ANOMALY BASED NETWORK INTRUSION DETECTION USING MACHINE LEARNING

Master's Thesis

Abdisalam Abdullahi MOHAMED

Eskişehir, 2020

ANOMALY BASED NETWORK INTRUSION DETECTION USING MACHINE LEARNING

Abdisalam Abdullahi MOHAMED

Master's Thesis

Department of Electrical and Electronics Engineering

Programme in Telecommunication

Supervisor: Assoc. Prof. Dr. Nuray AT

Eskişehir

Anadolu University

Graduate School of Sciences

July 2020

ABSTRACT

ANOMALY BASED NETWORK INTRUSION DETECTION USING MACHINE LEARNING

Abdisalam Abdullahi MOHAMED

Department of Electrical and Electronics Engineering

Programme in Telecommunication

Anadolu University, Graduate School of Sciences, July 2020

Supervisor: Assoc. Prof. Dr. Nuray AT

The online dependency of the world has been sharply increasing for the last two decades, and this year, due to the pandemic, all sorts of life literally moved to online. Parallel to these, the number of attacks and criminal activities on the internet is steadily rising. Several security systems are deployed to protect network infrastructure against criminals. Network Intrusion Detection System (NIDS) is one of the most popular security systems used. It can be a signature-based or anomaly-based system. Although signature-based NIDS are efficient in preventing known attacks, they are futile against zero-day attacks. In detecting new attacks, the anomaly-based method is proved to be efficient. In this study, machine learning techniques are used for network anomaly detection. To this effect, a bunch mark dataset, CSE-CIC-IDS2018 is used. It is pre-processed and important features selected with Random Forest Regressor algorithm. The cleaned dataset with the selected features is fed to eight different machine learning algorithms. After reducing the number of features from the original 80 features to 17 features, the machine learning algorithms achieved the following success rates: Naïve Bayes 36%, QDA 50%, Random Forest 94%, ID3 94%, AdaBoost 94%, MLP 77%, and K Nearest Neighbours 95% and Gradient Boosting 95%.

Keywords: Intrusion Detection, Network Security, CSE-CIC-IDS2018, Naïve Bayes, QDA, ID3, Random Forest, AdaBoost, MLP, Network Anomaly Detection, Gradient Boosting, KNN, Feature Selection.

ÖZET

MAKINE ÖĞRENIMIYLE ANOMALI TABANLI AĞ SALDIRI TESPITI

Abdisalam Abdullahi MOHAMED
Elektrik-Elektronik Mühendisliği Anabilim Dalı
Telekomünikasyon Bilim Dalı
Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Temmuz 2020
Danışman: Doç. Dr. Nuray AT

Dünyanın internete bağımlılığı son 20 yıldır keskin bir şekilde artmaktadır; ve bu yıl pandemiden dolayı neredeyse her türlü yaşam internete taşınmış durumdadır. Buna paralel olarak, internette gerçekleşen saldırı sayıları ve suç faaliyetleri de sürekli olarak artmaktadır. Suçlulara karşı ağ altyapısını korumak üzere çeşitli güvenlik sistemleri uygulanmaktadır. Kullanılan en popüler güvenlik sistemlerinden biri ağ saldırı tespit sistemidir (NIDS). Bu sistem imza tabanlı veya anomali tabanlı olabilmektedir. İmza tabanlı ağ saldırı tespit sistemleri bilinen saldırıların önlenmesinde etkili olmalarına rağmen sıfır-gün saldırılarında etkisizdir. Yeni saldırıların tespit edilmesinde anomali tabanlı sistemlerin etkili olduğu kanıtlanmıştır. Bu çalışmada, ağ anomalilerinin tespitinde makine öğrenimi teknikleri kullanılmıştır. Bu amaçla referans veri seti CSE-CIC-IDS2018 kullanılmıştır. Bu set ön-işleme tabi tutulmuş ve önemli öznitelikler rasgele orman regresyon algoritması ile belirlenmiştir. Ayıklanarak düzenlenen veri seti seçilen özniteliklerle sekiz farklı makine öğrenimi algoritmasında kullanılmıştır. Gerçekte 80 olan öznitelik sayısının 17'ye düşürülmesiyle makine öğrenimi algoritmalarının başarı oranları şu şekilde elde edilmiştir: Naif Bayes 36%, QDA 50%, Rasgele Orman 94%, ID3 94%, AdaBoost 94%, MLP 77%, En Yakın K Komşu 95% ve Hızlandırılmış Gradyan 95%.

Anahtar Kelimeler: Saldırı Tespiti, Ağ Güvenliği, CSE-CIC-IDS2018, Naif Bayes, QDA, ID3, Rasgele Orman, AdaBoost, MLP, Ağ Anomali Tespiti, Hızlandırılmış Gradyan, KNN, Öznitelik Seçimi.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my supervisor, Assoc. Prof. Dr. Nuray AT for her endless support and patience throughout my research. Without her contribution and help, the goal of this project would not have been realized.

I am extremely grateful to the Turkish Government Scholarships Program (YTB), for providing me this precious opportunity. Their benevolent help highly contributed to my career and future life.

I owe a special thanks to my family, my mother, Fatima, and my father, Abdullahi, for their support and love kept me going on this work.

The contributions and the input of my mentor Kahraman KOSTAS are truly appreciated. His technical support is well appreciated.

Abdisalam Abdullahi MOHAMED

17/07/2020

STATEMENT OF COMPLIANCE WITH ETHICAL PRINCIPLES AND RULES

I hereby truthfully declare that this thesis is an original work prepared by me; that I have behaved in accordance with the scientific ethical principles and rules throughout the stages of preparation, data collection, analysis, and presentation of my work; that I have cited the sources of all the data and information that could be obtained within the scope of this study, and included these sources in the references section; and that this study has been scanned for plagiarism with "scientific plagiarism detection program" used by Anadolu University, and that "it does not have any plagiarism" whatsoever. I also declare that, if a case contrary to my declaration is detected in my work at any time, I hereby express my consent to all the ethical and legal consequences that are involved.

Abdisalam Abdullahi MOHAMED

CONTENTS

Page

CURRICULUM VITAE

LIST OF TABLES

Page

LIST OF FIGURES

Page

GLOSSARY OF SYMBOLS AND ABBREVIATIONS

AdaBoost	:	Adaptive Boosting
ANIDS	:	Anomaly-based NIDS
ANN	:	Artificial Neural Networks
AWS	:	Amazon Web Service
BW	:	Baum Welch
CART	:	Classification and Regression Trees
CIC	:	Canadian Institute for Cybersecurity
CSE	:	Communications Security Establishment
DARPA	:	Defence Advanced Research Project Agency
DDoS		Distributed Denial of Service Attack
DGSOT	:	Dynamically Growing Self-Organizing Tree
DoS	:	Denial of Service
HMM	:	Markov Model
HOIC	:	High Orbit Ion Cannon
ID3	:	Ross Quinlan called Iterative Dichotomiser 3
IoT	:	Internet of Things
IPS	:	Intrusion Prevention System
ISCX 2012	:	Intrusion detection evaluation dataset
KDD	:	Knowledge Discovery and Data Mining
KNN	:	K Nearest Neighbour
LOIC		Low Orbit Ion Cannon
LSTM	:	Long Short Term Memory
MLP	:	Multi-layer perceptron
NIDS	:	Network Intrusion Detection System
QDA	:	Quadratic discriminant analysis
SNIDS	:	Signature-based NIDS
SSH	:	Secure Shell
VT	:	Viterbi Training
XGboost	:	Gradient Boosting.
XSS	:	Cross Site Scripting attacks

1. INTRODUCTION

In 1995 only 0.4% of the world population had access to the internet; today (2020), more than 53% of the world population are connected to the internet [1]. The internet is one of the most significant technological achievements the world has seen in the 21st century, but the proliferation of internet technology came with immense security problems. The industry of computing is exposed to an ever-increasing likelihood of unexpected downtime resulting from various cyber-attacks and security breaches by crooked cybercriminals and terrorists [2]. In addition to that, with the emergence of the internet of Things (IoT) and 5G communication technology, more gadgets will be connected to the internet and the internet dependence of our daily life will grow higher [3].

Together with the immense benefits of the internet, it also introduced many ways cybercriminals can cause all kinds of havoc by compromising the security of the systems connected to it. A report by Symantec in 2019 shows that the Formjacking attack, which is the kind of attack where the cybercriminals load malicious code onto retailers' webpages and hence steals customers' credit card information, targeted more than 4,800 websites on average every month in the year 2018 [4]. The tactics used by cybercriminals like intruders become more rampant and sophisticated with time, ways to tackle them like intrusion detection and firewall systems are also needed to keep evolving [5].

For any given communication to be a secure communication, it must have the following characteristics: Confidentiality, Integrity and Availability. Network intrusion, however, is any set of activities that aim to compromise one or more of those pillars of network security. To monitor networks or systems for malicious activity or policy violations, a device or a software application called Network Intrusion Detection System (NIDS) is used [5].

Based on their approaches of detection, NIDS can be divided into two types: Signature-based NIDS (SNIDS) and Anomaly-based NIDS (ANIDS). Signature-based NIDSs detect attacks by searching for specific patterns in network traffic or predefined well-known malicious instruction tactics used by the attacker. Signature-based NIDS performs well in detecting any known attack. For the unknown attacks or the cases when there is anomaly in the network traffic, the performance of Signature-based NIDS decreases sharply [5], [6].

The Anomaly-based NIDS emphasizes the detection of unusual network activities, examining the general patterns of the network traffic flow without the need to inspect each packet. In that way, the ANIDSs have been successful in detecting previously unseen (novel) attacks, hence efficient against zero-day attacks (unknown and unaddressed vulnerability) [6]. With the ever-changing attack scenarios, ANIDSs have become the most appropriate intrusion detection technology. There are three steps in IDS operations: tracking and collecting the network flow data, cleaning the raw data and converting it into an input-format and lastly classification engine is employed to classify the network traffic as a normal or a malicious symbols.

This thesis aims to create an anomaly-based intrusion detection system utilizing machine learning using the latest dataset available. Eight different machine learning algorithms will be trained with an up-to-date dataset by aggressively reducing the number of features and the performance levels achieved by the machine learning algorithms will be compared with the results obtained by the previous studies in the same field.

1.1. Research Objectives

This study aims to achieve the following goals:

- To study the most appropriate machine learning algorithms for network intrusion detection.
- To explore different forms of attacks on a network system and enhance the accuracy rate with the latest dataset available.
- To evaluate the accuracy level achieved in this thesis by comparing the results achieved with some studies previously conducted on the same area, hence contributing to the literature in the anomaly detection domain by presenting models trained with minimum number of features possible.

1.2. Thesis Structure

This thesis is organized in the following manner:

In the second chapter, some background information and basic concepts in the study area ranging from basic computer networking concepts to the details on topics like accessible IDS datasets, network security and attacks, are presented. Since machine learning algorithms are used in thesis work, some basic definitions about machine learning are briefly noted. In the subsequent sections, details specific to this study, like

the nuts and bolts of the dataset and machine learning algorithms used in this research are discussed. Finally, some important research papers in the literature are reviewed.

The third chapter consists of two sections. In the first section, tools and platforms utilized in the research implementation in the aspects of software and hardware platforms, pre-processing techniques, experimentation steps, and performance matrices are illustrated. The experimentation roadmap is elaborated in the second section, starting with the pre-processing methods, followed by feature engineering and experimental procedure.

In the fourth chapter, the results of the machine learning algorithms are discussed and compared with other studies selected from the literature.

In the fifth chapter, the conclusion is presented, and future works are recommended for subsequent enhancements.

2. BACKGROUND AND LİTERATURE REVİEW

2.1. Computer Network Security

As a general definition, a computer network is defined as a set of computers connected to share resources [5]. The internet, which is a vast universal network open to all, dramatically increases the risk of network attacks. Although securing a network became close to impossible, there are several systems such as intrusion detection systems, intrusion prevention systems and firewalls intended to defend networks against malicious activities from inside or/and outside. An attack instigated by an individual within the limits of a particular network is defined as an inside attack. A person aiming to get access to a specific resource beyond their authorized reach is an example of inside malicious activity. To spot an insider is, to some extent difficult compared to an outsider. An attack from the outside mainly comes from an unauthorized or illegitimate user of a given system. It can be from a juvenile or experienced and highly organized criminals, terrorist groups or sometimes it can be state-sponsored terrorists [5].

Networks have two major components: hardware and software. Although there are some risk vectors and vulnerabilities for both of these two components, researches indicate that hardware threats are detected with physical damage on the devices but not the data. An attack against non-hardware computing resources and data is termed as a software attack [7].

2.2. Network Attacks

A set of malicious activities aiming to disrupt, deny, degrade or destroy information and services in computer networks can be defined as a network attack. A network attack is targeted to the systems through the data flow on network systems intending to compromise the Integrity, Confidentiality or Availability of computer network resources [2],[8].

The classification of network attacks contributes to the necessary theoretical interpretation in explaining attacks. Attack classification is made according to general and specific usage and the motif of attackers [2].

Classic taxonomy research by Hansman [9] classified attacks into ten categories based on the attacker's main behaviour:

Virus: A self-reproducing piece of software that can reside on an existing program and corrupt a given system without being noticed.

Worm: A self-replicating software program that spreads through computer network interfaces without user involvement.

Trojan: A program designed for a particular normal action but executes malicious code.

Buffer overflow: A process initiated to gain control or to crash other processes by overflooding the capacity of a given buffer.

Denial of service: A network attack that aims to block legitimate users from accessing a system or network resource.

Network attack: An attack designed to crash the users on a given network or even the network itself by interfering network protocols from lower layers to the upper layers.

Physical attack: Aims to inflict physical damage on computer system components or network infrastructure.

Password attack: An attack with the ambition to illegally gain a password of a system or a network user. Observing a series of failed login attempts is a clear indication of a brute-force attack to guess a password.

Information gathering attack: It is the act of collecting information to find vulnerabilities by scanning or exploring existing computer networks.

2.3. Network Intrusion

Any attempt to break or/and misuse a system is considered as an "intrusion." An intruder skilfully takes advantage of specific system loopholes and vulnerabilities and, if not detected and stopped very quickly, can cause havoc in networks and systems [2].

The process of identifying and tackling any malicious activity aiming at the computing and network resources of any given system is called intrusion detection. Any piece of hardware or software intended to monitor, detect or respond to activities in a network or on a host computer is defined as an Intrusion Detection System [2].

Network Intrusion Detection Systems (NIDSs) are used to shield networks against external attacks, providing a second line of defense for computer systems on the internet. It strengthens the efficiency of firewall systems in detecting different types of malicious network and computer system activities. They have been explored in the field of network

security research for over three decades, with extensive documentation that outlines the requirements and potential design of such systems [2],[10]. Figure 2.1 shows the classification of intrusion detection systems according to different aspects.

