He/She/They are permitted to appear in the External Major Project Examination

Mr. Vishal Sharma
Assistant Professor

Dr. Pranav Dass
Head, CSE

The B. Tech Minor Project Viva-Voce Examination of **N.Ramanujam(01751202717)**, **Paras Prakash(02051202717)**, **Shubham Dhingra(20251202717)** and **Lakshay Arora(41551202717)** has been held on **09/01/2021**.

Project Coordinator

Project Coordinator

(Signature of External Examiner)

ABSTRACT

Security and privacy threats are an ever-rising trend nowadays. Malicious Hackers are trying to infiltrate into toughest of defences by using advanced threat vectors. In a hostile environment, an IDS can be the best detection and prevention option. The dataset which we used, is a combination of CIC-IDS-2017 and CIC-IDS-2018. We performed both multiclass and binary classification to identify whether there is a malicious attempt or attack, and if there is then to identify the type of attack. Statistical features are generated from the dataset and their correlation is measured after extensive visualization. The implemented model uses a stacking ensemble machine learning model that dwells and develops its learning process on the results provided by the previous algorithm. The algorithms used in stacking are Logistic Regression, Decision Tree, Random Forest, Support Vector Machine and the XGBoost. This tool can work as a standalone protection mechanism as a fully comprehensive IDS system. The implemented model achieved an accuracy of 98.1% for multi-class classification and 98.2% for binary classification with very few false positive instances.

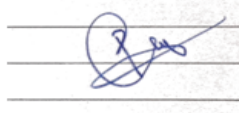
ACKNOWLEDGEMENT

We express our deep gratitude to **Mr Vishal Sharma**, Assistant Professor, Department of Computer Science & Engineering for his valuable guidance and suggestion throughout my project work. We are thankful to **Mr Vishal Sharma**, Project Coordinators, for their valuable guidance.

We would like to extend my sincere thanks to the **Head of the Department, Dr Pranav Dass** for his time to time suggestions to complete our project work. We are also thankful to **Dr Dharmender Saini, Principal** for providing us with the facilities to carry out our project work.



N.Ramanujam
01751202717



Paras Prakash
02051202717



Shubham Dhingra
20251202717



Lakshay Arora
41551202717

TABLE OF CONTENTS

CANDIDATE DECLARATION	2
ABSTRACT	3
ACKNOWLEDGEMENT	4
TABLE OF CONTENTS	5 – 6
LIST OF FIGURES	7
LIST OF TABLES	8
LIST OF ABBREVIATIONS	9
 Chapter 1: Introduction	 10-14
1.1 The Problem	10
1.2 Network Intrusion Detection System	10
1.3 Objectives	12
1.4 HIDS vs NIDS	12
1.5 Cyber Attacks	13
 Chapter 2: Related Work	 15-21
 Chapter 3: Methodology	 22-29
3.1 Dataset Description	22
3.2 Specifics of the data	24
3.3 Architectural Framework	25
3.4 Preprocessing	26
3.5 General Approach and Algorithms used	26
3.6 Logistic Regression Classifier	27
3.7 Support Vector Machine	27
3.8 Decision Tree	27
3.9 Random Forest	28

	3.10	XGBoost Classifier	28
	3.11	Ensemble Stacking	28
Chapter 4:	Results and Analysis		30-38
	4.1	Overview of results	30
	4.2	Analyzing the ROC curves	32
	4.3	Analyzing individual models	33
	4.4	Analyzing feature importances	36
	4.5	Results of different sampling techniques	38
Chapter 5:	Conclusion and Future Scope		39
Chapter 6:	References		40-45

LIST OF FIGURES

Figure 1	Comparison of HIDS and NIDS
Figure 3.1	Distribution of different type of attacks
Figure 3.2	Distribution of binary classes (malicious attack or general packet)
Figure 3.3	Forward Feature selection
Figure 3.4	Multilayer perceptron
Figure 3.5	Ensemble Learning
Figure 4.1	Receiver Operating Characteristics curves for different models.
Figure 4.2	Confusion Matrix for XGBoost Classifier (Binary)
Figure 4.3	Confusion Matrix for all Level-0 Classifiers (Binary classification).
Figure 4.4	Confusion Matrix for XGBoost Classifier (Multi-class)
Figure 4.5	Feature Importances
Figure 4.6	Distribution plot of Packet Length Mean
Figure 4.7	Distribution plot of Total Forward Packets
Figure 4.8	Scatterplot of Packet Length Standard Deviation

LIST OF TABLES

Table 2.1	Related Works Comparison
Table 3.1	A sample of Packet Flow Features used
Table 3.2	List of executed attacks and duration
Table 4.1	Classification metrics (binary)
Table 4.2	Classification metrics (multi-class)
Table 4.3	Comparison of Sampling Techniques

LIST OF ABBREVIATIONS

NIDS	Network Intrusion Detection System
IT	Information Technology
ML	Machine Learning
DDOS	Distributed Denial of Service
QoS	Quality of Service
ACCS	Cyber Range Lab of the Australian Centre for Cyber Security
CSV	Comma Separated Values
UNSW	University of New South Wales
ROC	Receiver Operating Characteristics
DNS	Domain Name System
FTP	File Transfer Protocol
TELNET	Teletype network
P2P	Peer to Peer
WWW	World Wide Web
IM	Instant Messaging
DARPA	Defense Advanced Research Projects Agency
SIEM	Security Information And Event Management

CHAPTER 1: INTRODUCTION

1.1 The Problem

Security failures and intrusion has been a major problem since the commercialization of the internet. It's a more valid issue now more than ever due to the ever-increasing number of devices being linked to networks. While this has indeed led to big possibilities for online storage and others, it also has led to huge security risks which need to be sufficiently countered. Network Intrusion Detection Systems have been around for decades now but traditional detection was more reliant on static analysis of the packets and extraction of simple attributes of data packets like header length, body size, flags used, etc, have been replaced by more robust and efficient IDS systems that have the capability of deep packet inspection and uses Machine Learning algorithms to find correlations and extrapolate new attributes and relations out of the given data to predict the result. But most of these solutions suffer from high false positivity rate problems or extensive consumption of computational power problems.

Our approach tries to redress this weakness by using an ensemble of machine learning algorithms that can learn the nature of attacks from the data flow statistics of some channel and gauge whether it is malicious in nature or not. The implemented approach has a very low false positivity rate and a very high true positivity rate.

1.2 Network Intrusion Detection Software

An Intrusion Detection System (IDS) may be a system that monitors network traffic for suspicious activity and issues alerts when such activity is discovered. it's a software application that scans a network or a system for a harmful activity or policy breaching. Any malicious venture or violation is generally reported either to an administrator or collected centrally employing a security information and event management (SIEM) system. A SIEM system integrates outputs from multiple sources and uses alarm filtering techniques to differentiate malicious activity from false alarms.

The primary advantage of an intrusion detection system is to make sure IT personnel are notified when an attack or network intrusion could be happening. A network intrusion detection system (NIDS)

monitors both inbound and outbound traffic on the network, also as data traversing between systems within the network. The network IDS monitors network traffic and triggers alerts when suspicious activity or known threats are detected, so IT personnel can examine more closely and take the acceptable steps to dam or stop an attack.

The earliest generations of IDS were completely based on a rule-set and needed frequent updates. These rulesets were highly specific to malware families like Gozi, Phynx, Robbox, etc and needed to be completely modified to detect some different malware families. Furthermore, it was really difficult for network administrators to maintain a log-chart of all these rule-sets and mostly these rules failed on zero-day attacks. Also, even minor changes in malware behaviour like polymorphic malware(which automatically changes its codes and stub sequences) bypassed these statistics based IDS. These IDS couldn't do packet inspection on the network flow, and also failed to decode encrypted traffic. The next generation of IDS was based on heuristics and used more sophisticated analysis chains to detect malicious traffic. Shiravi in [1] used a heuristic chain to make visualization charts for the network administrators, which could help them to save their time and energy wasted on scrolling through long logs stored by the IDS. Their proposed solution could also correlate the irregular behaviour logs to multiple IP's and thus could even detect the Distributed Denial of Service(DDOS) attacks. The output graphs generated by their tool could easily amalgamate gigabytes of log data onto a page and reveal the most important information out of it. Similar attempts were made by Babu in [2], wherein a heuristics-based IDS was made for the Internet of Things(IoT) devices, that could couple with most of the highly variant IoT hardware and could still detect IoT based malware like Marai malware. Also, the proposed solution used very little computational power and thus could be properly integrated with any low-powered IoT device. The heuristic rules of the proposed solution were based on network flow dynamics and statistics. The latest addition to the legacy of IDS has been Artificial Intelligence oriented solutions that do learn from past experiences and could handle even the zero-day threats. These solutions can be made through supervised or unsupervised algorithms. Mahdi[3] and Srinivas[4] were amongst the pioneers who used modern and modular machine learning techniques like Support Vector Machine for solving the zero-day threat problem.

1.3 Objectives:

- 1) A single standalone tool framework that can detect as well as classify network attacks with high accuracy and low false positivity rate.
- 2) Use novel training and sampling techniques that formulates to remove the shortcomings of the previously used machine learning approaches.
- 3) Combining 2 similar datasets(CICIDS2017 and CICIDS2018 dataset): By using several balancing and sampling techniques, finding an optimum way to combine two similar datasets.
- 4) Develop a machine learning architecture that fits perfectly on the data and provides robust security even against zero-day attacks.
- 5) Develop a proof of concept of a live performance of IDS, while checking its compatibility with modern systems.
- 6) Test several sampling techniques to merge and preprocess the dataset

1.4 HIDS vs NIDS

There are primarily two configurations of using an IDS --- NIDS(Network Intrusion Detection System) and HIDS(Host Intrusion Detection System). NIDS provide holistic protection to a subnetwork whereas the HIDS provide on-site protection to a machine. NIDS are mostly connected through taps, which duplicates all the ingress traffic to a subnet. HIDS provide better protection to an individual machine, throwing very few false positives and consuming almost negligible bandwidth.