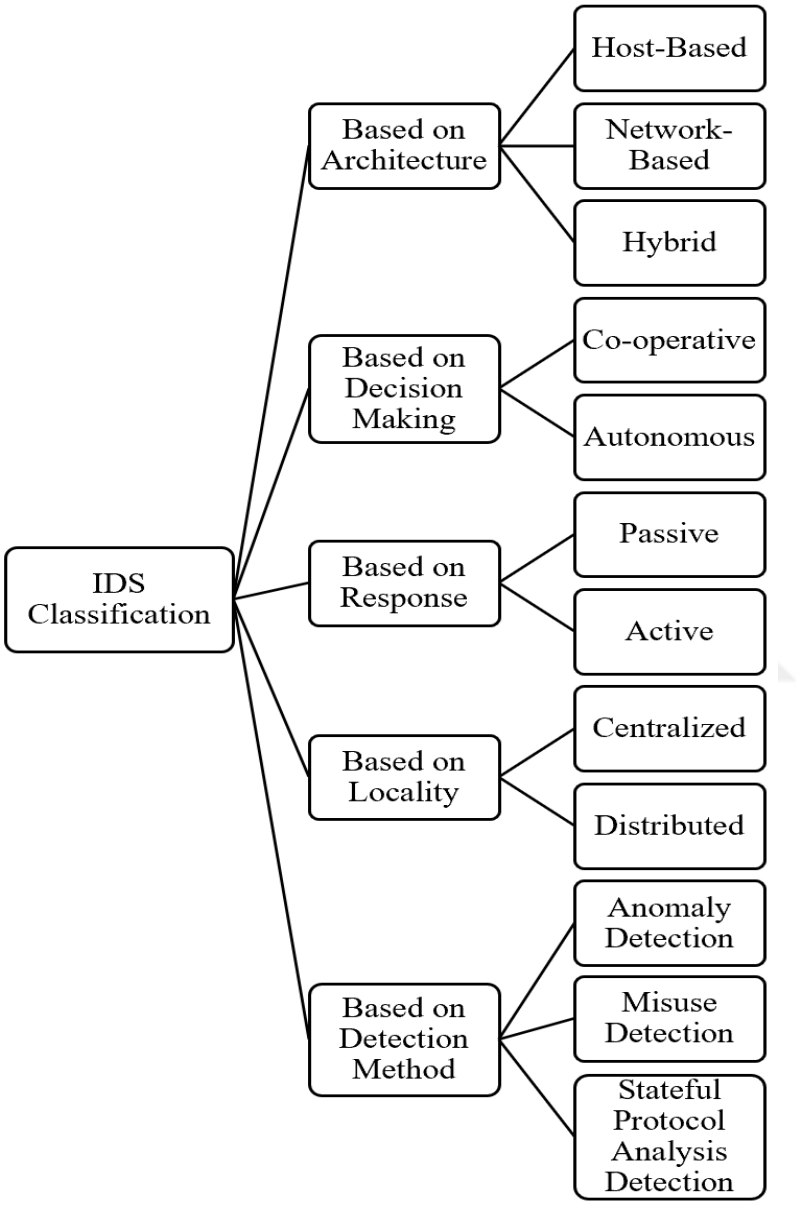

Figure 2.1. *Intrusion detection system classification*

Intrusion detection systems are typically made up of three major parts: sensors, analyzer and user interface. Sensors collect data. The analyzer is used to digest data collected to detect anomalies. Upon detecting an intrusion, the analyzer(s)'s output is translated into an alarm. The user interface provides a way to monitor and interact with the analyser's output and operations [12].

In general, IDSs are deployed beyond firewall systems to detect malicious activities firewalls may miss providing an extra layer to the Defence-in-depth strategy. A typical example of the Defence-in-depth strategy is shown in Figure 2.2 [13].

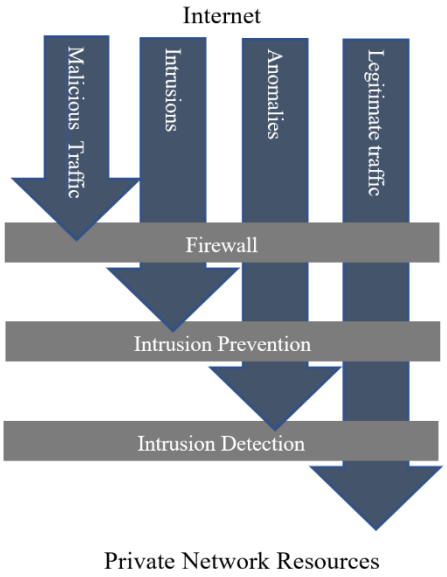

Figure 2.2. *Defence-in-depth strategy [13]*

As discussed above, intrusion detection systems merely detect and report suspicious activities, so the network administrator can be notified and decide the appropriate action to take. An IDS with an automatic decision-making procedure is an Intrusion Prevention System (IPS). IPSs can act autonomously against malicious network activities in a quicker way than human-dependent IDS systems. However, the performance of IPSs is negatively affected by its high false-positive rate flagging legitimate traffic as an attack. Researches are underway to overcome IPS drawbacks and increase its efficiency [14].

2.4. Open Source IDS Software

There are several freely available IDS software in the literature. The following sections will highlight two of the most popular ones.

2.4.1. Snort

Snort is a free network-based intrusion detection and prevention software built by Martin Roesch in 1998[15]. It is a signature-based intrusion detection system that relies

on a set of rules for spotting malicious activities in the network. The reference set contains a mixture of specific byte pattern called signatures that shows a particular attack. A sample of signatures of the attacks is available in the snort website and once it is configured, it can be applied to snort to detect attacks. Snort has some other applications that can be used as a packet sniffer and packet logger. In addition to that, Snort is used for protocol analysis and content matching as well. It is the most widely used IPS software and like all other open-source software, community contributions help to enhance Snort by reporting new threats as they emerge and reviewing rules get updated frequently [16], [17].

2.4.2. Bro

Bro is a passive publicly available IDS analyser initially created by Vern Paxson in 1995. It is used as a security monitor to scrutinize all data traffic on a given data link hunting for suspicious activity indicators. Moreover, Bro has other usages like performance measurements and troubleshooting. It generates a massive set of detailed log files of the network's activity. In addition to the log files, Bro provides useful built-in functionality like file extraction from HTTP sessions, malware detection, SSL certificate validation and more [18].

2.5. Machine Learning

Machine learning is the art and science of programming computers to optimize a given performance criterion by learning from example data or experience. In the machine learning process, computers are trained on data samples (training data) fed to them, and their learning performance can be tested on different data (test data). Machine learning is usually used in applications in which classical techniques are ineffective, like interpreting complicated data, simplifying complex problems, and predicting trends and outputs of different processes [19].

Based on the method of training and supervision, machine learning algorithms are grouped into four categories: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning [20].

2.5.1. Supervised Learning

Supervised learning algorithms are implemented when the dataset in hand is labelled. In the training phase, the algorithms will be exposed to different examples and statistically infer knowledge from the information in the sample examples. With the experience gained in the training session, algorithms will be able to classify or predict new data. Comparing the output of the algorithms with the expected result will give a clear indication of how well a model learnt from the input examples. Popular supervised learning algorithms are Decision Trees, K-Nearest Neighbours, and Random Forests [20].

2.5.2. Unsupervised Learning

Machine-learning algorithms don't rely on labelled data samples for learning but strive to find hidden characteristics in an unlabelled dataset. It only has access to only input data with no associated target variables. Unsupervised learning approach has two main classes:

Clustering: Using the input dataset to uncover natural groupings in the given data like K-Means Clustering, Gaussian mixture models and hierarchical clustering.

Dimensionality reduction: Multidimensional input data is transformed into smaller dimensions, which can reflect the real image of the original data such as principal component analysis (PCA), multidimensional scaling and manifold learning [21].

2.5.3. Semisupervised Learning

Labelling data is tedious and costly, especially when the dataset is big and has plenty of unlabelled samples. Algorithms devised to deal with such datasets are called semisupervised learning algorithms. The goal of semi-supervised learning is to comprehend how amalgamating labelled and unlabelled data may alter the learning performance, and construct systems that take advantage of such a mixture [20].

2.5.4. Reinforcement Learning

Reinforcement Learning is the technique where the learner is at the same time decision-making entity that leads activities in a system and gets a reward or penalty for its actions to solve a particular problem. After a set of brute-force methods, the algorithms

learn the best strategy, which is the chain of activities that maximize the total reward value hence constructs a new rule [19].

2.5.5. Machine Learning Algorithms Used In This Thesis

Different machine algorithms are used for various purposes. In selecting a machine learning algorithm, several thigs are put into the consideration like the kind of the problem to be addressed, size of the dataset, required accuracy, training time, number of parameters and number of features. Another critical choice to make in choosing which algorithm to use is whether it is going to be a multi-class or a one-class algorithm, one-class classification (OCC) algorithms are made to construct classification models when the negative class is either missing, improperly sampled or not explicitly defined. This exceptional situation pressures the learning of efficient classifiers by making class boundary just with the knowledge of a single (positive) class.

In this thesis, eight different supervised learning algorithms are used, taking advantage of the labelled dataset available. Used machine learning algorithms are: Multilayer Perceptron (MLP), Naïve Bayes, Decision Trees (ID3), Random Forest, AdaBoost and Gradient boosting (XGboost). Quadratic discriminant analysis (QDA), K-Nearest Neighbours.

2.5.5.1. Multi-layer perceptron (MLP)

Multi-Layer Perceptron (MLP) is the basic building block for Artificial Neural Networks (ANN). ANN is a machine learning model that imitates the way the human brain operates in learning, decision making, and gaining new information from given data. The human brain has lots of interconnected nerve cells known as neurons, copying that concept, artificial neural networks are made up of a network of artificial cells [22].

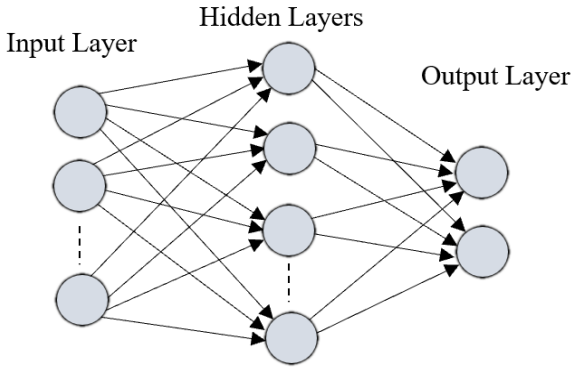

Figure 2.3. *Multilayer perceptron*

MLP has three different stages or layers: an input layer, hidden layer and output layer. The input data is fed to an MLP at the input layer, multiplied with the weights in each neuron, and then transferred to the hidden layer where the data is processed and sent to the output layer. The output will be compared with the target value, and their difference, which is known as the error term will be backpropagated to the hidden and the input layers to adjust the weights of each neuron in the network to achieve minimal error in the output. The complexity of an MLP is determined by the number of hidden layers present in that particular MLP; more layers and neurons mean better performance but increase the processing time. In this research, a Multilayer Perceptron with three hidden layers with thirteen neurons in each layer is used as a result of experimental trails and fine tuning [19].

MLP Algorithm:

- **Initialisation**
  - Initialize all weights to small random values
- **Training**
  - Repeat:
    - For each input vector:

      Forwards phase:

      - Compute the activation of each neuron j in the hidden layer(s) by:

$$h_\zeta = \sum_{i=0}^{L} x_i v_{i\zeta} \tag{2.1}$$

$$a_\zeta = g(h_\zeta) = \frac{1}{1 + \exp(-\beta h_\zeta)} \tag{2.2}$$

- Work through the network until you get to the output layer neurons, which have activations:

$$h_k = \sum_j a_i w_{jk} \tag{2.3}$$

$$y_k = g(h_k) = \frac{1}{1 + \exp(-\beta h_k)} \tag{2.4}$$

Backward phase:

- Compute the error at the output using:

$$\delta_o(k) = (y_k - t_k) y_k (1 - y_k) \tag{2.5}$$

- Compute the error of the hidden layer(s) using:

$$\delta_h(\zeta) = a_\zeta (1 - a_\zeta) \sum_{k=1}^{N} w_\zeta \, y_k \delta_o(k) \tag{2.6}$$

- Update the output layer weights using:

$$w_{\zeta k} \leftarrow w_{\zeta k} - \eta \delta_o(k) \, a_\zeta^{hidden} \tag{2.7}$$

- Update the hidden layer weights using:

$$v_i \leftarrow v_i - \eta \delta_o(k) x_i \tag{2.8}$$

  - (if using sequential updating) randomize the order of the input vectors so that you don't train in exactly the same order each iteration
- until learning stops

- Recall
  - Use the Forwards phase in the training section above.

2.5.5.2. Naïve Bayes

Naïve Bayes is an algorithm simplified with the addition of the independence condition on the popular Bayes' theorem illustrated below:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.9}$$

Naïve Bayes algorithm applies a network of probabilities characterized by a parent node for an unobserved event and several child nodes representing the observed events. It was primarily created for text classification. In the case of a statistical network, it is presumed that each child node to be independent of others, an assumption regarded as the base for the Naïve Bayes theory but not correct all time. This unfound (very "naive") assumption leads the algorithm to be less useful for dense problems because the features are improbable to be independent. Despite that big shortcoming it has, Naïve Bayes remains one of the most used algorithms in machine learning, thanks to its short training time and low computational cost [21],[23]. In this study, a particular type of Naïve Bayes algorithm called Gaussian Naïve Bayes is used.

The following is the Naïve Bayes algorithm pseudocode [24]

Naïve Bayes Algorithm

Input:

Training dataset T,

F= (f1, f2, f3,.., fn) // value of the predictor variable in testing dataset.

Output:

A class of testing dataset.

Step:

1. Read the training dataset T;

2. Calculate the mean and standard deviation of the predictor variables in each class;

3. Repeat

Calculate the probability of *fi* using the gauss density equation in each class;

Until the probability of all predictor variables (f1, f2, f3,.., fn) has been calculated.

4. Calculate the likelihood for each class;
5. Get the greatest likelihood.

2.5.5.3. Decision Tree

Decision Tree is a machine learning algorithm used for numerical and class data classification. Its operation is based on a predefined target variable and leaf nodes that support the decision-making steps leading to one of the hierarchical structures of the algorithm. Decision tree makes classification using short series of decisions where each decision is made based on the input data.

Decision trees, however, fail to generalize the same way as deep neural nets because a node at the lower levels (leaf) of a given decision tree is used only by a small portion of the available training samples. This, in turn, leads for the lower parts of the decision tree to overfit if the size of the training set is not substantially larger than the depth of the tree in use. To avoid overfitting, some leaf nodes should be pruned out of the tree [25].

An algorithm invented by Ross Quinlan called Iterative Dichotomies 3 (ID3) is used in this study. It is a well-known classification algorithm based on Information Entropy used in machine learning and natural language processing. Its basic working concept is to map all examples to different groups according to different values of the predetermined attribute set; its principle is to ascertain the best classification character form given condition attribute sets. Information gain is used as an attribute selection criterion and the attribute with the highest information gain is set to the splitting attribute of the present node, to keep information entropy needed by the divided subsets small [26]:

Suppose:
S is the set of examples, F as a possible group of features out of the set of all possible ones, and $|S_f|$ is the tally of the number of members of *S* that have value *f* for feature *F*:

$$Gain(S,F) = Entropy(S) - \sum_{f \in values(F)} \frac{|S_i|}{|S|} Entropy(S_f) \tag{2.10}$$

Decision Tree Algorithm:

- **If** all examples have the same label:
    - Return a leaf with that label
- **Else** if there are no features left for testing:
    - choose the feature $\hat{f}$ that maximises the information gain of S to be the next node using equation (2.10).
    - add a branch from the node for each possible value f in $\hat{f}$
    - **for** each branch:
        - calculate S_f by removing $\hat{f}$ from the set of features.
        - recursively call the algorithm with S_f, to compute the gain relative to the current set of examples.

2.5.5.4. Random Forest

Random forest is a machine learning algorithm that utilises the concept of decision trees. It is one of the ensemble algorithms in machine learning. In this algorithm, a network of trees called forest is formed by combining a vast number of decision tree structures built in different mechanisms [27].

The most common approach in data-driven algorithms is to construct one robust model. However, a more useful method would be to make an ensemble of algorithms for some unique learning objectives like a bunch of neural networks added to give rise to a better result. However, in real-life applications, the ensemble techniques combine a huge number of weak, simple algorithms to attain a sharper ensemble result. Some common examples of ensemble models are neuron network ensembles, Random forest and Boosting techniques [28].

In the training phase, the Random Forest algorithm constructs *N* decision trees using the training dataset. In this process, each tree's training set is randomly resampled, making sure that unique N decision trees are obtained. At last, voting is done through selecting updated estimates from estimates of the N number of trees made. The highest-rated value is concluded as the final value. The error resulting from the forest of tree classifiers varies with the strength of each tree and the correlation between the whole trees in the forest [29].

Random Forest Algorithm:

- **For** each of *N* trees:
    - create a new bootstrap sample of the training set
    - use this bootstrap sample to train a decision tree
    - at each node of the decision tree, randomly select *m* features, and compute the information gain (or Gini impurity) only on that set of features, selecting the optimal one
    - repeat until the tree is complete

2.5.5.5. AdaBoost

In general, boosting (hypothesis boosting) implies any ensemble method of gathering some weak learners to form a substantially strong learner. Boosting algorithms train predictors in succession such that each rectifies the errors of the one ahead of it. There are several boosting techniques available, but in this thesis work, two of the most popular ones namely Adaptive Boosting (AdaBoost) and Gradient Boosting are used [20].

AdaBoost is a machine learning algorithm that works in a little different form than other classifier algorithms. It is based on the same decision tree concept, but the training data is continuously tailored to direct the algorithm to focus on misclassified samples. Furthermore, the classifiers are combined in succession for each classifier to contribute to the general boost [30],[20].

Several iterations are made, and at each iteration, the misclassified samples' importance is increased and those correctly classified will be decreased. The algorithm is frequently boosted from a very weak model to the maximum number of estimators is reached. The majority vote attains the results. The following is the AdaBoost algorithm [20],[31].

AdaBoost Algorithm

- initialise all weights to 1/N, where N is the number of data points
- while $0 < \epsilon_t < \frac{1}{2}$ (and $t < T$, some maximum number of iterations):
    - train classifier on $\{S, w^{(t)}\}$, getting hypotheses $h_t(x_n)$ for datapoints x_n
    - compute training error

$$\epsilon_t = \sum_{n=1}^{N} w_n^{(t)} I\left(y_n \neq h_t(x_n)\right) \tag{2.11}$$

- set $\alpha_t = \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- Update weights using:

$$w_n^{(t+1)} = w_n^{(t)} \exp\left(\frac{\alpha_t I\left(y_n \neq h_t(x_n)\right)}{Z_t}\right) \; where\; Z_t is\; a\; normalisation \tag{2.12}$$

- Output

$$f(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{2.13}$$

2.5.5.6. Gradient Boosting

Gradient Boosting algorithm works the same way as AdaBoost. It sequentially adds predictors to a band, each one improving the results of the former. Instead of weight adjustment at each iteration as in the AdaBoost, this algorithm seeks to fit the new predictor to the lingering errors of the previous predictor. The algorithm is detailed below [28],[31]:

Gradient Boosting Algorithm

Inputs:

- input data $(x, y)_{i=1}^{N}$
- number of iterations M
- choice of the loss-function $\psi(y, f)$
- choice of the base-leaner model $h(x, \theta)$

Steps:

1. initialize $\widehat{f_0}\; with\; constant$
2. **for** $t = 1$ to M do
3. compute the negative gradient $g_t(x)$
4. Fit a new base-learner function $h(x, \theta_t)$

5. Find the best gradient descent step-size ρ_t:

$$\rho_t = argmin_t \sum_{i=1}^{N} \psi\left[y_i, \widehat{f_{t-1}}(x_i) + ph(x_i, \theta_t)\right] \tag{2.14}$$

6. Update the function estimate:

$$\hat{f}_t \leftarrow \hat{f}_{t-1} + \rho_t h(x_i, \theta_t)$$

7. End for

2.5.5.7. K-nearest Neighbour

K Nearest Neighbour (KNN) is one of the most popular sample-based machine learning algorithms. It is built on the assumption that similar samples in a dataset will exist close to each other with similar properties in another dataset. With that assumption set, KNN finds the class of a new dataset which is unclassifiable by utilizing training data of known classes. This conclusion is reached by observing the nearest neighbours of the new input data vector [30].

Using a plane with N characteristics, when input data is fed to the model, K number of candidate classes will be examined by checking the distance of that particular input from each category and the smallest K number of voting classes are chosen from these distance values [20]. In Figure 2.4, k=3 because three nearest neighbours are voting for the new sample's class.

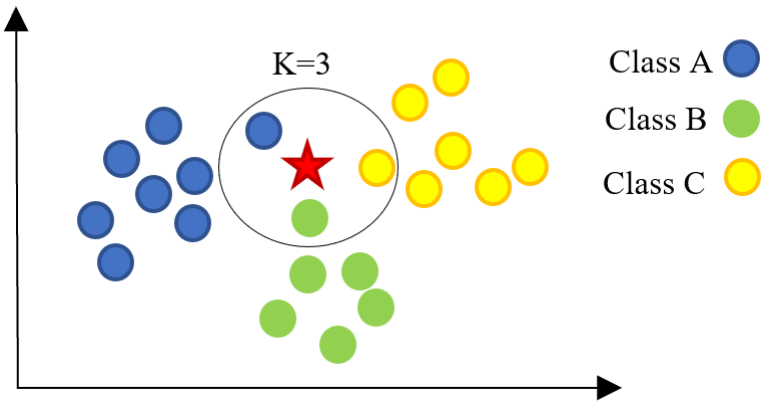

Figure 2.4. *KNN algorithm with K = 3*

KNN Algorithm [32]

- **Input:** x,S,d (inputs, set of distances, distance between two points)
- **Output:** class of x
- **For** $(\acute{x}, \acute{l}) \in$S do
    - Compute the distance d($\acute{x}, x$)
- **End for**
- Sort the $|S|$ distances by increasing order
- Count the number of occurrences of each class l_j
- Among the k nearest neighbours
- Assign to x the most frequent class

2.5.5.8. Quadratic Discriminant Analysis (QDA)

Quadratic Discriminant Analysis (QDA) is a statistical algorithm for assigning a measured data to one set of many available groups. When the assignment process is finished, every observed data sample should be assigned to its appropriate class. If miss classification occurs, and the error term is created, this is called error rate [33].

The goal of discriminant analysis is to make an assignment decision with a minimum error rate. In the case of normal distribution data with equal variances, Linear Discriminant Analysis is made. However, finding a class of data with equal variance is not an assumption that holds in real-life cases. To deal with such conditions, Quadratic Discriminant Analysis is used with the condition that the number of samples observed is more than the number of groups made [33].

2.6. Some Popular Intrusion Detection Datasets

Anomaly detection has been a hot topic for many researchers in their quest for detecting novel attacks. However, deploying it to the real-world systems remained impossible due to system complexity in fulfilling the pre-deployment criterion of testing, evaluation, and tuning. A realistic dataset is indispensable for testing and evaluation.

To get such a phenomenal dataset for public research is very challenging because most descriptive datasets are internal or confidential and the few available are heavily anonymized or outdated, thus fail to represent new trends. As the overall network

activities and network attacks evolve, the use of optimal and dynamic datasets has become a condition for developing a reliable intrusion detection system. According to Gharib et al. [34], for a dataset to be a benchmark dataset for intrusion detection, it should have the following eleven properties: Anonymity, Available Protocols, Attack Diversity, Complete Interaction, Complete Capture, Complete Network Configuration, Heterogeneity, Complete Traffic, Feature Set, Metadata and Labelling.

For decades, different institutions and researchers have been trying to develop realistic datasets for better intrusion detection [35]. In the following subsections, some of the most popular datasets in the IDS field will be presented.

2.6.1. DARPA/KDD Cup99

Defence Advanced Research Project Agency (DARPA) funded a project by the MIT Lincoln Labs to deliver an IDS benchmarking dataset. Through that project, the KDD98 (Knowledge Discovery and Data Mining (KDD)) dataset was created in 1998. At that time, the KDD98 dataset greatly contributed to the research on IDS, although its accuracy and real-life applicability have been widely criticized [36].

The dataset is generated using multiple computers with an nternet connection to simulate a small-size US Air Force base with restricted personnel. The data stream contains a mixture of processes like file transfer via FTP, browsing, e-mail services and IRC messages. The dataset includes 38 attack scenarios and normal traffic. That attacks can be grouped as follows: Denial of Service (DoS), User to Remote (U2R), Probe, and Remote to Local (R2L) attack groups [37].

The 1998 DARPA Dataset is the parent dataset which is used to create the KDD Cup99 dataset used in Third International Knowledge Discovery and Data Mining Tools Competition (KDD, 1999). These two datasets are outdated because of the lack of records for the newer malware attacks. Nevertheless, KDD99 is still in use within the IDS research area [38].

2.6.2. NSL-KDD

NSL-KDD is a publicly available dataset, which has been derived from the KDD cup99 dataset in 1999. It became a perfect substitute for DARPA98, which had too many duplications. These repetitions had very adverse effects on the performance of the

machine learning algorithms, and randomly selecting records from within the dataset could not reflect the dataset's properties. Aiming to bridge that gap, Tavallaee et al. [39] created the NSL-KDD dataset in 2009. It contains 22 attack scenarios and 41 features, 21 of which refer to the connection itself and 19 others illustrate the nature of connections within the same host.

2.6.3. ISCXIDS2012

ISCX 2012 (Intrusion detection evaluation dataset) was created by analysing seven-day internet stream on a testbed designed by the Canadian Institute for Cybersecurity. In the creation process of this dataset, real network traffic records were analysed to differentiate normal computers' activities from real traffic of network protocols like HTTP, SMTP, SSH, IMAP, POP3, and FTP. At the time this dataset was created, it was a very realistic, labelled dataset and with good attack scenarios. But, according to today's internet traffic, it is inadequate because it does not have any content representing new and crucial network protocols like HTTPS. The simulated attack distribution does not reflect real-world statistics [34],[40].

2.6.4. CICIDS2017

CICIDS2017 contains both benign/normal and a bunch of new malware attack types such as Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. It is labelled and the labelling is accomplished using indicators like timestamp, source and destination IPs, source and destination ports, protocols and attacks. In the creation process, a complete network topology including Modem, Firewall, Switches, Routers, and machines with diverse operating systems like Windows 10, Windows 8, Windows 7, Windows XP, macOS, iOS, and Linux was set up to collect the dataset which contains 80 features from the captured network traffic. The CICFlowMeter tool [41] is used to extract the features from the network flow. It is worth noting that this dataset has a considerable class imbalance [42].

2.6.5. CSE-CIC-IDS2018

CSE-CIC-IDS2018 dataset is developed by the collaboration of the Government of Canada's national cryptologic agency called Communications Security Establishment

(CSE) and the Canadian Institute for Cybersecurity (CIC). It includes six different attack scenarios, such as Brute-force, DoS, DDoS, Web attacks, Botnet, and Infiltration. In the dataset generating phase, an attacking infrastructure and a victim organisation are made, the attacking infrastructure has 50 machines and the victim organization has five departments with 30 servers and 420 machines. The Using CICFlowMeter-V3 tool [41], 80 features being extracted from traffic generated (network traffic and system logs of all machines). The end product is a diverse, inclusive and world-class dataset in the field of intrusion detection. It was made to reflect on complete user profiles to represent events and behaviours observed on the network. Those profiles were merged to generate a distinct set of datasets with a unique feature set contributing to the evaluation domain [43].

As mentioned in the previous sections, in the CSE-CIC-IDS2018 dataset creation, two network infrastructures are designed, attacking and victim networks. The attacking side contains 50 machines and the victim side has 420 machines and 30 servers in 5 departments. The captured network traffic and logs of each machine, together with 80 features extracted from the captured traffic using the CICFlowMeter-V3 tool, constitute the dataset [37]. Profiles are used to generate datasets in an orderly way containing comprehensive descriptions of intrusions and prototypes of protocols and applications. Two different kinds of profiles used in this process are discussed below:

The B-Profile used to make benign traffic by extracting the behaviour of a bunch of human users by encapsulating event logs with machine learning and statistical analysis, producing features like packets per flow, payload forms, request time and packet size distribution of protocols. It can be easily run by a human operator or CIC-BenignGenerator to generate realistic benign events. For CSE-CIC-IDS2018 dataset, HTTP, HTTPS, SMTP, POP3, IMAP, SSH, and FTP protocols are simulated [43].

M-Profiles are and autonomously used to label an attack scenario implicitly. For this dataset, six different attack scenarios are simulated. For each attack, a particular scenario is defined according to the used network topology and the attacks are executed from one or more machines of the attacking infrastructure. Overall, the implemented network is a normal LAN network topology on the Amazon Web Service (AWS) platform. Five subnets (R&D department, Technician Department, Management Department, IT department, Secretary and operation department, and server rooms) are

created to make experiments more realistic. In the IT department, the Ubuntu operating system is used, and in all other departments, MS Windows operating systems (Windows 8.1 and Windows 10) are used. For the server room, different MS Windows servers like 2012 and 2016 are installed [43].