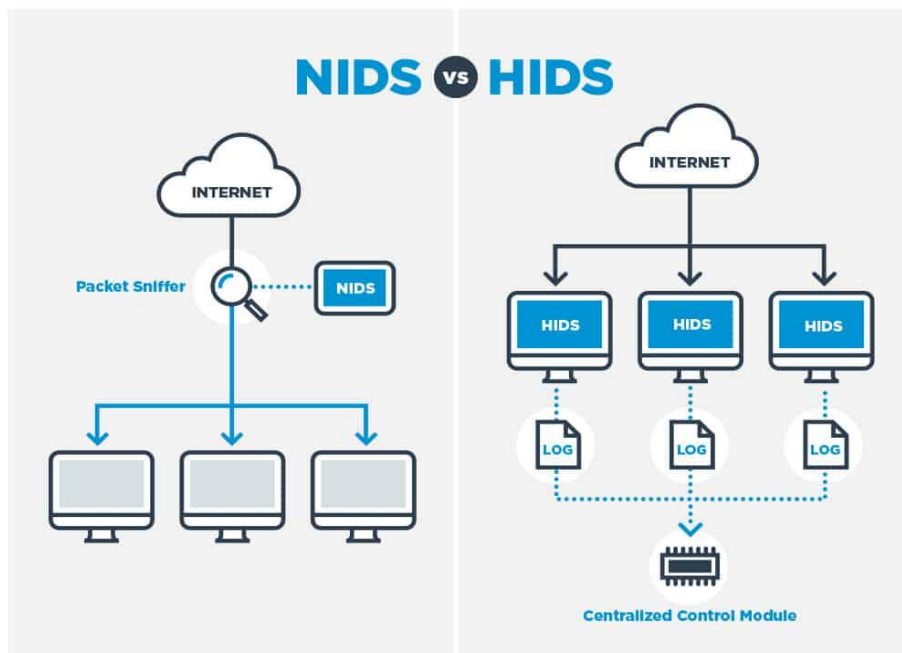


Figure 1: Comparison of HIDS and NIDS

As shown in figure 1, the NIDS system requires its own infrastructure, bandwidth and mostly sits at the egress point of the network. On the contrary, HIDS are mostly installed on the system and utilizes the system's resources like RAM, Disc Space, etc to function properly. HIDS generate separate logs for each system and mostly require a centralized module to compile all the systems log for documentation and compliance of the security policies.

1.5 Cyber Attacks

Hackers use tons of techniques, tactics and procedures(TTP) to compromise a machine or a network. There are several steps like infiltration, lateral movement that a malicious person needs to do before executing the malware. It can be as simple as causing a denial of service or a trickier web-based attack like XXE injection, XSS attack, etc. Some of the attacks are as follows:

- 1) DOS: Denial of Service situation arises when malware is able to temporarily disable any service on a platform. It may be done via flooding of an open port, using vulnerability of a service running on a machine or flooding the network devices RAM like MAC flooding attacks.
- 2) DDOS: Distributed Denial of service is an advanced form of DOS attacks, wherein several(even

in millions) of machines try to connect to the services of the target machine, thus flooding all its computational resources and bringing down the machine to its knees. This type of attack is difficult to defeat as IP banning does not work in this case.

- 3) Infiltration: Infiltration is the second step of any successful hacking attempt, wherein a bad guy tries to attain a shell or command prompt on a machine. The hacker may or may not run any malicious file in this step but this step ensures that internal machines become in-reach to the bad guy.
- 4) Botnet: Botnet is a zombie host that remains infected for a long period of time until its master bot(controller bot) issues a command to it. These botnets can be used to conduct other types of attacks like DDOS or can be used as front-end in any legal breach attempt. Botnets can also be anytime converted into trojan infected systems and thus can leak credentials, information about the user of that machine,
- 5) PortScan: Port Scan is the basic and most primary step in any penetration testing. In PortScan, the malicious hacker tries to send different types of normal as well as deformed packets to the vulnerable target, so that the target leaks out some information about its configurations, services or ports. Port Scanning is illegal in the USA, but it is within legal boundaries in countries like India.

CHAPTER 2: RELATED WORK

The first generation of Intrusion Detection systems was purely rule and signature-based. They had a very resource-intensive processor and analyzer unit which scanned all network traffic by matching all the predefined ruleset with the network statistics. This type of system required frequent updates and was prone to new zero-day attacks. As an earlier NIDS example, Crotti[5] proposed that apart from network statistics, packet inspection at the IP level can also reveal some useful artefacts that can help in the segregation of traffic. But these attempts failed miserably in front of newer attacks like DNS tunnelling [6], wherein forged DNS queries are transferred to the destination server which on decryption, results in the execution of HTTP protocol. To counter these problems, newer and more robust rules had to be created like [7], which can inspect deep into the network packets and calculate indirect statistical features like inter-arrival time, etc. Tian in [8], proposed a novel model named smooth K-Windows architecture which recorded a user's normal behaviour over a period of time and then compared it to future situations. Muhammed in [9] compared all statistical approaches that were used in NIDS and gave out conclusions that signature-based NIDS are the most accurate defence system but this less false-positive rate comes at the cost of high resource consumption and higher operational costs. The software used in this research was Weka, which was used to compile all research data and find the resultant outcomes.

Traffic flow statistics is not a new term in packet inspection, the earliest attempts to implement flow statistics from sampled data can be seen in [10], wherein the problems related to traffic generation and monitoring, the effect of long traffic sequences of traffic flow were discussed. In the end, a model was proposed that could successfully determine the number of active connections and the export rate of the communication environment. This proposed model was claimed to be victorious against several network intrusion attacks like Denial of Service attacks.

There are several ways in which network traffic can be intercepted and parsed to decrease the manual work done by the investigator who is trying to find an ongoing or past attack. In [11], an apparatus was proposed which could intercept and do all sorts of packet traffic statistics generation. The result of this apparatus would be a NetFlow export diagram which could be easily fed into any traffic analyzing tool. This model could be used as a module of Network Intrusion Detection System (NIDS), wherein the

NIDS workload would be significantly reduced as the NIDS system doesn't have to deal with complicated data streams and this work could be easily done by this proposed model.

Amongst several other benefits of understanding flow traffic, prediction of future traffic flow was also used to mitigate or stop a Distributed Denial of Service (DDOS) attack even before it reaches its peak. In [12], a methodology was put forth, wherein a networking device was used to capture traffic flow for a significant period of time. These traffic patterns were grouped together and were sent to rigorous preprocessing. This processed traffic flow was used as final flow statistics to predict the initiation of a DDOS attack. If this methodology raises a red flag, then the rate limit of traffic was checked and according to that the traffic data was either sent to a policy server or the traffic mitigation protocols. Abhishek in [13] used basic euclidean distance-based algorithms like the K-means algorithm to segregate pristine network behaviour patterns to the abnormal patterns.

In [14], a method to defeat DDOS attacks was proposed, which could automatically find the malicious packets and then could on-the-flow discard these packets. The method implemented a score based selection of packets, wherein every packet was assigned a score based on the statistical flow attributes and finally, this score would be used to make classification between pristine and malevolent packets. The method used software pipelining and optimization, which would make this project ready to be implemented.

Most of the research done on NIDS using Machine Learning has been performed using the KDD99 dataset [15]. This dataset is more than 20 years old and has been widely used in academic research. However, with the advent of IoT (Internet of Things) and a substantial change in network architecture and design over the years and the huge increment in the number of users, it is required that newer network attacks and their patterns must also be taken into consideration. Due to this reason, we used the UNSW-NB15[16] dataset. A thorough statistical comparison and analysis were performed between these two datasets [17] which demonstrated the complexity of the UNSW-NB15 data set in different characteristics. They performed the analysis of correlations of the different features with each other and also the target variable. Another statistical analysis was done for explaining the observations and attributes. Lastly, they evaluated the complexity and accuracy in terms of FARs or false alarm rates, comparing them to the legacy KDD99 data set. A conclusion was drawn that UNSW-NB15 is more complex in nature and can be used as a modern benchmark dataset.

Janarthanan and Zargari [18] used a number of algorithms for the purpose of feature selection on the UNSW-NB15 dataset. They used the Weka tool and used the Kappa Statistic measure for evaluating the effectiveness of different subsets of features. They proposed that the Random Forest Classifier outperformed other classifiers after feature selection. However, the forward feature selection method was not employed here. We use this method of feature selection for iteratively selecting the best feature which provides an increase in the performance of our model.

The authors in [19] provided an approach primarily based on filters. It was used for DDoS or Distributed Denial of Service type attacks. Various traditional statistical methods such as information gain, Chi-Square and Gain Ratio were used. They used the KDD99 attack dataset. A decision tree algorithm was trained for classification. However, the research did not explore the aspects of multiclass classification problems for NIDS.

Primartha et al.[20] performed anomaly detection for networks using Random Forest Classifier. They used ensembles and chose the base models along with grid search for hyperparameter tuning. The authors claim that a random forest model performs better than ensembles with other single classifiers such as Naive Bayes and neural networks. However, they have not used more powerful ensembles or combined several weak learners into a strong learner. An ensemble of combining a single strong learner with a weak learner is less effective than combining several weak learners [21].

Kasongo et al. [22] used feature selection for reducing the model complexity using different methods. They proposed that a reduced feature vector using the XGBoost feature extraction algorithm performed best in terms of increasing the detection accuracy on the test data as well as decreasing the model complexity and hence training time.

Husain et al. [23] performed Extreme Gradient Boosting as a classifier along with feature selection. It was reported to outperform other classifiers. An ensemble approach of combining different base learners and the XGBoost model was not taken here. Our experimental results show that it outperforms the traditional approach.

HK Lim et al. (2019) trained around five deep learning models [24] using the convolutional neural network (CNN) along with residual network (ResNet) to perform the task of network traffic classification. They developed a data preprocessing method to create a dataset using only the payload of the packet. This approach claims to generalize well to unseen packets by excluding header information which has a typical structure and inconsistent values.