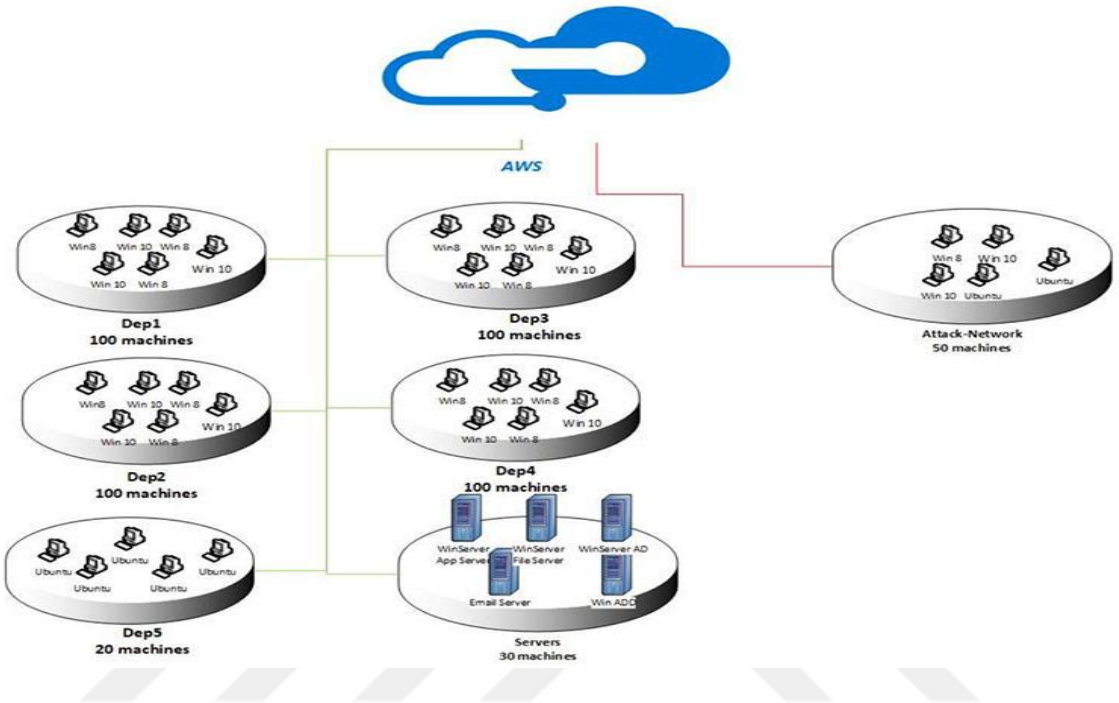

Figure 2.5. *Network topology of the CSE-CIC-IDS2018 dataset generation platform [43]*

The CSE-CIC-IDS2018 dataset contains six different attack scenarios: Brute-force, Botnet, DoS, DDoS, Web attacks, and infiltration, those six scenarios constitute to the fully labelled data of a total of 14 types of attacks called, Botnet attack, FTP-BruteForce, SSHBruteForce, BruteForce-Web, BruteForce-XSS, SQL Injection, DDoS-HOIC attack, DDoS-LOIC-UDP attack, DDoS-LOIC-HTTP attacks, Infiltration, DoSHulk attack, DoS-SlowHTTPTest attack, DoS-GoldenEye attack, and DoS-Slowloris attack. The subsequent sections describe individual attacks in detail.

2.6.5.1. FTP-BruteForce

A brute force attack is: "An attempt to crack a password or a username or find a hidden web page or find the key used to encrypt a message." [44]. It ultimately relays on trial and error. Depending on the complexity and the length of the victim's password or username and the technical level of the attacker. A brute-force attack may take anywhere from milliseconds to years. It is one of the most popular attack techniques hackers use,

although it is an old attack technique that existed before the internet in the form of a cryptanalytic attack [45],[46].

Tools like Hydra, Medusa, Ncrack, Metasploit modules, and Nmap NSE scripts are some of the most popular tools used for brute-force attacks and password cracking purposes. Other tools like Hashcat and Hashpump are used in password hash cracking. In creating the CSE-CIC-IDS2018 dataset, the Patator, which is one of the most comprehensive multi-threaded, reliable and flexible tools written in Python, is used. In this dataset, there are two modules of brute-force attacks, FTP and SSH, where a Kali Linux machine is set to be the attacker machine and an Ubuntu 14.0 system as the victim machine. Going through every password needs more time; to shorten the attack time, hackers wage a type of attack called dictionary attack [47], which involves storing most likely username and passwords in a dictionary and then executes the attack using those entities in the dictionary. A dictionary attack is used in generating this dataset by using a colossal dictionary containing 90 million words as the list of passwords to be cracked [43].

File Transfer Protocol (FTP) is a network protocol that facilitates client-server file transfer in a network. For any files transfer to occur through FTP, a verified username and password are required. In that sense, the FTP-Patator attack targets to illegally acquire this username and password, thus compromise the victim's system [45].

The FTP-BruteForce attack scenario is executed on Wednesday-14-02-2018 and recorded in CSE-CIC-IDS2018 dataset in a file with the same name as the execution date [43].

2.6.5.2. SSH-Bruteforce

SSH (Secure Shell) is a network security protocol that encrypts and decrypts operations of various network services over networks. SSH had overtaken Telnet (which is not encrypted) in remote access aspects. That popularity makes SSH a vital target for cybercriminals, especially SSH Brute-force attacks [49], [50].

SSH brute force attack is a popular type of attack waged by an attacker aiming to get access to a remote machine by performing repetitive authentication credential guessing trails by using all possible combinations of passwords until the correct one is reached [51]. The SSH-Brute-force data in the CSE-CIC-IDS2018 dataset is created with

the same Patator [48] tool used in the FTP Brute-force attack. Data generated is stored in the file with the name Wed-14-02-2018 in the dataset file [43].

2.6.5.3. Brute Force-XSS

Cross Site Scripting attacks are in the list of the well-known web attack techniques. It all starts with the injection of a malicious piece of code into the targeted webpage. Immediately a victim visits this webpage; the code gets executed without being noticed. The results can be very devastating to the extent that it may include identity theft, impersonation and arbitrary code execution [52]. XSS is reported by [53]as the most vital vulnerability for web applications. In the dataset used in this study, the data samples representing XSS attack are generated in two days and its files labelled as Thurs-22-02-2018 and Fri-23-02-2018, respectively [43].

2.6.5.4. Bruteforce -Web

It is a type of attack involving brute-force techniques and targeting web applications. More on this attack has been discussed under the sections above. The samples representing this type of attack are collected under the file named Thurs-22-02-2018 [43].

2.6.5.5. Botnet

The word Bot comes from the word “Robot.” It is a common term used to describe an automated process like Google bot, which is used to collect information from the net. Bots, however, are not only used for good intentions; but they are also used for malicious activities such as information theft and launching pads for the distributed attacks. Malicious software is installed on the victim machine without being noticed, then infects the machine and, in that process pass the control of the machine to a remote attacker and after that, becomes under the direct command of the attacker. Such infected machines used by the attackers are referred to as zombie machines and they are used primarily to wage Distributed Denial of Service attacks (DDoS) [54], [55].

Botnets use a level command and control mechanism with a wide range of protocols; this diversity makes botnet detection hard. In the passive mode of a given network, there are limited or no activities in the network and in that condition detecting

Botnets is even harder [56]. In the active mode, however, the packet size and the packet flow with the TCP Push (PSH) flag used to speed-up packet flow and help IDSs detect Botnets [57].

In the CSE-CIC-IDS2018 dataset, a Trojan horse malware package that runs on versions of Microsoft Windows called Zeus is used to simulate many malicious and criminal acts like banking information theft by man-in-the-browser keystroke logging and form grabbing. Similarly, it is used to install Crypto-Locker ransomware. Also, in the dataset generation, an open-source botnet called Ares botnet is used. Ares has the following capabilities:

Remote cmd.exe shell.

Persistence.

File upload/download.

Screenshot.

Keylogging.

In the execution scenario, target machines are infected with Zeus and Ares botnets and every 400 seconds screenshots are requested from the zombies. The samples collected for this type of attack is simulated in the Friday-02-03-2018 file of the dataset [43].

2.6.5.6. DoS-GoldenEye

DoS attacks directly aim at blocking legitimate users' rights of system access by reducing system availability. DoS attacks flood networks with malicious load comprising of superfluous network datagrams or normal packets that consume more of network buffers, CPU processing capabilities, and overall network bandwidth. As system resources form a bottleneck, performance drops and eventually, the targeted system may crash [58]. DoS attacks have different types by in CSE-CIC-IDS2018 dataset; there are four DoS attack scenario DoS-GoldenEye, DoS-Slowloris, DoS-SlowHTTPTest and DoS-Hulk. GoldenEye is a multi-threaded python-based HTTP DoS Tool used to test site vulnerabilities regarding DoS attacks. It allows numerous parallel connections to a given URL. It uses the Keep-Alive method, to substantially increase the size of the files transmitted over a given TCP connection. In addition to that, it deactivates HTTP- Cache-Control using the NoCache message feature. By using an attack vector containing 'HTTP

Keep-Alive and NoCache', GoldenEye attack immediately depletes system resources. Attack packets of this attack type are unencrypted; it doesn't support fabricated IP (spoofed) addresses. GoldenEye tool can be executed on operating systems like Windows, Linux and macOS [59], [60]. In the CSE-CIC-IDS2018 dataset, the DoS-GoldenEye attack scenarios are created and collected in a file named Thurs-15-02-2018 [43].

2.6.5.7. DoS-Hulk

HULK stands for HTTP Unbearable Load King, a multi-purpose attack tool that can be used to execute both DoS and DDoS attacks. It can take servers down in a very short period by generating TCP SYN flood and HTTP GET flood requests. Another extremely dangerous trait it has is that it can completely conceal the real user platform and instead use a different sample for each attack execution [59]. The file Fri-16-02-2018 in the CSE-CIC-IDS2018 dataset contains the DoS-Hulk attack samples.

2.6.5.8. DoS-Slowloris

A tool developed on Perl programming language that has both GUI and the command line interfaces is used to generate a Slowloris attack [63]. It floods the victim with TCP SYN requests to the target victim machine. When the target of this type of attack is a web server, it is reasonable to observe incomplete and smaller in sizes TCP packets with SYN flags [59]. The samples collected in the Thurs-15-02-2018 file of the dataset contains data generated for DoS- Slowloris [43].

2.6.5.9. DoS-SlowHTTPTest

Slow Read DoS attack is created by Sergey Shekyan [64]. In this attack scenario, an attacker typically sends a legitimate HTTP request to the target Web server and then reads the response very slowly. The attacker sends a legitimate request after the normal three-way-handshake of the TCP protocol. Following that, the attacker advertises window size smaller than average window size and slows

down the HTTP response operation. Upon advertising, a window size of 0, the Web server stops sending data despite maintaining the connection open. Receiving more legitimate requests makes the Web server quickly reach maximum capacity and ceases to serve legitimate clients. Since this attack uses legitimate HTTP requests, it is extremely

difficult to segregate the normal from malicious traffic. On the way to safeguard networks against this type of attack is to continuously monitor the network layer and pay attention to the small-sized packets. Fri-16-02-2018 file in the dataset is the file designated for the samples of this attack [43].

2.6.5.10. DDOS-LOIC-(UDP/HTTP)

DDoS is an organized attack intended to prevent legitimate users from accessing network resources and even causes systems to crash from extraordinary laud. It is executed with a huge number of compromised hosts. At the preliminary stage, attackers spot vulnerabilities in a given network to infect a vast number of machines with malware and eventually remotely control them. In the second phase, the attackers exploit the infected machines (Zombies) to spread attack packets to the victim machine(s) without the infected machines knowing the harmful activities they are partaking. Depending on the concentration of attack packets and the number of zombies used to launch the attack, corresponding damage will be inflicted on the victim network and machines [61].

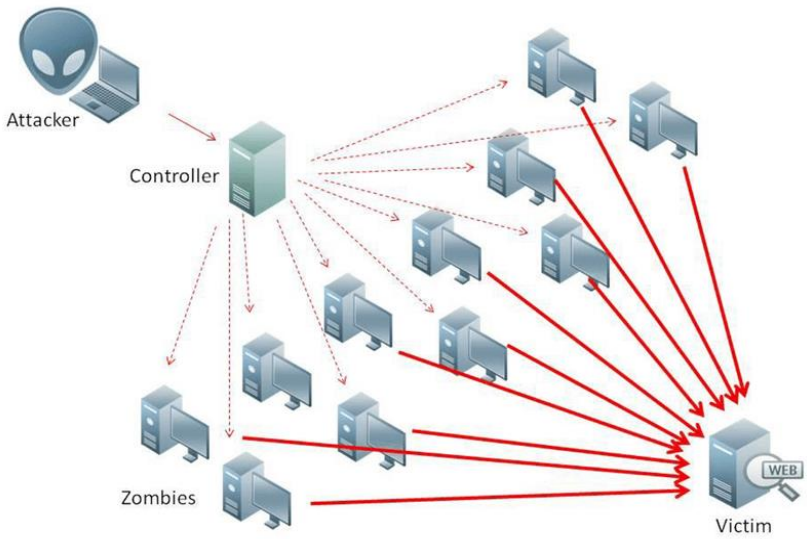

Figure 2.6. *DDoS attack [62]*

LOIC is an open-source network testing tool created by Praetox Technologies for the network stress test. It is a GUI based DDoS attack tool and can run on Windows and macOS operating systems. Its ease of use and widespread availability allows criminals to

efficiently conduct coordinated and severe DDoS attacks without the need to be experienced [59],[65].

Using the LOIC attack tool, a hacker needs to acquire target URL or IP, after that, chooses whether to launch an HTTP, UDP or TCP flood. In the TCP and UDP cases, it sends message strings and packets to specific ports on the target. In the case of HTTP flood, it sends a continuous stream of GET requests. If the attack turns out successful, LOIC opens several connection requests towards the target server and pumps it with huge traffic, thus overwhelming the server. In the CSE-CIC-IDS2018 dataset, LOIC has being used to conduct two kinds of DDoS attacks, LOIC-HTTP and LOIC-UDP and their data samples are recorded in the files: Tues-20-02-2018 and Wed-21-02-2018 [43], [59].

2.6.5.11. DDOS-HOIC

High Orbit Ion Cannon (HOIC) is an open-source, BASIC-written, speedy, multi-threaded, network stress testing tool that can take down hundreds of websites at once through HTTP flood and POST requests sent to the target server. HOIC has the ability to drain the resources of the victim server [66]. In the CSE-CIC-IDS2018 dataset, using a free HOIC tool with four machines, the DDoS attack is conducted, and the data generated is stored in the Wed-21-02-2018 file [43].

2.6.5.12. SQL Injection

Unchecked input fields at the user interface of applications with bugs help the attacker to alter the SQL commands and queries sent to the database, such an attack is called SQL Injection attack. These kinds of attacks are prevalent because finding vulnerabilities and exploiting it is very easy and the damages it may inflict on the victim are so huge due to the direct access it has to the database. A successfully executed SQL injection attack gives the hacker an unauthorized executive command (read, insert, change or delete) over data, gain access to privileged database accounts, impersonate a legitimate user and get access to the webserver [67],[68].

To avoid the danger posed by this attack, developers must strictly follow recommended coding practices and security guidelines and use code analysers to perform security reviews of the code and regular auditing to make code vulnerability assessments.

The dataset has two files Thurs-22-02-2018 and Fri-23-02-2018 for this attack type [43],[67].

2.6.5.13. Infiltration

Network Infiltration is a network attack from inside; it is mainly carried out through exploiting a vulnerable software like Adobe Acrobat Reader. After the exploitation phase completes with success, a backdoor will be implemented on the targeted machine allowing the attacker to wage different attacks on the victim's network like IP sweep, service enumerations using Nmap and port scan [69]. The data samples representing this attack type are in two files, Wed-28-02-2018, and Thursday-01-03-2018.

2.7. Related Works

Intrusion Detection Systems (IDS) have been a hot research area for not only the cybersecurity industry but also academia. Therefore, a lot of research papers have been published on IDS technology over the past three decades. In this section, several pieces of research employing machine learning algorithms for network anomaly detection have been inspected in chronological order.

Chebrolu et al. in 2005 conducted a study to find computationally useful features for IDS applications using two popular algorithms: Bayesian networks (BN) and Classification and Regression Trees (CART). The researchers finally came up with a hybrid model of ensemble and base classifiers for intrusion detection purposes. Applying their model on KDD cup 99 datasets, the study achieved some impressive results of 100 % accuracy for Normal (Benign) traffic, Probe and DOS attack detection. And some average 84% accuracy in detecting U2R and R2L scenarios [70].

Latifur Khan et al. used the SVM algorithm with improved training time by combining it with Dynamically Growing Self-Organizing Tree (DGSOT) algorithm. The model commences with an initial training set and gradually moves to the clustering structure formed by the DGSOT algorithm. 1998 DARPA dataset is used with four classes to be classified, Normal, DOS and Probe detected with an accuracy above 90%, whereas 43% and 23% accuracy for R2L and U2R are recorded respectively. Comparing the outcome of their study, researchers proved that their methodology outperformed the Rocchio Bundling technique and random selection in the accuracy loss and training period [71].

Weiming Hu et al. presented an AdaBoost algorithm-based IDS using decision stumps as weak classifiers. The weak classifiers are used with continuous and categorical features together, forming a strong classifier. For a minimum overfitting, an Adaptable initial weights strategy is utilized, resulting in improved performances. Four attack types: DOS, U2R, R2L and PROBE from Knowledge Discovery and Data Mining CUP 1999 dataset are examined in the experiments with and without overfitting handling techniques. Without handling overfitting, a detection rate of 99.159% is achieved while in the case of overfitting handled 99.166% accuracy is achieved. The researchers presented that their algorithm has a lower error rate and computational complexity compared to other algorithms tested with the same dataset [72].

Yinhui Lia et al. proposed research with a steadily feature elimination technique to classify five different scenarios in the KDD cup99 dataset: Normal traffic, DOS, R2L, U2R and probing. The goal of the study was first to compress the dataset need in making IDS, implement dimensionality reduction strategies to reduce the features from 41 to 19 critical features with the help of the GFR technique and in the end, develop a classifier using SVM. The researchers achieved to propose a stable, reliable IDS with 98.6249% accuracy and an MCC value of 0.861161 [73].

A research conducted by Warusia Yassin et al. in 2013 came up with a new model which is combining K-Means Clustering and Naïve Bayes Classifier algorithms to prevent false alarm constraints. The IDS model is implemented on the ISCX 2012 dataset. The researchers came up with an IDS with a high performance of 99% for the combination of K-Means clustering and Naïve Bayes and 98.8% accuracy for Naïve Bayes alone [74].

Sharafaldin et al. In 2017, published a yardstick research paper by implemented seven popular machine learning algorithms (including Random Forest (RF), K-Nearest Neighbours (KNN), Naïve -Bayes (NB), AdaBoost, Multilayer perceptron, Quadratic Discriminant Analysis (QDA) and ID3) on CICIDS2017 dataset to detect fifteen attack types. To operate with minimum and most important features, the Random Forest Regressor class of Scikit-learn is used. The study showed the following results for each algorithm: KNN 0.96%, QDA 0.92%,RF 0.97%, MLP 0.76%, Naïve -Bayes 0.84%, AdaBoost 0.77% and ID3 0.98% [69].

V. Kanimozhi and T. Prem Jacob proposed an IDS model that detects Botnet attacks targeting the financial sectors. The IDS model is created by using artificial intelligence with the CSE-CIC-IDS2018 dataset. The system proposed is a real-life

applicable system with a superb 99.97% Accuracy and an average area under the ROC curve of 0.999 with minimal False Positive rate of 0.03. To make the system more powerful, the study suggests the use of a framework based on GPU instead of CPU [75].

A research paper by Zhou and Pezaros implemented six popular machine learning algorithms (Random Forest, MPL, Naïve Bayes, ID3, KNN and QDA) with CSE-CIC-IDS2018 dataset to train a model to detect Zero-Day attacks. In the training phase, the models train with fourteen attack scenarios in the dataset and in testing, eight novel (Zero-Day) attack type and regular traffic are used. The model accuracy is evaluated in terms of precision, recall, F1 value, and time overhead. Decision tree algorithms outperformed all other algorithms. The model has 100% accuracy on Zero-Day attack and normal data alone, and 96% accuracy for scrambled Zero-Day attack and benign data, and with 5% false-positive rates [76].

A study by Chadza et al. trained two Hidden Markov Model (HMM) algorithms called Baum Welch (BW) and Viterbi Training (VT) on CSE-CIC-IDS20118 dataset. Using uniform random and count-based initialisation methods, the algorithms are ranked in All States (AS), Current State (CS), and the Next State prediction (NS). The study revealed that the count-based initialisation technique gives the best results in detecting all state and current state. Count-based Viterbi Training algorithm has 97.0% accuracy and with BW, it achieved as high as 97.5% accuracy for all states and current states predictor. The prediction of the next state, the accuracy rate, became close to 65% for random and uniform initialisation methods on both BW and VT [77].

Lin et al. presented a dynamic network anomaly detection system based on deep learning methods. In the study, Long Short Term Memory (LSTM) is used to make a deep neural network architecture and to improve the model performance, an Attention Mechanism (AM) is used. To deal with the class imbalance problem in the CSE-CIC-IDS2018 dataset, SMOTE and an improved loss function algorithm are deployed. The model achieved an accuracy of 96.2% [78].

3. METHOD AND MATERİALS

In this chapter, tools and platforms used in the research implementation procedure, be it software or hardware, in-depth pre-processing techniques, experimentation steps and performance evaluation matrices, will be discussed in detail.

3.1. Materials

3.1.1. Software Tools

3.1.1.1. Python programming language

Python is an open source, interpreted, object-oriented, high-level programming language. The built-in data structures and dynamic typing and binding make Python the best in rapid application development. It is easy to learn with a readable syntax; this reduces the cost of program maintenance. With the use of modules and packages, Python supports code reuse practices. Since it is an open-source programming language, Python interpreters and libraries are available and are freely distributed. In addition to that, it has numerous specialized libraries for machine learning and data science applications. In this study, a free Python distribution called Anaconda running Python 3.7.4 is used [79].

3.1.1.2. Scikit-learn

Scikit-learn (Sklearn) is an open-source Python module used for machine learning made with SciPy and distributed with the 3-Clause BSD license. It allows users to run several machine learning algorithms with ease. All algorithms used in this study are run on the Scikit-learn module [80].

3.1.1.3. Pandas library

Pandas is a free, flexible, speedy, robust and easy to use tool used for data analysis and exploitation created with Python programming language. Users use Pandas to undertake several operations like column/row omission, filtering, addition, and replacement. In the experiments, the Pandas library is extensively utilized [81].

3.1.1.4. Matplotlib

Matplotlib is an open-source, multiplatform library created on NumPy arrays used to visualize data in Python. As needed by this thesis work, the Matplotlib library is used to create graphs of different kinds [82].

3.1.1.5. NumPy (Numerical Python)

NumPy is a Python library that is intended to operate as an interface for storing and operating on complex data buffers allowing the mathematical and logical operations to be conducted swiftly. Since this thesis work involves some calculations, this library became inevitable [83].

3.1.2. Hardware Platform

In this section, the technical specifications of the machine used to implement all experiments are outlined. The information given below is crucial for getting the experiments' real sense of execution time since the execution time is one of the most descriptive evaluation criteria for machine learning algorithms:

Processor:	Intel(R) Core(TM) i7-4712HQ CPU @ 2.30GHz 2.30 GHz
Installed Memory (RAM):	8 GB
System Type:	Windows 10 Home 64-bit Operating system, x64-based processor
GPU:	NVIDIA GeForce GT 750M

3.2. Model Evaluation Metrics

Evaluation metrics are connected to machine learning algorithms. Different metrics are used for various tasks, but there are some commonly used metrics like precision and recall. In this research, four familiar information retrieval evaluation metrics (Precision, Recall, Accuracy and F1-Score) are used. The criterion is valued between 0 and 1.

The confusion matrix is an outline of the prediction results of a classification models [19]. The total number of correctly and incorrectly classified samples are summarized in True Positive, True Negative, False Positive, and False Negative format, as shown below:

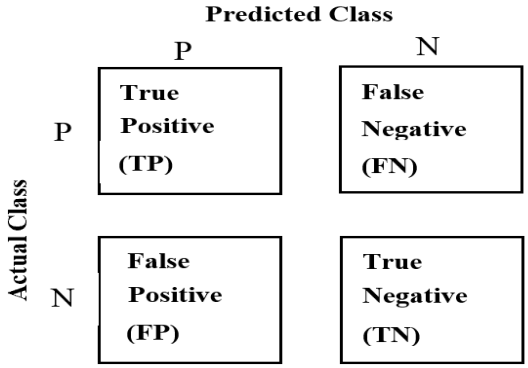

Figure 3.1. *Confusion Matrix*

In the case of Attack and Benign classification, the following four scenarios are considered:

True Positive (TP): An Attack sample classified as attack sample.

False Positive (FP): A Benign sample classified as an Attack sample.

False Negative (FN): An Attack sample classified as a Benign.

True Negative (TN): A Benign sample classified as a Benign.

Accuracy: The ratio of correctly classified samples to the total number of samples [84].

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

Recall: The ratio of samples classified as an attack to all attack samples [84].

$$Recall = \frac{TP}{TP + TN}$$

Precision: The ratio of correctly classified samples as the attack to all samples classified as attacks [84].

$$Precision = \frac{TP}{FP + TP}$$

F1-Score: Harmonic-mean of recall and precision. In this research, F1-Score is considered over other performance matrices because it a clear description of the model success [84].

$$F1 - Score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precission}}$$

3.3. IMPLEMENTATİON METHOD

3.3.1. Implementation Procedure Overview

Getting data is a good beginning in the Machine Learning implementation process, but the data obtained is not always clean enough to be directly fed to the learning algorithms. Data pre-processing is a fundamental step in cleaning the data from the errors resulting from missing records or redundant values. It also involves encoding labels, visualizing and understanding the dataset itself [85].

As indicated in the previous sections, CSE-CIC-IDS2018 dataset [43] is used in this study and this section; several pre-processing steps are taken to clean the dataset from some defects observed before implementing the machine learning algorithms for classifying normal and malicious traffic. Upon cleaning the dataset, it is divided into two parts, training and test. Since the dataset contains 80 features, the most important features are selected to be fed to the algorithms. Finally comes the implementation of the machine learning algorithms in two approaches. Figure2.6. below illustrates the implementation process.

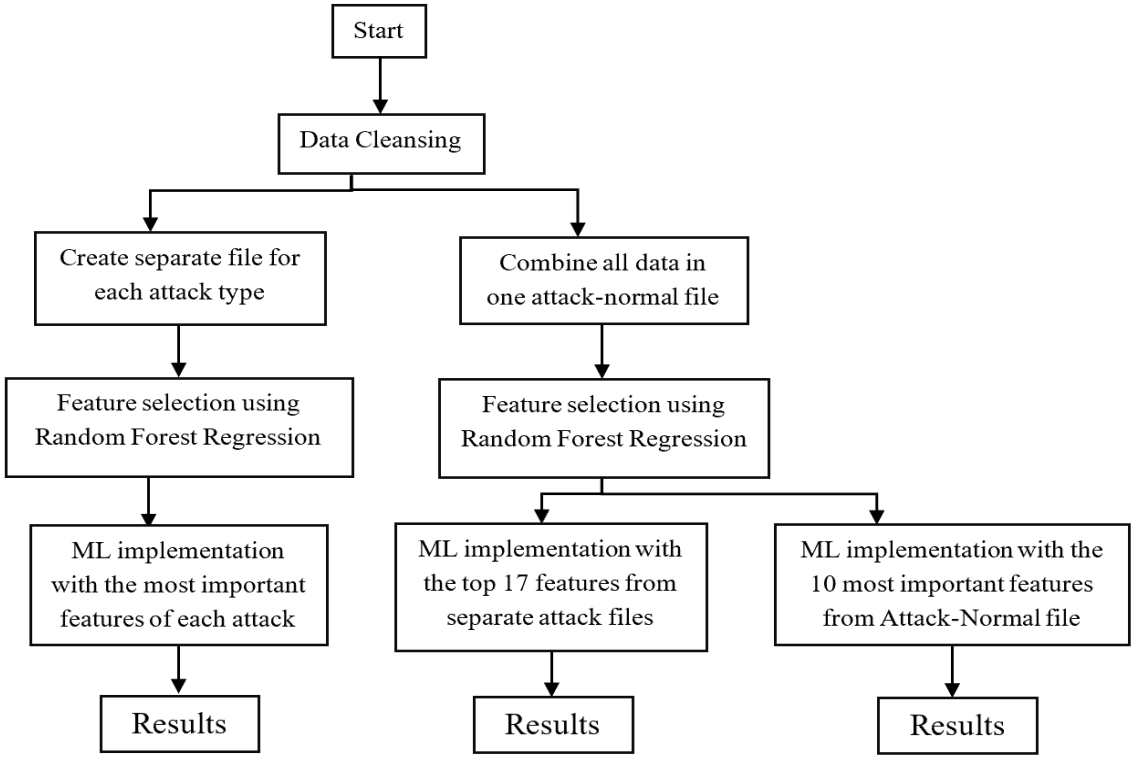

Figure 3.3.1. *Experimental Setup*

3.3.2. Data Cleanup

In the process of data cleansing, some defects of the CSE-CIC-IDS2018 dataset are cleaned, and some records are edited. We start with analysing the data to comprehend its characteristics. The dataset consists of original traffic in Packet Capture (Pcap) files, logs, pre-processed and labelled, and feature selected CSV files. In this research labelled CSV files with a total of 80 traffic features extracted using CICFlowMeter are used. The dataset contains normal (Benign) and attack traffic comprised of some of the most common attack types (14 types of attacks) in ten CSV files. In total, the dataset used consists of 9332770 records constituting 71.1% Benign 28.9% attack labels.