Raikar et al. (2020) used integration of [25] Software Defined Networks (SDN) architecture along with machine learning. They used different algorithms such as SVM, Naive Bayes and nearest centroid for classification. They captured the network flows and used the machine learning algorithms for prediction. The Naive Bayes algorithm was reported to achieve better accuracy than others.

C Yu et al. (2018) proposed an architecture that combined deep packet detection and semi-supervised machine learning of multi-classifier in SDN. They reported that the architecture could classify flows into different QoS categories. Based on the above classification the network can achieve [26] fine-grained adaptive QoS traffic engineering and maintain a dynamic flow database.

Erman et al. (2007) argued that byte accuracy [27] is an important measure for a classifier's performance. Byte accuracy better accounts for the class imbalances amongst applications and is a better indicator of the performance the classifier would obtain for typical traffic classification tasks such as traffic shaping, and traffic analysis. They proposed that in most of the real-time traffic usages, byte accuracy is very important to the classifier. According to them, just using flow accuracy as a metric might cause a heavy bias in classification results.

Muhammad Shafiq et al.(2017)[28] in their paper attempt two data sets, HIT and NIMS data for classifying network traffic. They captured online internet traffic of seven different kinds of applications such as DNS, FTP, TELNET, P2P, WWW, IM and MAIL to make data sets. Then, they extracted the features of captured packets using the NetMate tool. They applied three Machine Learning algorithms: ANN(Artificial Neural Network), C4.5 Decision Tree and SVM(Support Vector Machine) compared their results which showed that all the algorithms provided them good results but Decision Tree won out with an accuracy of 97.5%.

Mohammad Reza Parsaei et al(2017).In [29] their paper used four variants of Neural Networks to categorize traffic by application. Those were: FeedForward NN, Multilayer Perceptron (MLP), NARX (Levenberg-Marquardt) and NARX (Naïve Bayes). These algorithms each provided accuracy of 95.6%, 97%, 97% and 97.6%.

We Li et al(2007)[30] presented a machine-learning approach that accurately classifies internet traffic using the C4.5 decision tree. They did not inspect the packet payload for the classification purpose but rather collected 12 features regarding flow statistics to identify traffic of different types of applications with 99.8%.

Ferrag et al[31] in this paper a survey of deep learning approaches to IDS, a comparative study of different models on various datasets of Intrusion Detection. Seven different deep learning models were compared on CIC-IDS 2018 and Bot-Iot dataset. This paper provided the necessary analysis of different types of deep learning approaches being used for IDS and their performances and helped shape our own understanding of how to deal with the CIC-IDS dataset effectively.

A. Binbusayyis et al[32] in their paper the key impetus was to identify and benchmark the potential set of features that can characterize network traffic for intrusion detection. For this an ensemble approach was proposed along with four different feature evaluation measures, such as correlation, consistency, information, and distance, to select the more crucial features for intrusion detection. They showed that an ensemble approach can prove to be a viable technique for IDS and provided information on how to select more crucial features effectively.

B. A. Tama et al[33], this paper proposed a stacked ensemble approach for anomaly-based intrusion detection systems. Unlike a conventional stacking, where some single weak learners are prevalently used, the proposed stacked ensemble is an ensemble architecture, yet its base learners are other ensembles learners, i.e. random forest, gradient boosting machine, and XGBoost. This provided sufficient information about the performance of stacked ensembles for anomaly-based intrusion detection systems and provided the basic architecture for our work.

A. Tesfahun et al[34], their paper made use of Random Forest classifier along with SMOTE and Feature Reduction methods on an IDS. Their primary idea was to deal with the inherent class imbalance in their dataset NSL-KDD so that machine learning models can be applied effectively to produce better results. Their results showed that using Random Forest along with SMOTE and information gain based feature selection gives significantly better performance for IDS. They also showed how to tune the model to better handle the class imbalance problem.

Jinghua Yan et al(2018)[35] in their survey paper introduce SDN and further move on to discuss traffic classification and then review several representative works of traffic classification in SDN. The works were reviewed in line with the choice of classification strategies and contribution to the literature. The paper ends with a discussion on research challenges and future directions for SDN traffic classification.

In [36] their survey paper mentioned that the implementation of a dynamic approach of network analysis would require the use of Software Defined Networking(SDN) as it allows us to implement intelligence into the system which makes it useful in real-time scenarios. Not only that but SDN can provide a centralized controller, dynamic update of flow table and traffic analysis, global view of network topology and dynamic routing. Thereby various traffic classification techniques can be implemented using SDN as a basis.

[37] Mestres et al.(2017) in their paper it has mentioned the challenges faced in deploying AI-based techniques for network control and operation. They suggested that two paradigms namely (SDN) Software Defined Network and (NA) Network Analytics can potentially lead to the implementation of AI techniques and also provided a new paradigm that can potentially exploit the two and provide various use cases for their applicability and benefits. This paradigm they experimented with is called (KDN) Knowledge Defined Networking.

[38] Levin et al.(1997) this paper mentions that neural networks can be used effectively for the identification and control of nonlinear dynamical systems. The emphasis of the paper is on models for both identification and control. Static and dynamic back-propagation methods for the adjustment of parameters was also discussed. While a multi-layered and recurrent network model was used for the experimentation.

[39] Kreutz, D., et al. “Software-defined networking: A comprehensive survey.” discussed the current difficulties to manage the current IP addresses and networks and present a comprehensive survey on SDN. The main ongoing research efforts and challenges of SDN are also discussed. In particular, it explores the design of switches and control platforms – with a focus on aspects such as resiliency, scalability, performance, security and dependability.

Table 2.1: Related Works Comparison

Research Paper	Approach Used	Impact	Comparison
[40]Farhan, R., Maolood, A. and Hassan, N., 2020. Performance analysis of flow-based attacks detection on CSE-CIC-IDS2018 dataset using deep learning. Indonesian Journal of Electrical Engineering and Computer Science, 20(3), p.1413.	In this paper, an Anomaly-based Intrusion Detection System was implemented using DNN (Deep Neural Network) and their model made use of SMOTE algorithm for an imbalanced classification problem.	This paper provided us with the latest and diverse real-traffic Flow-based dataset named CIC-IDS 2018 containing varied types of attacks.	Our model used simple ML classifiers instead of DNN(Deep Neural Networks) but provided a higher accuracy of 98% as compared to 90% in this paper on CIC-IDS 2018 dataset.

<p>[41]Yulianto, A., Sukarno, P., & Suwastika, N. A. (2019, March). Improving Adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset. In Journal of Physics: Conference Series (Vol. 1192, No. 1, p. 012018). IOP Publishing.</p>	<p>This paper made an improvement in the performance of AdaBoost based IDS by making use of SMOTE and PCA and achieved 92% accuracy.</p>	<p>This paper made use of the CIC-IDS 2017 dataset which was found to be useful for our objectives and inspired us to explore SMOTE, PCA and EFS to achieve higher performance.</p>	<p>We also made use of a boosting technique called XGBoost in a stacked ensemble approach to boost our model's accuracy and achieved a higher accuracy than their model. We achieved this without using EFS and other sampling techniques like SMOTE used in this paper.</p>
<p>[42]Panigrahi, R., & Borah, S. (2018). A detailed analysis of the CICIDS2017 dataset for designing Intrusion Detection Systems. International Journal of Engineering & Technology, 7(3.24), 479-482.</p>	<p>This paper pointed out the various shortcomings faced while working on CIC_IDS 2017 dataset and provided some alternatives to resolve the class imbalance problem.</p>	<p>This paper provided us with some preprocessing measures and techniques to resolve the class imbalance problem and inherent shortcomings of the dataset. It also inspired us to consider a combined dataset for our experimentation.</p>	<p>This paper tried to divide the dataset into segments to better deal with its shortcomings whereas we followed a completely novel approach of combining both CIC-IDS 2017 and 2018 datasets and then tuning our model to achieve good results.</p>
<p>[43]Epp, N., Funk, R., Cappel, C., & Lorenzo-Paraguay, S. (2017). Anomaly-based web application firewall using HTTP-specific features and one-class SVM. In Workshop Regional de Segurança da Informação e de Sistemas Computacionais.</p>	<p>This paper used the one-class SVM technique and knowledge about the HTTP request structure to build feature extraction methods that improve the detection rates.</p>	<p>This paper inspired us to use SVM as a classifier and explore what kind of results we can obtain by adequate feature extraction methods.</p>	<p>Although their model is simpler in comparison our model does not require knowledge of HTTP request structure and still detected suspicious packets satisfactorily in comparison.</p>
<p>[44]C. Pontes, M. Souza, J. Gondim, M. Bishop and M. Marotta, "A new method for flow-based network intrusion detection using the inverse Potts model," in IEEE Transactions on Network and Service Management, doi: 10.1109/TNSM.2021.3075503.</p>	<p>This paper proposes a new algorithm, called Energy-based Flow Classifier (EFC). This anomaly-based classifier uses inverse statistics to infer a statistical model based on labelled benign examples. It showed that EFC is capable of accurately performing binary flow classification and is more adaptable to different data distributions than classical ML-based classifiers.</p>	<p>Provided us with a deep insight into simple ML classifiers and how they are not capable of domain adaptation. Also once they are trained on specific data distribution they are not general enough to be applied to other related data distributions.</p>	<p>This paper involved using an Energy-based Flow Classifier (EFC) to make the model more adaptable to various datasets which we achieved by concatenating two datasets and adequate preprocessing.</p>

CHAPTER 3: METHODOLOGY

3.1 Dataset Description

The datasets we chose for the task of training an ML model were the CSE-CIC-IDS-2018 [45] and CIC-IDS-2017 [46]. These datasets have been constructed such that they effectively capture a real-world network attack. The data creators in their paper [47] described in their seminal paper the notion of profiles to generate cybersecurity datasets. In their paper, they describe the use of M profiles in an attempt to make the attack setting simple. A straightforward way to accomplish this is by having humans interpret and act it out. In a more ideal setting, we would have an independent agent with compilers to carry them out. Some examples of the attacks carried out include Heartleech, Botnet, DDos, Dos, Infiltration. On the other hand B profile can be understood as having ML and Statistical models capture the behaviour of the malicious user and have it perform attacks.