Table 3.3.1. *Number of normal and attack samples in the CSE-CIC-IDS2018 dataset*

Traffic Type	**Number of Samples**
Benign	6584535
DDOS attack-HOIC	686012
DDoS attacks-LOIC-HTTP	576191
DoS attacks-Hulk	461912
Bot	286191
FTP-BruteForce	193360
SSH-Bruteforce	187589
Infiltration	161934
DoS attacks-SlowHTTPTest	139890
DoS attacks-GoldenEye	41508
DoS attacks-Slowloris	10990
DDOS attack-LOIC-UDP	1730
Brute Force -Web	611
Brute Force -XSS	230
SQL Injection	87
Total Number Samples	9332770
Total Attack Samples	2748235 **(28.9%)**
Total Benign Samples	6584535 **(71.1%)**

In the first steps of data cleanup, thousands of incorrect records were deleted; samples containing Infinity and/or missing values (NaN) were converted to -1 and 0, respectively. The third column of the dataset 'Timestamp' has some irregularities that result in errors in processing and has the least feature importance among all features. Thus, it is completely dropped. Lastly, a separate CSV file for each attack type is created; samples representing individual attack types collected together with more normal traffic samples forming CSV files containing 30% single attack type and 70% normal traffic, leaving no room for errors resulting from the class imbalance [78] in the original dataset.

3.3.3. Experimental Setup

In general, Machine learning is used to acquire valid, novel, useful and informative patterns from a given dataset in that sense, data is one of the main components of every Machine Learning procedure, it allows for a model to be trained and tested to highlight the performance of the algorithms used. In short, Machine Learning is defined as building models of data [86]. Most of the datasets used in teaching Machine Learning are structured into two separate sets: training and testing sets. The former is fed to the Machine Learning algorithms in the learning phase and the latter is used to test the performance of the algorithms. CSE-CIC-IDS2018 dataset, however, does not have predetermined training and testing sets, it contains a single big labelled dataset. Hence, the data should be sub-divided into train and test sets.

There are different approaches used to evaluate machine learning models: n-fold cross-validation (mainly 10-fold) and single train-test split (70%-30% or 80%-20%) [20]. K-fold cross-validation is employed when the number of samples in some categories in the dataset is minor or lopsided. Simultaneously, the single train-test split is useful when the dataset contains a relatively large number of samples in every data category [87],[85].

In the experiments, Machine Learning Models are fed with the whole samples in the dataset combined in one file to make binary classification on the bases of attack-normal classification. Similarly, each attack type is separately evaluated by classifying a specific attack and normal traffic.

In this thesis work, the famous Sklearn command, 'train_test_split ' is used [88]. It is a command that divides the data into two portions at the size of the user's choice. The ideal partitioning is 20% test, 80% training data [20] and so is the preferred method.

The train_test_split command split arrays or matrices into a random train and test subsets. Performing the train-test split at once is not technically advised and the result may not be correct; to avoid such an inconvenience situation, the dataset splitting is performed ten different times and their mean takes us to the final result.

3.3.4. Feature Selection

Most of Machine Learning algorithms can handle a dataset with thousands of features. It's considered to be a very useful strategy to have more features to uplift the accuracy of the model. But in machine learning, more isn't always better. Because more features may usher model to memorize the features-target mapping in a detailed fashion; as a result, the model may face the overfitting problem. This results in an apparent increase in the model's accuracy at the training phase but degrades the model performance on previously unseen data in the testing phase. Selecting a subset of features is the preferred way to avoid overfitting and at the same time improve the accuracy of the model when fed with new data. In addition to that, the use of a smaller number of features substantially reduces the computational cost of training and prediction. Similarly, it helps to understand the data better, hence, decide the right model for the right data [21],[89].

The Cleaned-up CSE-CIC-IDS2018 dataset [43] used for this thesis work consists of 80 features; the details of all features are presented in the appendix. Training a model with all those features is computationally expensive, and a model trained with such a very appellant dataset may perform poorly in a real-world scenario. Therefore, feature selection is made by evaluating the importance of each feature in the dataset.

3.3.4.1. Random forest regressor

Random Forest is a multiuse tool; it can be used for both regression and classification purposes. It has a stand-alone estimate of generalization error mechanism, making cross-validation unnecessary. In addition to that, it is adjustable to attain desired results. Variable importance measure is one of the best services provided by Random forest algorithms. It is an algorithm that outperforms the other traditional methods like forwarding selection and backward elimination. It also has many other uses like to impute missing values [90],[69]. Random Forest Regressor is based on tree concept and during the tree construction phase, each variable undergoes several chances to be added in the classifier, therefore even weakly relevant features that are slightly related with the

decision feature will be added for making classifiers. In addition to that, the importance of the features is determined using only the trees using specific feature, hence, the signal for features contributing even to a limited number of trees remains visible. For that reason, Random Forest Regressor is most preferred, hence, in this study, the Random Forest Regressor class of Sklearn is used in implementing the feature selection operation [69],[91].

3.3.4.2. Importance of the domain knowledge in feature selection

The efficiency of machine learning models can be significantly enhanced by means of feature selection as a pre-processing step. But, the feature selection process to be effective, it should be a data-driven process [50]. The need for domain knowledge in feature engineering exists across the board, in network intrusion detection. However, there are some important considerations to be made when training a model with a given dataset. Some features can be misleading as they may look significant, but their underlying importance in identifying network anomalies may be relatively small or non-existing.

In a typical network-based detection, the traffic passing through the network should be monitored for malicious activity. At the same time, the detection mechanism also needs to be fast and efficient. To cope with those demands, flow-based analysis became the preferred option. According to some network security research [45], most researchers prefer to focus on analysing flow data in detecting attacks through network flow monitoring instead of packet inspection. The five most popular attributes used to define a network flow (mainly in TCP/IP connections) is a 5-tuple key constituting of source IP, destination IP, source port, destination port, and network protocol. In the detection process, analysing network traffic at the flow-level is faster than analysing each packet because of the reduced amount of data to be processed. Flow data is directed to packet headers and not to packet payload information, hence very convenient for detecting attacks with encrypted payload [45].

Since there is more attention to the aforementioned features, it has been discovered that the attackers are evading those well-monitored features. They avoid using well-known ports, sidestep security apparatus by using generated/fake IP addresses (spoofing). In addition to that, some ports are applied dynamically; port numbers can range from 1 to 65535, meaning port numbers other than those observed in training

sessions my appear in the testing phase, thus disrupting the evaluation mechanism. Also, some applications are transmitted over the same port as well, therefore focusing on port number or any other feature that has to do with source or destination ID will be terribly misleading [50],[92].

3.3.4.3. Feature selection (phase one)

As discussed in the previous sections, separate files are created for each kind of attack, together with normal traffic samples. Each file has 80 features. To get the most important features for every attack type, the Random Forest Regressor class in Sklearn [91] is used. This algorithm generates a decision-forest, and in the decision forest, every feature is weighted based on its usefulness in constructing the decision tree. When the weight setting process finishes, the resulting importance of weights assigned for all features are compared and after that sorted. The grand importance weight of the decision tree is given by the sum of the importance weights of all the features. Comparing the score of individual features to the whole tree provided information regarding the importance of that particular feature in the decision tree. The most important feature for every attack is collected in a CSV file and used in the training machine learning models. The list of the most important features for each attack type is illustrated in appendix.

Before going to features importance calculation phase, three of the features in the dataset: (Destination Port, Protocol, Timestamp) are excluded because they are considered to be misleading features as they are popular features the attackers may dodge.

3.3.4.4. Feature selection (phase two)

In the first phase of feature importance calculation, the most important features of each attack type were found. In the second phase, however, the most important features for the whole dataset combined as attack and normal samples are calculated and stored in a CSV file to be used in training the machine learning models. Figure 3.4.2. shows the twenty most important features and their importance value.

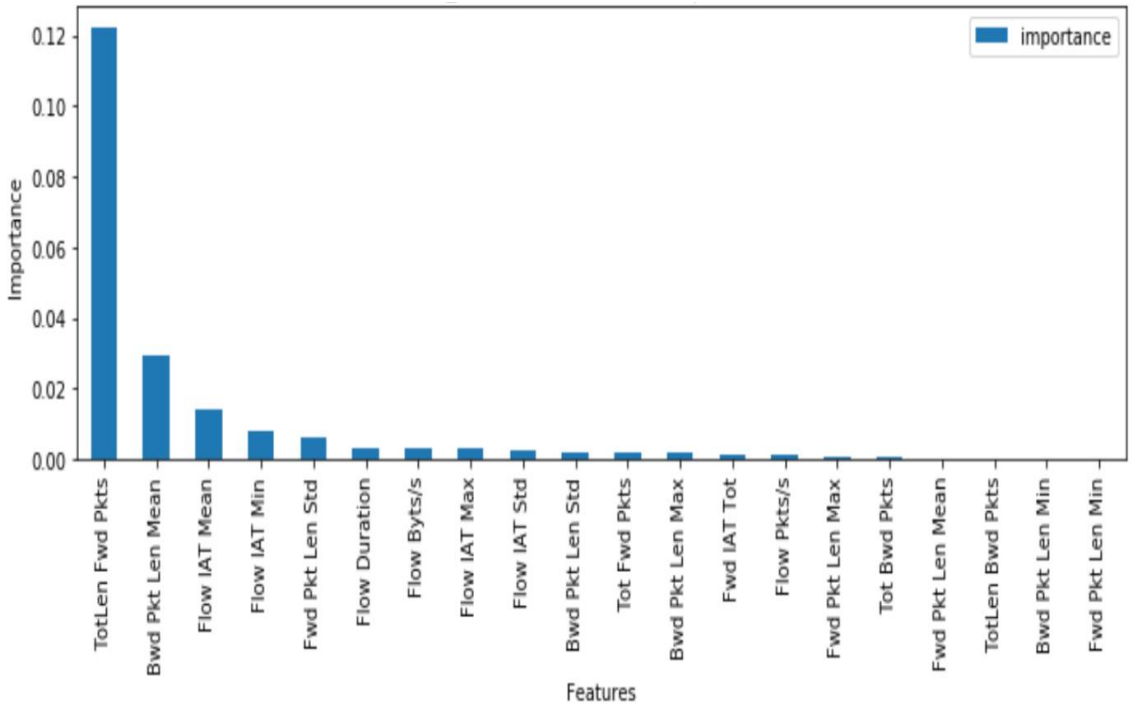

Figure 3.3.2. *Feature importance weights chart (Attack and normal traffic)*

3.3.5. Experiment Implementation

As indicated in the previous sections, the CSE-CIC-IDS2018 dataset contains fourteen different attack types. In the pre-processing stage, separate files for each attack have been created and at the same time, one big file containing only attack and normal traffic labels also created. The processed dataset is fed to eight different machine learning algorithms (KNN, MLP, Random Forest, AdaBoost, Naïve Bayes, Gradient Boosting, ID3 and QDA) ten times in three experiments.

In the first experiments, each specific attack file is applied to the machine learning algorithms using the five most important features of each attack found in the feature importance calculation phase. In the second experiment, however, the big attack-normal dataset file is used in machine learning implementation process with the seventeen most important features shown in Table 3.4.2, from the separate attack files.

In the third experiment, the ten most important features found in the second phase of feature selection shown in Table 3.4.3 are used with the attack-normal dataset in implementing machine learning algorithms to see the response of the model to a minimized feature.

Table 3.3.2. *The 17 features selected for the second experimentation process*

Bwd Pkt Len Mean	Flow Pkts/s
Flow IAT Mean	Flow IAT Min
Fwd Pkt Len Mean	Flow Byts/s
Flow IAT Max	TotLen Fwd Pkts
Tot Bwd Pkts	Fwd Pkt Len Max
Flow Duration	Bwd Pkt Len Std
Fwd IAT Tot	Flow IAT Std
Fwd Pkt Len Std	Bwd Pkt Len Max
Tot Fwd Pkts	

Table 3.3.3. *The 10 features selected for the third experimentation process*

TotLen Fwd Pkts	Bwd Pkt Len Mean
Flow IAT Mean	Flow IAT Min
Fwd Pkt Len Std	Flow Duration
Flow Byts/s	Flow IAT Max
Flow IAT Std	Bwd Pkt Len Std

4. RESULTS AND DİSCUSSİONS

This section depicts results of the experiments discussed on the implementation section.

4.1. Method 1: Individual Attack Classification

As discussed in the previous section, the dataset is grouped in different files for each attack type. Each file (containing specific attack type and normal traffic) is fed to eight different machine learning algorithms using the most important features for each attack type. It is noteworthy that only five (shown in appendix-B) out of eighty features of the dataset are used to detect each attack scenario. Even the most distinctive features like Destination Port and Protocol are removed to train the model under the most aggressive conditions. The results obtained are presented in tabular form in the following tables:

Table 4.1. *Attack detection accuracy levels of different machine learning algorithms*

Attacks	**Machine Learning Algorithms**							
	RF	**KNN**	**ID3**	**NB**	**AdB**	**MLP**	**GrB**	**QDA**
	Accuracy	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**
B-Force -Web	0.93	0.94	0.92	*0.43*	0.94	0.81	0.93	*0.54*
Bot	**0.99**	**1.0**	**1.0**	*0.6*	**1.0**	**1.0**	**1.0**	0.89
B-Force -XSS	0.87	0.81	0.89	*0.77*	0.88	0.81	0.88	0.8
DDOS-HOIC	**0.99**	**0.99**	**0.99**	*0.62*	**0.99**	0.94	**0.99**	0.91
DDOS-LOIC-UDP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.96	**1.0**	**1.0**
DDoS-LOIC-HTTP	**0.99**	**0.99**	**0.99**	*0.39*	**0.99**	*0.74*	**0.99**	*0.63*
Dos-GoldenEye	0.98	0.98	0.98	*0.38*	0.97	*0.7*	0.98	*0.61*
DoS-Hulk	0.98	0.98	0.98	*0.36*	0.98	0.96	0.92	0.81
DoS-SlowHTTPTest	0.98	0.98	0.98	*0.6*	0.98	0.95	0.97	*0.3*

DoS-Slowloris	0.95	**0.99**	**0.99**	*0.35*	**1.0**	0.9	**1.0**	*0.63*
FTP-BruteForce	0.98	0.98	0.98	*0.71*	0.98	0.98	0.98	*0.29*
Infiltration	*0.75*	*0.74*	*0.75*	*0.35*	*0.75*	*0.73*	*0.75*	*0.34*
SQL Injection	0.89	0.84	0.83	*0.59*	0.9	0.68	0.91	*0.64*
SSH-Bruteforce	**0.99**	**0.99**	**0.99**	*0.65*	**0.99**	0.91	**0.99**	*0.78*
Average	**0.947**	**0.943**	**0.947**	***0.557***	**0.953**	**0.862**	**0.949**	***0.655***

As per the information provided in the Table 4.1, all machine learning algorithms other than Naïve Bayes, QDA and MLP performed well above 94% accuracy. AdaBoost and Gradient Boosting algorithms, in particular, have scored an accuracy of around 95% detection accuracy for all attack types. ID3, Random Forest and KNN, on the other hand, achieved an average accuracy of 94.6% where Naïve Bayes, QDA and MLP performed poorly compared to other machine learning models used as Naïve Bayes basically got the lowest accuracy of 55.7%, where QDA and MLP ended up with an accuracy rate of 65.5% and 86.2% respectively.

The data available for training and testing from each attack type is different from one another; this makes the same algorithm have different accuracy rates for different input data. As illustrated in Table 4.1, it is observable that some attack types are easily detected by almost all algorithms used with higher accuracy while in some other attack types, most of the models showed a lower accuracy rate. In general, all models except the three low-performing models discussed before, accurately detected eleven out of fourteen attack types: Brute-Force -Web, Botnet, DDOS (HOIC, LOIC-UDP, LOIC-HTTP), DoS (GoldenEye, Hulk, SlowHTTPTest, Slowloris) and FTP-Brute force. The Infiltration attack is the worst in terms of detection with a maximum 75% accuracy, followed by Brute-force-XSS and SQL injection attacks having around 80% accuracy.

As previously noted, in this study, F1-score is considered the most important indicator for reflecting on the accuracy of the machine learning models used in this thesis. Table 4.2. below shows the F1-Scores of the models used.

Table 4.2. *Attack types and corresponding F1-scores of the machine learning algorithms*

Attacks	**Machine Learning Algorithms**							
	RF	**KNN**	**ID3**	**NB**	**AdB**	**MLP**	**GrB**	**QDA**
	F1-Score	**F1-Score**	**F1-Score**	**F1-Score**	**F1-Score**	**F1-Score**	**F1-Score**	**F1-Score**
B-Force -Web	0.91	0.92	0.89	0.43	0.92	0.72	0.9	0.54
Bot	**0.99**	**1.0**	**1.0**	0.6	**1.0**	**1.0**	**1.0**	0.88
B-Force -XSS	0.86	0.75	0.88	0.71	0.88	0.77	0.88	0.74
DDOS-HOIC	**0.98**	**0.99**	**0.98**	0.62	**0.98**	0.93	**0.98**	0.9
DDOS-LOIC-UDP	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.94	**1.0**	**1.0**
DDoS-LOIC-HTTP	**0.99**	**0.99**	**0.99**	0.38	**0.99**	0.43	**0.99**	0.63
Dos-GoldenEye	0.97	0.98	0.97	0.36	0.97	0.41	0.98	0.61
DoS-Hulk	0.98	0.97	0.98	0.32	0.98	0.95	0.91	0.8
DoS-SlowHTTPTest	0.98	0.98	0.98	0.6	0.98	0.94	0.97	0.23
DoS-Slowloris	0.94	**0.99**	**0.98**	0.31	**1.0**	0.88	**1.0**	0.63
FTP-BruteForce	0.98	0.98	0.98	0.7	0.98	0.98	0.98	0.23
Infiltration	0.57	0.67	0.58	0.32	0.59	0.49	0.62	0.31
SQL Injection	0.88	0.82	0.81	0.59	0.89	0.6	0.9	0.63
SSH-Bruteforce	**0.99**	**0.99**	**0.99**	0.65	**0.99**	0.9	**0.99**	0.77
Average	**0.93**	**0.930**	**0.929**	**0.542**	**0.939**	**0.781**	**0.935**	**0.635**

F1-Score combines recall and precision, hence fully reflects the overall model success goes with the same pattern. The same algorithms presented fairly close results in both cases. Naïve Bayes, QDA and MLP ranked as the lowest-performing algorithms while the remaining five algorithms (Random Forest, AdaBoost, Gradient Boosting, KNN and ID3) resulted an average F1-Score of 93%. Figure 4.1. below illustrates how close the model accuracy and F1-Score are.

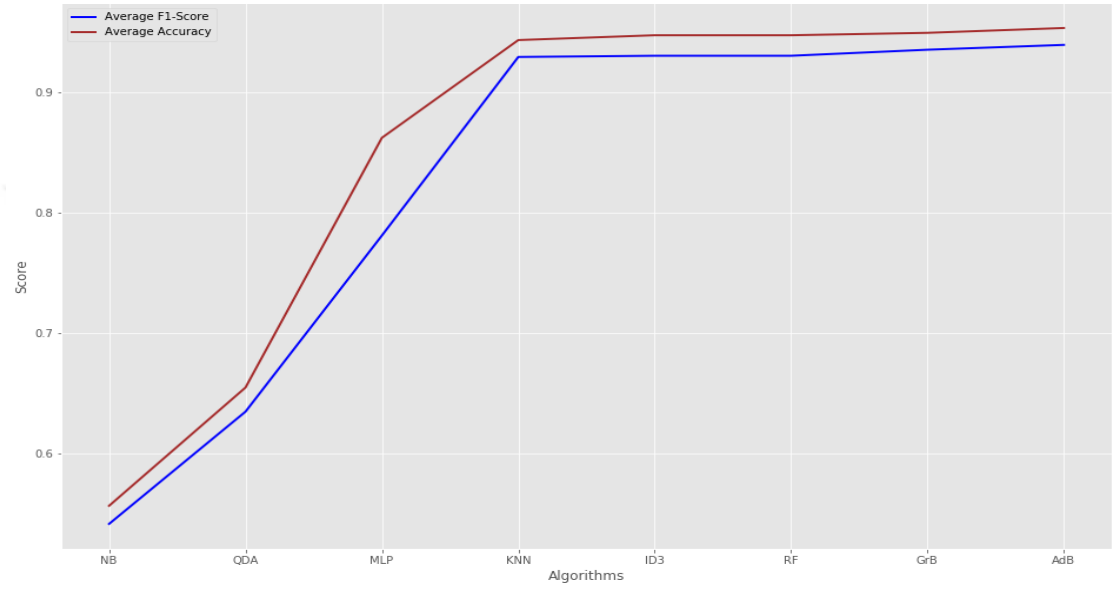

Figure 4.1. *Average F1-Score and accuracy of machine learning algorithms*

Naïve Bayes algorithm has the lowest F1-Score in 10 out of 14 tasks. It is at the same time the fastest algorithm used in this study because it trains very quickly as it needs a single pass on the data for counting frequencies or computing normal probability function. In addition to that, the QDA algorithm achieves results close to that of Naïve Bayes. A common factor of two is that the two algorithms are both statistical machine learning algorithms and for that, they have lower performance than other machine learning algorithms except in the case of DDOS-LOIC-UDP attack where they both have 100% F1-Scores.

MLP is the third-worst performing algorithm used. DoS, DDoS and Infiltration have a reasonably low F1-Score with MLP algorithm. Some studies [93] suggest that there may be a linear relationship between the number of input samples fed to an MLP and its F1-Score. However, that assumption does not seem to be true for all attack types,

attack scenarios with a substantial number of samples like Infiltration, which has 161934 examples and the lowest F1-Score of 49%. Therefore, the number of samples trained with the MLP is not the only factor to decide the F1-Score achieved for a particular case. Figure 4.2. illustrates the F-measures of the MLP algorithm and the number of samples in each attack file. According to the graph, in some instances, F1-Scores change the same way as the number of the input samples and in some other cases, there is no credible relationship between the two quantities.

The performance of MLP is determined by the number of hidden layers used. Fewer neurons can speed up the processing time but also lead to sparse approximation, too many neurons, however, may result in overfitting. Therefore, a suitable selection of the number of neurons for a given MLP is seen as a crucial step towards a better performing and fast model [25]. Tables illustrating the performance of measures each algorithm in attack types are given in the appendices.

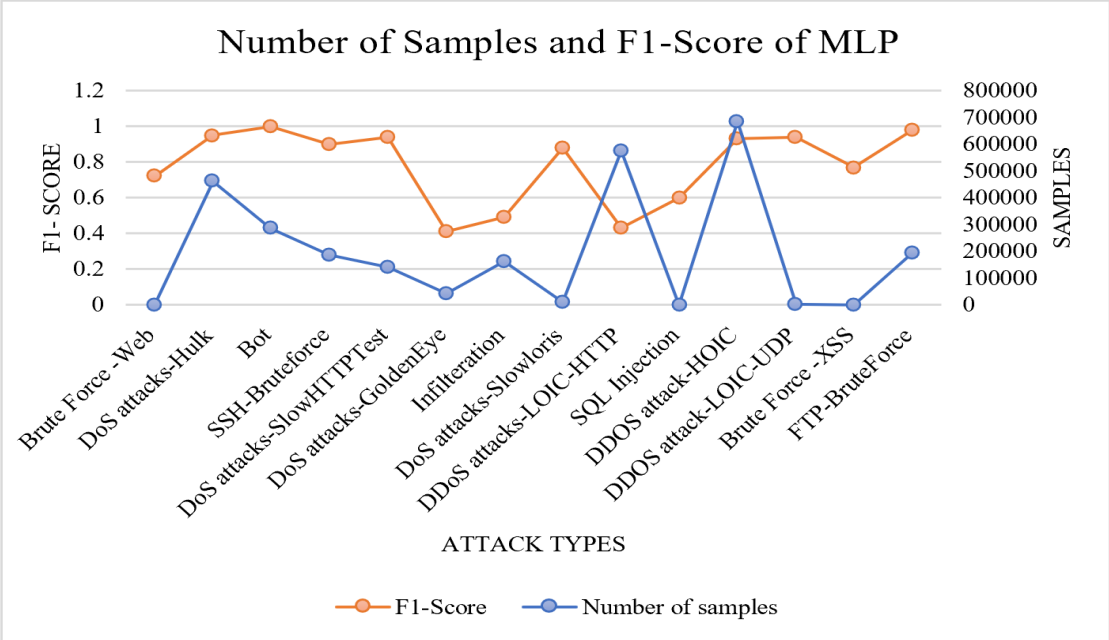

Figure 4.2. *F1-Score of the MLP and the number of samples in the attack files*

4.2. Method 2: Attack and Benign Classification

As per data pre-processing, the entire dataset is combined in a single dataset file, all attacks unified under one label ‘Attack’ and the 'normal' traffic labelled as ‘Benign’.

This big dataset is fed to eight different machine learning algorithms using two different approaches, in the first method, the most critical features extracted for each attack data (used in the previous procedure) are combined and the seventeen most vital of them are used in machine learning implementation phase. In the second approach, the most important features of the unified dataset are extracted and then the ten most important of them are used in the machine learning implementation.

4.2.1. Using Seventeen Features Extracted for Attacks Files

In section 4.1, machine learning implementation on the separate attack files is done with only five crucial features for each file, the important features used to detect each attack type are listed in appendix-B. Some of those features like Flow IAT Max, Flow IAT Mean and Flow Duration are found to be important features for more than two attack types. With the aim of reducing the number of features to train the models, all important features used for separate attacks are combined. After removing the redundant features, a total of seventeen features shown in Table 3.4.2 are compiled.

After collecting all attack and normal traffic data in one file (attack-normal) and labelling all malicious samples as 'Attack' and normal traffic samples as 'Benign', machine learning algorithms are implemented on the attack-normal file by using the seventeen features found from the separate attack files. The results of all machine learning algorithms used in this study are presented in Table 4.3.

Table 4.3. *Performance matrices of the machine learning algorithms using 17 features*

ML Algorithms	F1-Score	Precision	Accuracy	Recall	Time (s)
Random Forest	**0.94**	**0.94**	**0.95**	**0.93**	**125.059**
QDA	0.5	0.67	0.5	0.64	220.8242
ID3	**0.94**	**0.95**	**0.95**	**0.94**	**91.8628**
AdaBoost	0.94	0.94	0.95	0.94	*1516.6172*
Naïve Bayes	*0.36*	*0.62*	*0.38*	*0.55*	*15.9225*
MLP	*0.77*	*0.85*	*0.83*	*0.75*	*2340.7845*
K Nearest Neighbors	0.95	0.95	0.96	0.95	*6781.5838*
Gradient Boosting	**0.95**	**0.95**	**0.95**	**0.94**	**839.9821**

In Table 4.3, five out of eight machine learning algorithms performed well, KNN and Gradient Boosting algorithms, notably outperformed others by having a 95% F1-Score

whereas ID3, Random Forest and AdaBoost achieved 94% F1-Score. However, ID3 and Random Forest are remarkably the fastest of the best-performing algorithms. Naïve Bayes and QDA are quickest and the lowest scoring algorithms at the same time, with 36% and 50%, respectively. KNN, MLP and AdaBoost are very slow algorithms with thousands of seconds needed to train each model.

4.2.2. Using Ten Features Extracted for Attack-Benign File

As noted in the feature selection section, the third step of feature selection is done by applying the Random Forest Regressor to the attack-benign file. In the third phase of the machine learning implementation, however, ten of those features are used to train and test machine learning algorithms, further complicating the model operating conditions. The flowing Table 4.4 shows the performance matrices of all algorithms used and their respective running periods.