In the end, their dataset constituted seven different attack scenarios and their attacking infrastructure was composed of 50 independent machines. The victim had 5 departments made up of 420 machines and 30 servers. The dataset has 80 features derived from CICFlowMeter [48], the record of logs and network traffic of each system. Some of the features from the dataset are depicted in Table 3.1.

Table 3.1 A small sample of Packet Flow Features used

Feature Name	Description
fl_dur	Flow duration
tot_fw_pk	Total packets in the forward direction
tot_bw_pk	Total packets in the backward direction
tot_l_fw_pkt	The total size of the packet in forward direction
fw_pkt_l_max	Maximum size of the packet in forward direction
fw_pkt_l_min	Minimum size of the packet in forward direction
fw_pkt_l_avg	The average size of the packet in forward direction
fl_byt_s	flow byte rate that is the number of packets transferred per second
fl_pkt_s	flow packets rate that is number of packets transferred per second
fl_iat_avg	The average time between two flows
pkt_len_min	Minimum length of a flow
pkt_len_max	The maximum length of a flow
pkt_len_avg	Mean length of a flow
pkt_len_std	Standard deviation length of a flow
pkt_len_va	Minimum inter-arrival time of packet
fin_cnt	Number of packets with FIN
syn_cnt	Number of packets with SYN
rst_cnt	Number of packets with RST
pst_cnt	Number of packets with PUSH
ack_cnt	Number of packets with ACK
urg_cnt	Number of packets with URG
cwe_cnt	Number of packets with CWE

3.2 Specifics of the Data

We in attempts to have more data to work with and to build a more robust model concatenated IDS 2017 with IDS 2018 to get a total of 2.8M + 8.2M samples ie. approximately an 11.1M samples. As mentioned earlier the dataset consisted of seven different attack scenarios Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside. The combined dataset had the following labels consisting of both malignant and benign samples. These are:-

DoS Hulk, PortScan, DDoS, DoS, GoldenEye, FTP-Patator, SSH-Patator, DoS slow loris, DoS Slowhttptest, Bot, Web Attack - Brute Force, Web Attack - XSS, Infiltration, Web Attack - SQL Injection and Heartbleed.

Table 3.2 List of executed attacks and duration

Attack	Tools	Duration	Attacker	Victim
Bruteforce attack	FTP – Patator SSH – Patator	One day	Kali Linux	Ubuntu 16.4 (Web Server)
DoS attack	Hulk, GoldenEye, Slowloris, Slowhttptest	One day	Kali Linux	Ubuntu 16.4 (Apache)
DoS attack	Heartleech	One day	Kali Linux	Ubuntu 12.04 (Open SSL)
Web attack	Damn Vulnerable Web App (DVWA) In-house selenium framework (XSS and Brute-force)	Two days	Kali Linux	Ubuntu 16.4 (Web Server)
Infiltration attack	First level: Dropbox download in a windows machine Second Level: Nmap and portscan	Two days	Kali linux	Windows Vista and Macintosh
Botnet attack	Ares (developed by Python): remote shell, file upload/download, capturing screenshots and keylogging	One day	Kali Linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)
DDoS+PortScan	Low Orbit Ion Canon (LOIC) for UDP, TCP, or HTTP requests	Two days	Kali Linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)

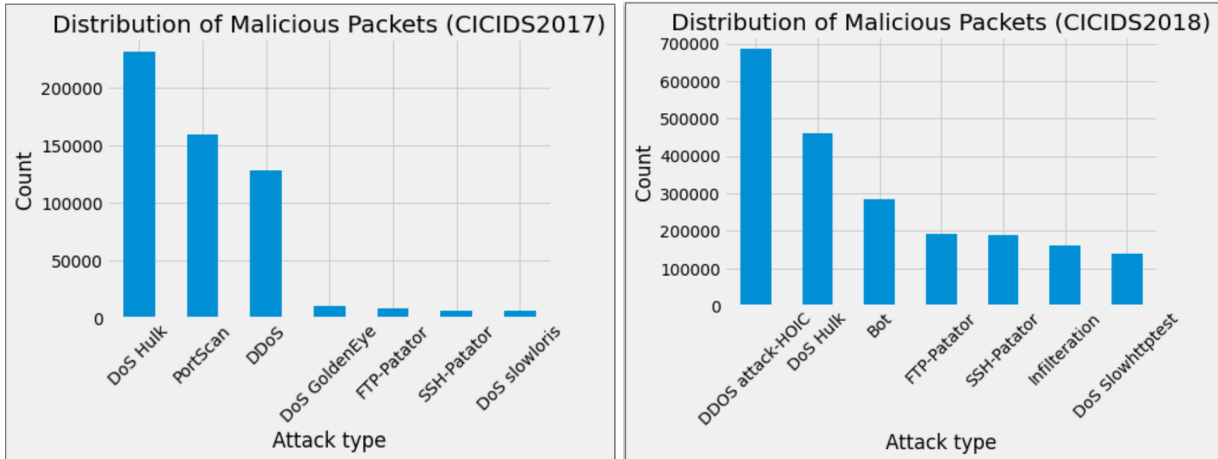


Figure 3.1 and 3.2: Distribution of different types of attacks in both datasets.

3.3 Architectural Framework

As mentioned earlier the authors of the datasets used an infrastructure consisting of 50 independent machines which operated the 7 different attack scenarios: - Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside. The victim had 5 departments made up of 420 machines and 30 servers. The attacks were separately conducted and logged. To make the data more realistic each attack's dataset consists of both benign and the attack's samples.

For capturing of packets and feature extraction CICFlowMeterV3 was used. It is a network traffic flow generator coded in Java. It provides the user with a high degree of freedom in choosing the features he/she wants to include. The flow generator renders Bidirectional Flows where the first packet is from the source to the destination (the forward flow) and the other from destination to source (the backward flow). 80 or so statistical features such as Duration, Number of packets, Number of bytes, Length of packets, etc. were calculated separately both from source to destination and vice versa.

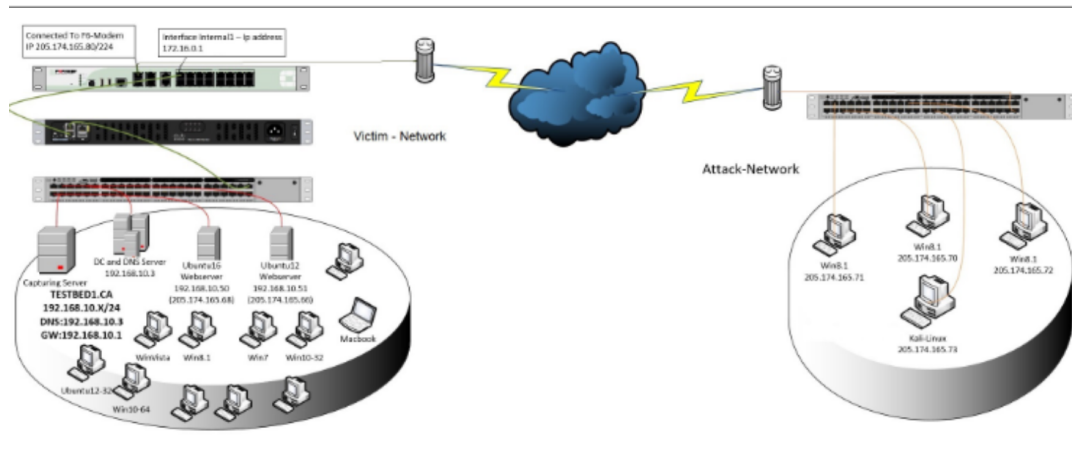


Figure 3.3: CIC-IDS Testbed Visualized.

3.4 Preprocessing

As preprocessing we removed the features which had a high correlation to an existing feature; this helps reduce the dataset's dimensionality while preserving information. We dropped all duplicate samples and removed samples of labels whose counts were too low to take into consideration. Further, we dropped samples that had Null fields. Finally, we encoded our labels to fit them in ML models. During preprocessing we also experimented with different types of Up and Down Sampling but overall they either gave us very poor performance or prediction scores no better than what we achieved without them.

3.5 General Approach and Algorithms used

The dataset provides two columns as targets one with normal-not normal values and one with attack type values. We found that classifying by attack type and aggregating the results to normal and abnormal traffic gave us better results than classifying on normal/abnormal traffic. Thus, our problem is categorized as a supervised multiclass classification problem. supported the papers we mentioned we selected the utilization of three algorithms to specialize in the issue:

- 1) Decision Tree Classifier
- 2) Random Forest Classifier
- 3) Support Vector Machine.
- 4) Logistic Regression.
- 5) XGBoost Classifier.

We will analyze the work implemented in all, rated by increasing performance.

3.5.1 Logistic Regression

Logistic Regression is a generalized linear model which uses the 'logistic' function. It performed the worst of all classifiers used by us, for both binary as well as multi-class classification. We assume the inability of the model to fit non-linearly separable data was the reason and decided to apply non-linear classifiers.

3.5.2 Support Vector Machine

Support Vector Machine maps data samples as points in space in an attempt to maximize the distances between the classification targets. New samples of data are then mapped into the same space where they get predicted to belong to one of the many different categories based on which side space they belong to. The thing that differentiates SVM is its ability to use kernel tricks to implicitly map points into higher dimensional spaces.

Linear SVM seems to mimic logistic regression's performance as they both possess similar structures. A slight increase of scores observed in logistic regression might be due to its probabilistic nature.

3.5.3 Decision Tree

A Decision Tree Classifier is a tree-based model whose internal nodes represent a label with an input feature. When a sample flows down a tree decisions are taken on each node with regards to the value that the sample possesses. The end nodes or leaf nodes represent probabilities of the sample having a particular class in the case of classification trees and continuous in the case of regression trees.

Decision Tree Classifier performed relatively well compared to Logistic Regression, which motivated us to experiment with more tree-based models. With it having relatively shorter training time to boosted algorithms it truly gave a proper insight into whether tree-based modelling was proper for this particular problem

3.5.4 Random Forest

Random Forests are based on an ensemble learning approach where multiple decision trees are constructed at training time. Essentially decision trees can be made to learn unique patterns by allowing them to grow to high depths. This makes the decision tree models overfit on the dataset which means they have high variance and low bias. Random Forest remedies the problem as it averages over these deep trees which are trained on different portions of the training set. This leads to a reduction in variance in better performance overall.

After observing a better result with the Decision Tree, we applied the Random Forest Classifier which is just an ensemble of many Decision Trees. We used 100 estimators and kept the maximum depth the same as what we used for training a single Decision Tree.

3.5.5 XGBoost Classifier

XGBoost is an open-source software library that allows the user to make use of a regularizing gradient boosting framework. Gradient Boosting trees also are a form of ensemble learning where weak learners are used for prediction. When the weak learners are Decision Trees then they are called gradient boosted trees which in most cases outperforms Random Forest.

XGBoostClassifier is an optimized distributed gradient boosting classifier designed to be highly efficient, flexible, and portable. XGBoost provides a parallel tree boosting which provides fast and accurate algorithms for predictive analysis purposes.

3.5.6 Ensemble Stacking

Stacked Generalization (also known as *Stacking*) is a type of ensemble ML algorithm. It involves combining the predictions from multiple machine learning models on the same dataset, like bagging and boosting.

Unlike bagging, the models we make use of are typically different but fit on the same dataset. Unlike boosting, a single model has been used to fit over the predictions from the base models

The architecture of a stacking model generally has two and above base models, known as level-0 models, and a meta-model that fits over the predictions of the base models, known as a level-1 model.

- **Level-0 Models (*Base-Models*):** Models fit on the train set and predictions are made into a dataset for the meta-model.
- **Level-1 Model (*Meta-Model*):** Model that fits on predictions of the base models.

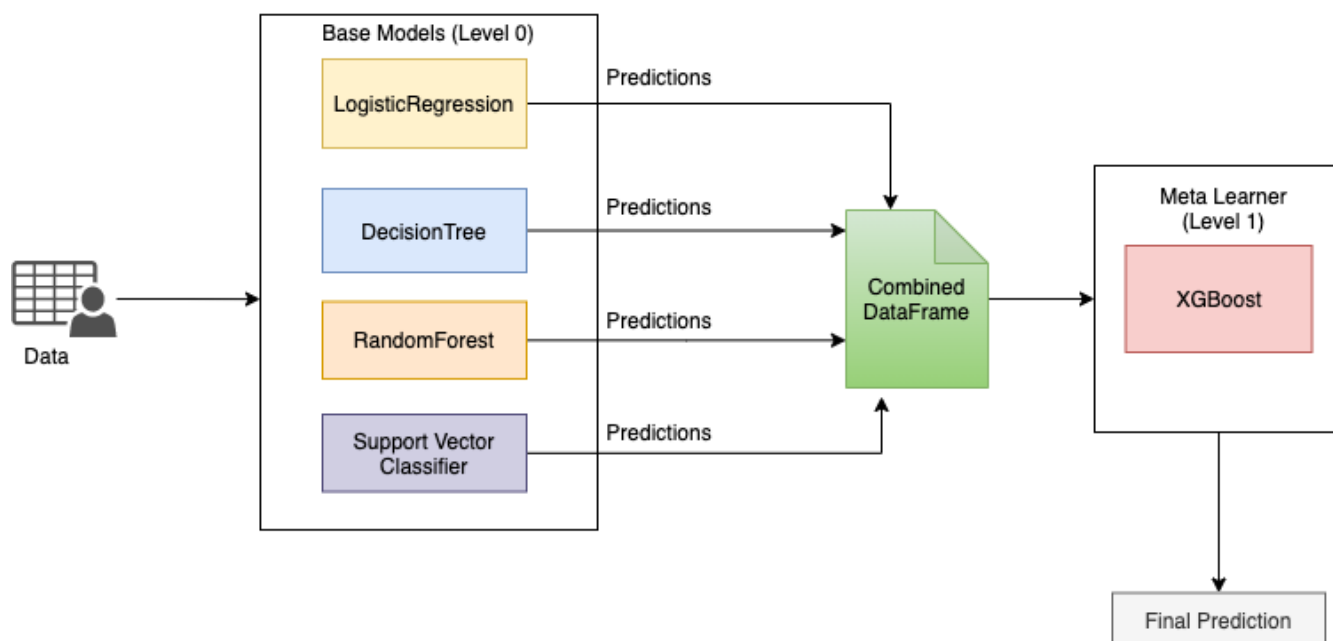


Figure 3.5: Ensemble Learning

CHAPTER 4: RESULTS AND ANALYSIS

4.1 Overview of results

Training both binary and multi-class classifiers on the training set, it could be observed that the hypothesis [49] of stacking weaker learners into a strong learner is true. Although the stacked ensemble is not vastly different from that of the strongest level 0 base learner, it still outperforms it by an appreciable margin. Considering that this stacked ensemble model did not look at the training data at all and identifies the cases in which our strongest base learner gets the predictions wrong and gives some weightage to other weaker learners to improve the strongest prediction even further.

We present the different metrics commonly used for classification. Proposing accuracy is perhaps not the best measure although not completely ignoring it, we look at ROC score [50] which signifies to some degree how confidently the model identifies what it does. It tells how much the model is capable of distinguishing between classes. We also look at the individual confusion matrices for binary classification and the final confusion matrix of the stacked ensemble for multi-class classification as well.

Table 4.1: Metrics for binary classification

Model/Metrics	Accuracy	Precision Score	Recall Score	F1 Score	ROC Score
Logistic Regression	0.909	0.919	0.979	0.948	0.741
Decision Tree	0.975	0.979	0.992	0.985	0.935
Support Vector Classifier	0.909	0.918	0.98	0.948	0.738
Random Forest Classifier	0.98	0.979	0.998	0.988	0.937
XgBoost Classifier	0.982	0.98	0.999	0.989	0.941

Table 4.2: Metrics for multi-class classification

Model/Metrics	Accuracy	Precision Score	Recall Score	F1 Score	ROC Score
Logistic Regression	0.869	0.268	0.197	0.214	0.564
Decision Tree	0.963	0.747	0.692	0.682	0.839
Support Vector Classifier	0.871	0.277	0.195	0.208	0.563
Random Forest Classifier	0.98	0.912	0.877	0.893	0.933
XgBoost Classifier	0.981	0.953	0.897	0.905	0.943

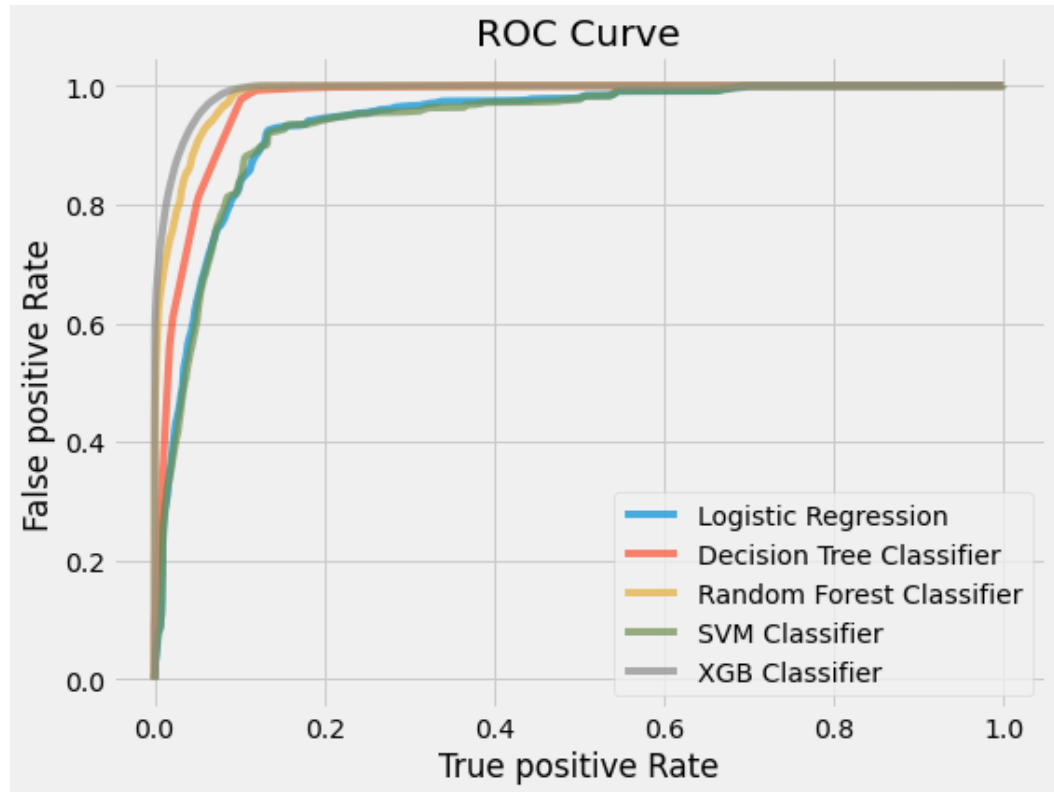


Figure 4.1: Receiver Operating Characteristics curves for all models.

4.2 Analyzing the ROC curves

The Receiver Operating Characteristics (ROC) curve shown in Figure 4.1 depicts that XGB Classifier has the best ROC-AUC score which is an improvement over Random Forest Classifier which itself (an ensemble of Decision Trees) is an improvement over Decision Tree ROC-AUC score. The linear classifiers (LogisticRegression and Support Vector Machine with a Linear kernel) have a similar unsmooth curve with an area under the curve which is although a little better than random naive classifier would, but still outperformed by a huge margin by the non-linear tree-based classifier and the final level 0 models stacked ensemble.

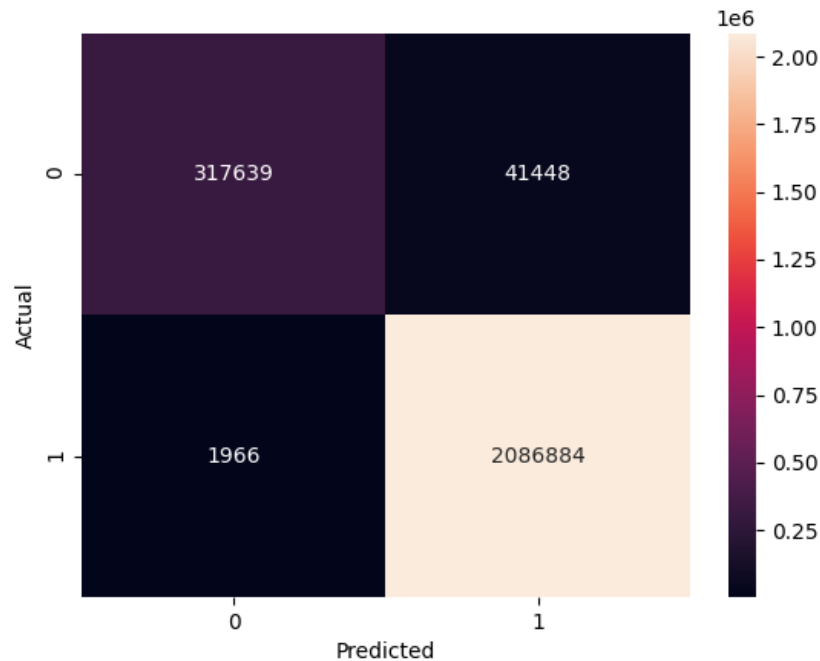


Figure 4.2: Confusion Matrix for XGBoost Classifier (Binary classification).

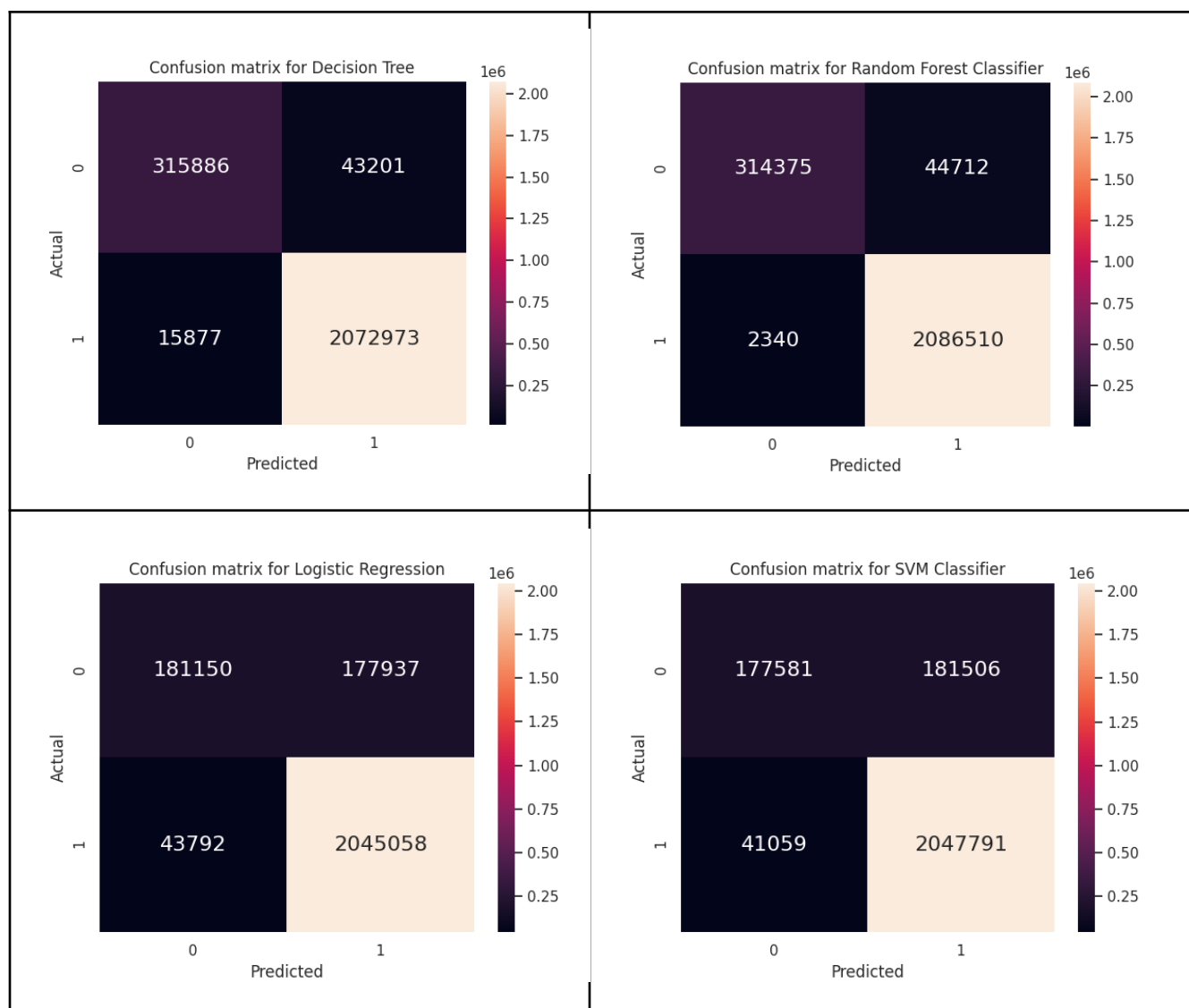


Figure 4.3: Confusion Matrix for all Level-0 Classifiers (Binary classification).

4.3 Analyzing individual models

4.3.1 Logistic Regression Classifier

As logistic regression was one of the simpler models to train, it ended training comparatively quickly and gave an accuracy of 90.9% for binary classification problems and 86.9% for multi-class classification. Out of the four classification techniques used it was one of the worst-performing in terms of all the metrics which we listed.

4.3.2 Decision Tree Classifier

As one of the most simple tree-based classifiers, we trained it for a maximum depth of 10, increasing which further we observed no improvement in the accuracy score. An accuracy of 97.5% for binary classification and ~96% for multi-class classification is observed. It outperforms the linear models but still is not fitting the data well. This model has high bias and low variance and seems to be underfitting but not as much as the linear classifiers. The difference between Decision Tree and Random Forest is not that huge in binary classification, but in multi-class classification, there is a very large difference in the performance of these two models.

4.3.3 Support Vector Classifier

The results obtained for the Support Vector Classifier did not justify the amount of time it took to train the model. Although we used a linear kernel which is the most simplistic kernel and assumes that the data can be linearly separated by a hyperplane, the results were still as bad as Logistic Regression. This model took the most amount of training time and had the worst performance.

4.3.4 Random Forest Classifier

Keeping the maximum depth of the tree the same as that used in Decision Tree and keeping several estimators, the performance of Random Forest is found to be better than Decision Tree. The gain in accuracy score is ~0.5% over Decision Tree in binary classification (98%), but a large margin of ~1.5% in the case of multi-class classification (98%). The ROC score is better as well which indicates that it is able to differentiate between the different classes better. Out of all the level 0 base learners, this is by far the best in terms of performance. It took considerably a large amount of time to train it, but less than what it took for training the SVM model.

4.3.5 XGBoost Classifier

Used XGBoost as the level 1 meta learner, which doesn't actually look at the training data but the prediction probabilities (or the predictions in case the implementation does not have a method to predict the probability such as LinearSVC) as input and the correct expected labels as output. Keeping default parameters, combining the above base learners and applying to stack using XGBoost observed the best score out of all. The performance was a little better than the best performing level 0 model but not by a huge gap. Although the model here is more confident as evident by a better F1 score and ROC score. Looking at the confusion matrix in Figure 4.4 it appears as though the only instances the model severely misidentifies are Infiltration *attacks* and *DoS Slowhttptest*.

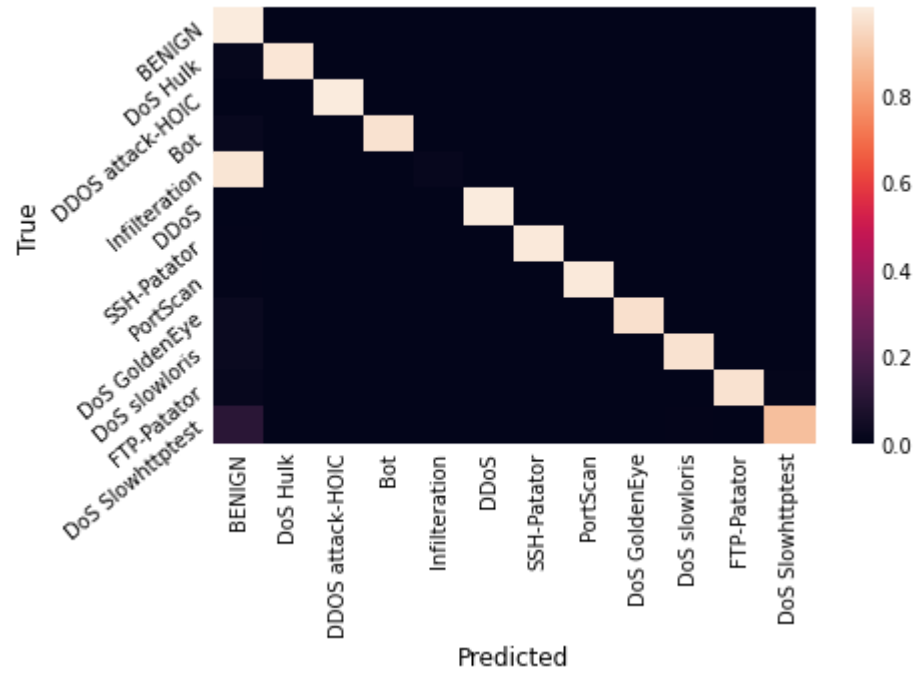


Figure 4.4: Confusion Matrix for XGBoost Classifier (Multi-class)

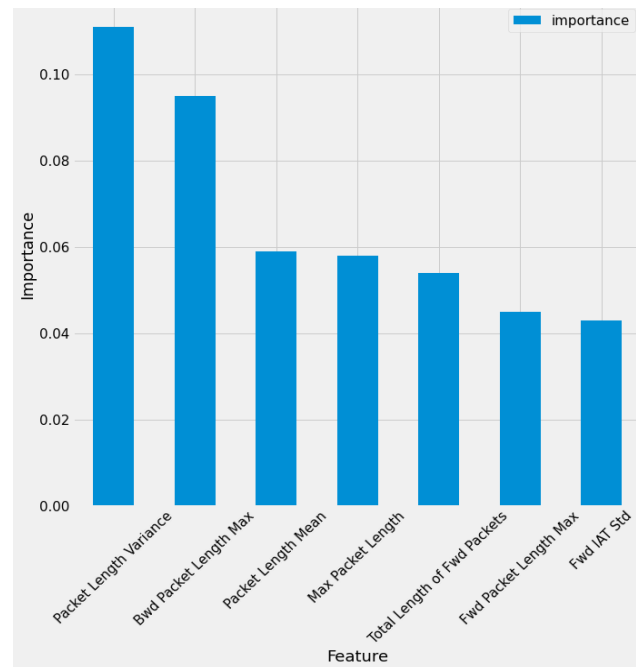


Figure 4.5: Feature Importances

4.4 Analyzing feature importances

Taking a look at feature importances from Figure 4.5 it can be observed that the most important features were statistics about the length of the packet, such as the variance, the mean length or the maximum packet length (forward or backward). In contrast, the flag-related features were the least useful according to the full results obtained. This makes sense intuitively as the different attacks could possibly be judged by the variability in these packet related features.

We go to our initial exploratory data analysis and try to decipher the results of the black box instead of taking it as a source of truth without investigation.

Looking at these features, for benign and malicious flows, we observe the following plots and make the key observations:

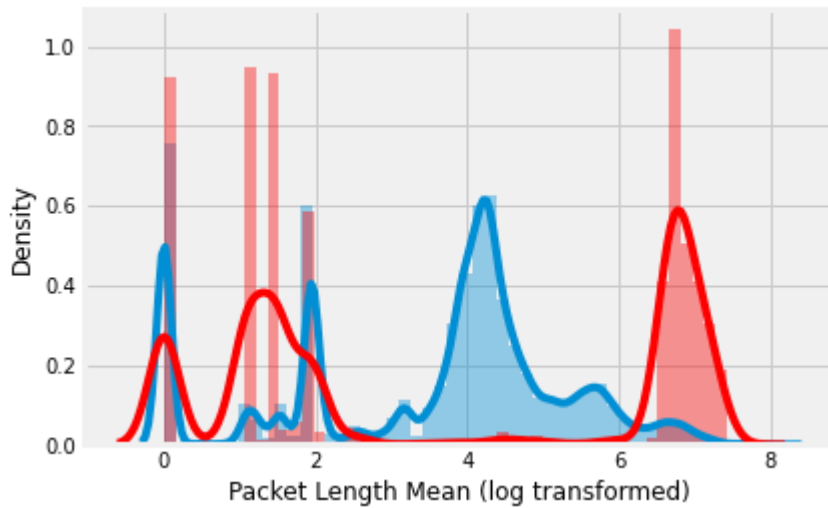


Figure 4.6: Distribution plot of Packet Length Mean

From Figure 4.6 which shows the frequency distribution of the packet length, it is clear that the benign flows have a well-defined peak that follows somewhat of a bell-shaped curve for a certain range in the interval $[2, 8]$ although a little skewed to the right. There are some peaks to the left of this curve as well, but the most interesting thing is that the attack flows have a well-defined peak that has a greater packet means and perfectly follows the bell-shaped curve for the interval $[6, 8]$. There is very little overlap for these two curves, although in the beginning they are less readily differentiable.

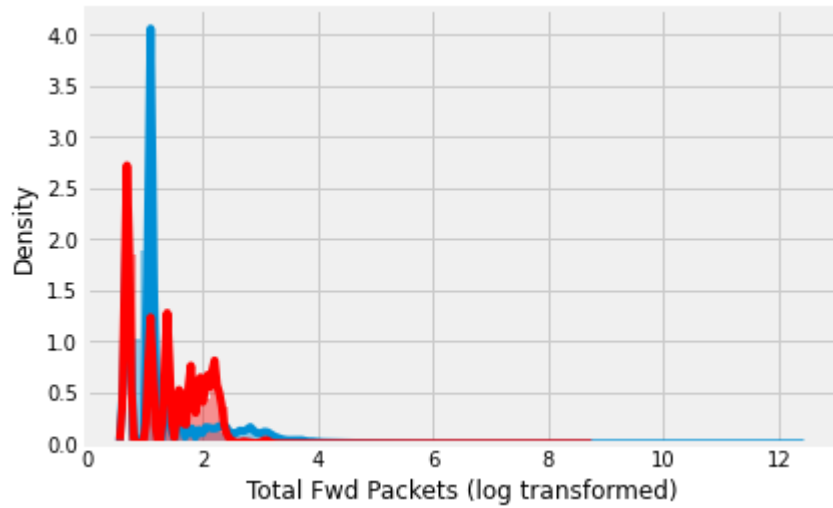


Figure 4.7: Distribution plot of Total Forward Packets

From Figure 4.7 it is observed that as the benign flows are more in frequency, the peak is higher but it is still a huge peak spread over a small interval as compared to the malicious flows where this peak is not that well defined and relatively spread over a larger interval.

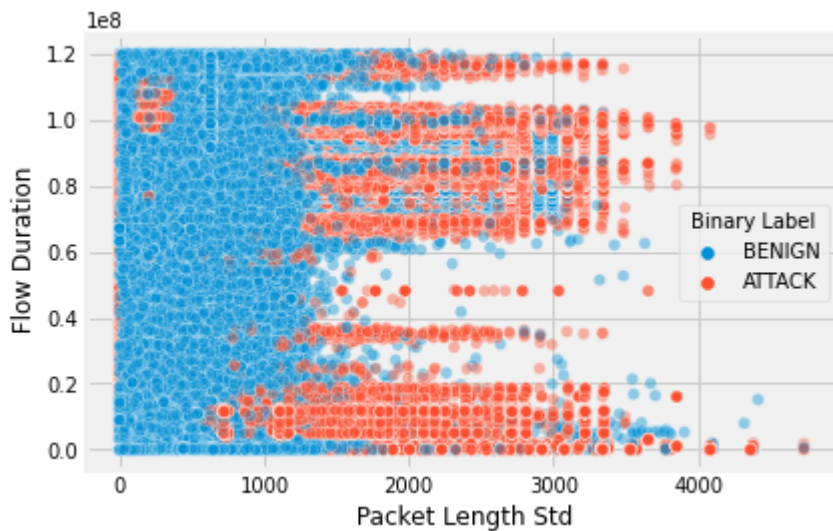


Figure 4.8: Scatterplot of Packet Length Standard Deviation

Looking at Figure 4.8, it is very clear that Attack flows have a differentially higher variance for the Packet Length feature. It can also be observed that malicious flows tend to have a slightly lower flow

duration as compared to benign flows for a small value of flow duration, but the difference is not that huge.

4.5 Results of different sampling techniques

Table 4.3 Comparison of Sampling Techniques

	Random_Over_Sampler	ADASYN	SMOTE	Near_Miss	Random_Under_Sampler	Edited_Nearest_Neighbours
Accuracy	0.968300	0.794019	0.968255	0.501347	0.969826	0.978565
Precision	0.986113	0.993346	0.985594	0.970989	0.986187	0.983466
Recall	0.976604	0.763725	0.977080	0.428426	0.978343	0.991550
F1	0.981335	0.863532	0.981318	0.594529	0.982249	0.987492
ROC_AUC	0.948299	0.866983	0.947000	0.676982	0.949315	0.947291

It is observed that sampling in most cases actually harms the performance of the classifier instead of improving it with little to no benefit of having a smaller training time. Out of the six sampling techniques we tried, only one gave better performance compared to unsampled data. The best performance was of the Edited Nearest Neighbours sampling method which we applied to the dataset before splitting and training. Sampling here was expected to be beneficial for most of these methods as they tend to work generally well when the dataset is imbalanced with the proportions it is in our case.

CHAPTER 5: CONCLUSION AND FUTURE SCOPE

The conclusion derived from the results obtained is that an ensemble learning-based machine learning approach performs better than the individual machine learning models. Sampling techniques can prove to be useful when the data has a severe imbalance, but are not guaranteed to increase the performance in terms of training time or metrics. Tree-based non-linear methods perform much better than simple linear methods. The model is very accurately able to identify the type of attack and has no difficulty differentiating between the various attacks except *Infiltration*. This proves that a signature-based intrusion detection system can be also accompanied by a machine learning-based intrusion detection system for better robustness. A machine learning-based system can also be kept up-to-date more easily with regular data feed to identify newer types of attacks.

The learning module of our model can be expanded to incorporate more types of Intrusion attacks and system vulnerabilities.

Increasing the number of base models for meta-learning can result in better results by combining more weak learners into a strong learner. Other methods for feature selection can be explored. One can perform more rigorous hyper-parameter tuning on base models used in meta-learning. Increasing the size of the data corpus and augmenting the data-set with more real-world data for a generalized model. Better handling of imbalanced data using different sampling techniques can be performed.

Our model primarily works on flows generated by data packets so it can be integrated along with a classical signature-based intrusion detection software, deployed together into the real world. This can enable real-time data collection along with automatic online learning, thus making the model much more dynamic in nature. Combining signature-based intrusion detection with machine learning-based intrusion detection will also allow better processing and analysis of the results. This not only allows the possibility of detecting attacks in real-time accurately but also to thwart such attempts at an early stage, thus maintaining system integrity.

REFERENCES

- [1] Shiravi, H., Shiravi, A., & Ghorbani, A. A. (2010, December). IDS alert visualization and monitoring through heuristic host selection. In *International Conference on Information and Communications Security* (pp. 445-458). Springer, Berlin, Heidelberg.
- [2] Babu, M.J. and Reddy, A.R., 2020. SH-IDS: Specification Heuristics Based Intrusion Detection System for IoT Networks. *Wireless Personal Communications*, pp.1-23.
- [3] Mahdi Zamani, Mahnush Movahedi *Machine Learning Techniques for Intrusion Detection*. University of New Mexico, 2013.
- [4] Srinivas Mukkamala, Guadalupe Janoski, Andrew Sung. *Intrusion Detection: Support Vector Machines and Neural Networks*. New Mexico Institute of Mining and Technology, 2002.
- [5] Crotti, M., Gringoli, F., Pelosato, P., & Salgarelli, L. (2006, June). A statistical approach to IP-level classification of network traffic. In *2006 IEEE International Conference on Communications* (Vol. 1, pp. 170-176). IEEE.
- [6] Raman, D., De Sutter, B., Coppens, B., Volckaert, S., De Bosschere, K., Danhieux, P., & Van Buggenhout, E. (2012, November). DNS tunneling for network penetration. In *International Conference on Information Security and Cryptology* (pp. 65-77). Springer, Berlin, Heidelberg.
- [7] Crotti, M., Dusi, M., Gringoli, F. and Salgarelli, L., 2007, June. Detecting http tunnels with statistical mechanisms. In *2007 IEEE International Conference on Communications* (pp. 6162-6168). IEEE.
- [8] TIAN, M. and ZHUANG, Y., 2009. Building the NIDS models with Smooth K-Windows statistical model. *Journal of Changchun University of Science and Technology (Natural Science Edition)*, (4), p.40.
- [9] Muzammil, M. J., Qazi, S., & Ali, T. (2013, September). Comparative analysis of classification algorithms performance for statistical based intrusion detection system. In *2013*

3rd IEEE International Conference on Computer, Control and Communication (IC4) (pp. 1-6). IEEE.

- [10] Duffield, N., Lund, C. and Thorup, M., 2002, November. Properties and prediction of flow statistics from sampled packet streams. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement* (pp. 159-171).
- [11] Tsuchiya, K. and Sato, H., ALAXALA Networks Corp, 2010. *Detection method for abnormal traffic and packet relay apparatus*. U.S. Patent 7,729,271.
- [12] Kim, H.S., Kang, K.S., Jeon, K.C., Kim, B.T. and Ahn, B., Electronics and Telecommunications Research Institute, 2011. *Method and system for ddos traffic detection and traffic mitigation using flow statistics*. U.S. Patent Application 12/946,849.
- [13] Verma, A., & Ranga, V. (2018). Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning. *Procedia Computer Science*, 125, 709-716.
- [14] Kim, Y., Lau, W.C., Chuah, M.C. and Chao, H.J., 2006. PacketScore: a statistics-based packet filtering scheme against distributed denial-of-service attacks. *IEEE transactions on dependable and secure computing*, 3(2), pp.141-155.
- [15] Özgür, A., & Erdem, H. (2016). A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. *PeerJ Preprints*, 4, e1954v1.
- [16] Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." *Military Communications and Information Systems Conference (MilCIS)*, 2015. IEEE, 2015.
- [17] Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, 25(1-3), 18-31.

- [18] Janarthanan, T., & Zargari, S. (2017, June). Feature selection in UNSW-NB15 and KDDCUP'99 datasets. In 2017 IEEE 26th international symposium on industrial electronics (ISIE) (pp. 1881-1886). IEEE.
- [19] Osanaiye, O., Cai, H., Choo, K. K. R., Dehghantanha, A., Xu, Z., & Dlodlo, M. (2016). Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2016(1), 130.
- [20] Primartha, R., & Tama, B. A. (2017, November). Anomaly detection using random forest: A performance revisited. In 2017 International conference on data and software engineering (ICoDSE) (pp. 1-6). IEEE.
- [21] Polikar, R. (2012). Ensemble learning. In *Ensemble machine learning* (pp. 1-34). Springer, Boston, MA.
- [22] Kasongo, S. M., & Sun, Y. (2020). Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. *Journal of Big Data*, 7(1), 1-20.
- [23] Husain, A., Salem, A., Jim, C., & Dimitoglou, G. (2019, December). Development of an Efficient Network Intrusion Detection Model Using Extreme Gradient Boosting (XGBoost) on the UNSW-NB15 Dataset. In 2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT) (pp. 1-7). IEEE.
- [24] Lim, H. K., Kim, J. B., Heo, J. S., Kim, K., Hong, Y. G., & Han, Y. H. (2019, February). Packet-based network traffic classification using deep learning. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)* (pp. 046-051). IEEE.
- [25] Raikar, M. M., Meena, S. M., Mulla, M. M., Shetti, N. S., & Karanandi, M. (2020). Data Traffic Classification in Software Defined Networks (SDN) using supervised-learning. *Procedia Computer Science*, 171, 2750-2759.
- [26] Yu, C., Lan, J., Xie, J., & Hu, Y. (2018). QoS-aware traffic classification architecture using machine learning and deep packet inspection in SDNs. *Procedia computer science*, 131, 1209-1216. Chicago

- [27] Eрман, J., Mahanti, A., & Arlitt, M. (2007, June). Byte me: a case for byte accuracy in traffic classification. In *Proceedings of the 3rd annual ACM workshop on Mining network data* (pp. 35-38).
- [28] Shafiq, Muhammad, Xiangzhan Yu, and Dawei Wang. "Network traffic classification using machine learning algorithms." *International Conference on Intelligent and Interactive Systems and Applications*. Springer, Cham, 2017.
- [29] Parsaei, Mohammad Reza, et al. "Network traffic classification using machine learning techniques over software defined networks." *International Journal of Advanced Computer Science and Applications* 8.7 (2017): 220-225.
- [30] Li, Wei, and Andrew W. Moore. "A machine learning approach for efficient traffic classification." *2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE, 2007.
- [31] Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50, 102419.
- [32] A. Binbusayyis and T. Vaiyapuri, "Identifying and Benchmarking Key Features for Cyber Intrusion Detection: An Ensemble Approach," in *IEEE Access*, vol. 7, pp. 106495-106513, 2019, doi: 10.1109/ACCESS.2019.2929487.
- [33] B. A. Tama, L. Nkenyereye, S. M. R. Islam and K. Kwak, "An Enhanced Anomaly Detection in Web Traffic Using a Stack of Classifier Ensemble," in *IEEE Access*, vol. 8, pp. 24120-24134, 2020, doi: 10.1109/ACCESS.2020.2969428.
- [34] A. Tesfahun and D. L. Bhaskari, "Intrusion Detection Using Random Forests Classifier with SMOTE and Feature Reduction," *2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies*, 2013, pp. 127-132, doi: 10.1109/CUBE.2013.31

- [35] Yan, Jinghua, and Jing Yuan. "A survey of traffic classification in software defined networks." *2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN)*. IEEE, 2018.
- [36] Mohammed, A. R., Mohammed, S. A., & Shirmohammadi, S. (2019). Machine Learning and Deep Learning Based Traffic Classification and Prediction in Software Defined Networking. *2019 IEEE International Symposium on Measurements & Networking (M&N)*. doi:10.1109/iwmn.2019.880504.
- [37] Mestres, A., Rodriguez-Natal, A., Carner, J., Barlet-Ros, P., Alarcón, E., Solé, M., Muntés-Mulero, V., Meyer, D., Barkai, S., Hibbett, M. J., Estrada, G., Ma'ruf, K., Coras, F., Ermagan, V., Latapie, H., Cassar, C., Evans, J., Maino, F., Walrand, J., & Cabellos, A. (2017). Knowledge-Defined Networking. *ACM SIGCOMM Computer Communication Review*, 47(3), 2–10. <https://doi.org/10.1145/3138808.3138810>
- [38] Levin, A., & Narendra, K. (1997). Identification of Nonlinear Dynamical Systems Using Neural Networks. *Neural Systems for Control*, 129-160. doi:10.1016/b978-012526430-3/50007-6
- [39] Kreutz, D., et al. "Software-defined networking: A comprehensive survey." *Proceedings of the IEEE*, vol. 103.1, pp. 14-76, 2015
- [40] Farhan, R., Maolood, A. and Hassan, N., 2020. Performance analysis of flow-based attacks detection on CSE-CIC-IDS2018 dataset using deep learning. *Indonesian Journal of Electrical Engineering and Computer Science*, 20(3), p.1413.
- [41] Yulianto, A., Sukarno, P., & Suwastika, N. A. (2019, March). Improving adaboost-based intrusion detection system (IDS) performance on CIC IDS 2017 dataset. In *Journal of Physics: Conference Series* (Vol. 1192, No. 1, p. 012018). IOP Publishing.
- [42] Panigrahi, R., & Borah, S. (2018). A detailed analysis of CICIDS2017 dataset for designing

Intrusion Detection Systems. *International Journal of Engineering & Technology*, 7(3.24), 479-482.

- [43] Epp, N., Funk, R., Cappo, C., Lorenzo-Paraguay, S. (2017). Anomaly-based web application firewall using HTTP-specific features and one-class SVM. In *Workshop Regional de Segurança da Informação e de Sistemas Computacionais*.
- [44] C. Pontes, M. Souza, J. Gondim, M. Bishop and M. Marotta, "A new method for flow-based network intrusion detection using the inverse Potts model," in *IEEE Transactions on Network and Service Management*, doi: 10.1109/TNSM.2021.3075503.
- [45] <https://www.unb.ca/cic/datasets/ids-2018.html>
- [46] <https://www.unb.ca/cic/datasets/ids-2017.html>
- [47] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018
- [48] <https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter>
- [49] Pavlyshenko, B. (2018, August). Using stacking approaches for machine learning models. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (pp. 255-258). IEEE.
- [50] Huang, J., & Ling, C. X. (2005). Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, 17(3), 299-310.