Table 4.4. *Performance matrices of the machine learning algorithms using 10 features*

ML Algorithms	F1-Score	Precision	Accuracy	Recall	Time (s)
Random Forest	0.92	0.93	0.94	0.92	120.3651
QDA	*0.38*	*0.64*	*0.4*	*0.57*	*21.6062*
ID3	**0.93**	**0.94**	**0.94**	**0.92**	**67.9749**
AdaBoost	0.93	0.93	0.95	0.94	1096.4063
Naïve Bayes	*0.36*	*0.62*	*0.38*	*0.55*	*12.7424*
MLP	*0.64*	*0.75*	*0.78*	*0.66*	*4637.6427*
K Nearest Neighbors	**0.95**	**0.95**	**0.95**	**0.94**	**5787.5496**
Gradient Boosting	**0.94**	**0.94**	**0.95**	**0.94**	**618.9853**

The data in Table 4.4, the overall performance measure of the machine learning algorithms didn't reduce significantly the same way the number of features reduced from 17 to 10 features as shown in Table 3.4.3. Only the QDA and MLP noticeably reduced; that is because the two algorithms already had low performance and the reduced number of features exacerbated their performances more than it does with other algorithms. Regarding the running times, only QDA has achieved some noticeable reduction in training and testing time, other algorithms slightly reduced training and testing time. The following figures compare the running time and the scores of the algorithms using 17 and 10 features of the dataset.

To better comprehend the impact of the feature reduction on the accuracy of the machine learning algorithms, the accuracy rates achieved with complete dataset features are explored and compared with those attained with reduced number of features. Table below shows the accuracy rates of the machine learning algorithms with and without reduced number of features.

Table 4.5. *Accuracy rate and computation time of the implemented machine learning algorithms*

	Case-1: Using 79 Features		**Case-2: Using 17 Features**	
ML Algorithms	**Accuracy**	**Time (s)**	**Accuracy**	**Time (s)**
Random Forest	**0.982**	**199.029**	**0.95**	**125.051**
QDA	0.610	293.2107	0.50	220.8242
ID3	**0.975**	**167.1140**	**0.95**	**91.8628**
AdaBoost	0.984	*21149.2*	0.95	*1516.6172*
Naïve Bayes	*0.431*	*27.9211*	*0.38*	*15.9225*
MLP	*0.910*	*1982.201*	*0.83*	*2340.7845*
K Nearest Neighbors	0.980	*8156.10*	0.96	*6781.5838*
Gradient Boosting	**0.975**	**16820.3**	**0.95**	**839.9821**
Average	**0.855875**	**6099.384**	**0.80875**	**1491.579**

As seen in Table 4.5, extraneous features substantially increase computation time of the algorithms, and at the same time can have an influence on the accuracy of the IDS. Reducing the number of features by 78.5% resulted an average 75.5% reduction in computational time and 5% reduction in the machine learning algorithms' average accuracy rates. For a real-time detection scenario, the running time of the IDS is very crucial making the feature reduction step super important.

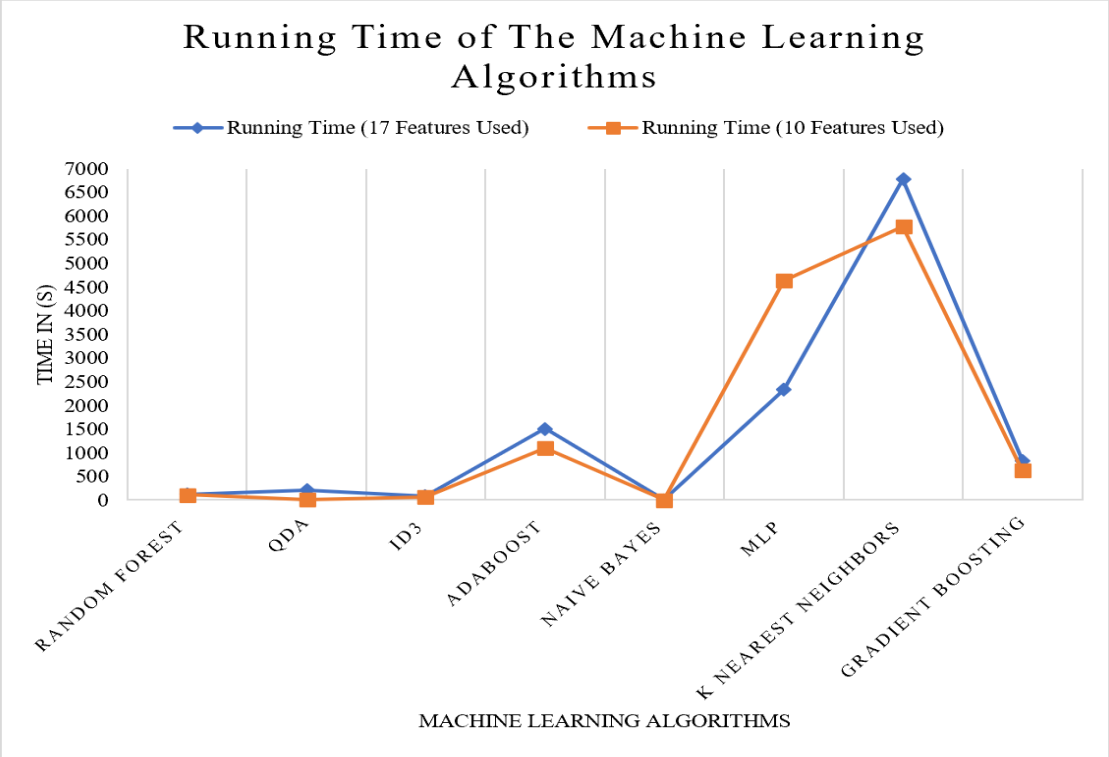

Figure 4.3. *Running time of the machine learning algorithms running with 10 and 17 features*

Figure 4.4. below, shows how the F1-Scores of the machine learning algorithms change with the change in the number of features used.

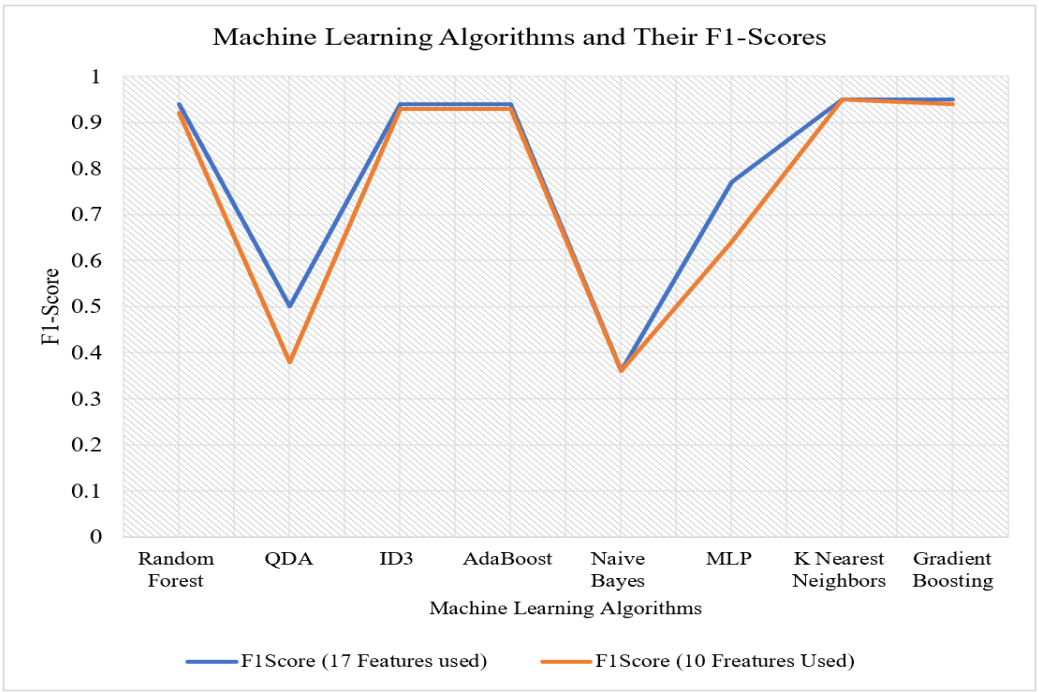

Figure 4.4. *F1-Scores of the machine learning algorithms running with 10 and 17 features*

4.3. Discussions

In case machine learning algorithms are trained with all features in the dataset, the accuracy rate of all machine learning algorithms except MLP and Naïve Bayes approach 100%, but the computational time of the algorithms becomes long hence, the need for higher computational power becomes clear as the process drains all computational resources of a normal personal computer.

It is evident from the results that particular classes are easier to be classified than others. FTP-Brute force, Dos, DDoS and Botnet classes contribute significantly to the attack samples in the dataset; this led to high-performance rates of above 98% on all performance measures, even with a significantly reduced number of features (5 features for each attack). The results of the machine learning algorithms on the two other attack types indicate some different aspects. Web attacks are considered to be underrepresented in CSE-CIC-IDS2018, accounting for around 880 samples only. As a result, the outcome is not as impressive as it is with the other attack classes. The low performance of approximately 50% F1-score experienced in detecting Infiltration traffic can be seen as a very absurd case because this class is well represented in the dataset with 161934 samples representing Infiltration attack.

According to the documentation of the dataset, unrestricted file upload is taken as a part of the Infiltration attack scenario, hence, to distinguish between illegitimate and legitimate file upload traffic at the network level becomes hard.

ID3 and Random Forest are found to be the best performing algorithms with a 94% accuracy in less than 2 minutes. Gradient boosted and AdaBoost achieved good results by getting impressive results from under-represented web attacks, which leaves Infiltration to be the hard-to-detect attack scenario. It takes an average of 15 minutes to complete the classification task.

KNN successfully achieves an excellent 94% F1-score under the hardest conditions, but as it is known to be one of the instance-learning algorithms, it is a lazy-learning algorithm [25]. Therefore, it took more than an hour and a half to train on the same dataset the ID3 algorithm trained within less than two minutes. In addition to that, dropping the most discriminative features negatively affects the detection performance on web attack and Infiltration traffic and slightly increases the running time.

Naïve Bayes and QDA algorithms are found to be the fastest algorithms, but their deficient performance (less than 50%) prevents them from being fully appreciated.

MLP is a bit better in terms of performance, but the second slowest algorithm used in this study.

4.4. Evaluation

In this section, the final results achieved in this study are compared with the research conducted by Lin et al. [78] and noted in the literature review section. Several machine learning algorithms used in this thesis are also presented in the selected research paper. The main difference between the cited research and this thesis is that there are no feature reduction procedures conducted in the article cited, as in this thesis, where several feature reduction steps are done. The results used for the evaluation are the output of the second experiment of this thesis. Precision, Accuracy and Recall are used as the evaluation criteria as preferred by Lin et al. [78].

Table 4.6. *Performance comparison of two studies against the evaluation criteria*

	Our Study (17 features used)			**Lin et al [78] (78 features used)**		
	Accuracy	*Recall*	*Precision*	*Accuracy*	*Recall*	*Precision*
Decision tree	***0.940***	***0.940***	***0.950***	0.938	0.93	0.93
Gaussian NB	0.360	0.550	***0.620***	***0.554***	0.55	0.66
Random forest	0.940	0.930	0.940	***0.942***	***0.94***	0.94
KNN	***0.950***	***0.950***	***0.950***	0.942	0.94	0.94

Table 4.6 shows the results achieved by the machine learning algorithms in this thesis and that of Lin et al [78]. Using only 17 features, this thesis study achieved overall similar accuracy rates as that of Lin et al. which is using 78 features (only two features removed from the 80 features of the original dataset). The research conducted by Lin et al. also presented a Deep Learning model with the same dataset and achieved an average 96.2% accuracy. The reduction of the number of features affected the Naïve Bayes algorithm, at the same time, it helped the KNN algorithm accuracy improvement. In addition to that, the AdaBoost and Gradient boosting algorithms which were not used in the other research performed the best in the machine learning scoop.

Reducing the number of features from 80 to 17 features reduces the computation time by 25%. This comes with subsequently comes with 5% reduction in the accuracy of machine learning algorithms.

5. CONCLUSİON AND FUTURE WORK

5.1. Conclusion

In this study, eight different machine learning models (ID3, Random Forest, MLP, KNN, Naïve Bayes, QDA, AdaBoost and XGboost) were implemented on the latest IDS dataset (CSE-CIC-IDS2018 dataset) to detect network anomalies by recognizing attack traffic from normal network traffic. The CSE-CIC-IDS2018 dataset used contains 80 features explaining the network traffic flow. At the first step of the experimentation process, the dataset is cleaned of minor errors. Feature importance weights are calculated with the Random Forest Regressor [91] algorithm to ascertain discriminative features to be used to train machine learning models. The feature selection is made in two ways. In the first method, feature importance weights for each attack are calculated separately. In the second step, the important features of the big attack-benign dataset file are calculated, making sure that the important and characteristic features of all attack types are attained. Finally, eight popular and diverse machine learning algorithms have been applied to the final dataset in three different experiments. Five of the eight algorithms (ID3, Random forest, KNN. AdaBoosta and XGboost) used achieved around 95% F1-Scores.

5.2. Future Work

In this study, a dataset in the form of CSV files with features extracted from the network flow is used. However, to make the detection more realistic, a model with the capability to retrieve and process real network data into a format effective with the machine learning algorithm is needed. Individual machine learning methods are used in this study, combining two or more of those methods can improve detection accuracy. Since the dataset available in this study is a big dataset, using personal computers may not be the best computing platform to be used. Machines with high computational power may be used or cloud computing platforms, and in doing so, more sophisticated techniques like Deep learning can be deployed to increase attack detection efficiency by taking advantage of robust platforms like TensorFlow, Amazon Machine Learning (AML) and Google Cloud ML Engine.

REFERENCES

[1] "Statistics." [Online]. Available: https://www.itu.int/en/ITUD/Statistics/Pages/stat/default.aspx. [Accessed: 29-May-2020].

[2] A. a Ghorbani, W. Lu, and M. Tavallaee, "Network Intrusion Detection and Prevention," *Inf. Secur.*, 2010, doi: 10.1007/978-0-387-88771-5.

[3] J. Kurose and W. Keith, *K. Ross Computer networking: a top down approach*. 2007.

[4] Symantec Corporation, "Symantec Internet Security Threat Report," 2019.

[5] S. Latha and S. J. Prakash, "A survey on network attacks and Intrusion detection systems," in *2017 4th International Conference on Advanced Computing and Communication Systems, ICACCS 2017*, 2017, doi: 10.1109/ICACCS.2017.8014614.

[6] "Which is better: anomaly-based IDS or signature-based IDS?" [Online]. Available: https://searchsecurity.techtarget.com/tip/IDS-Signature-versus-anomaly-detection. [Accessed: 25-Feb-2020].

[7] "Network Security Threat and Solutions." [Online]. Available: https://www.computernetworkingnotes.com/ccna-study-guide/network-security-threat-and-solutions.html. [Accessed: 20-Mar-2020].

[8] Q. Wang and Z. L. Piao, "Research on network attack and detection methods," in *2nd International Workshop on Education Technology and Computer Science, ETCS 2010*, 2010, doi: 10.1109/ETCS.2010.196.

[9] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," *Comput. Secur.*, 2005, doi: 10.1016/j.cose.2004.06.011.

[10] D. E. Denning, "An Intrusion-Detection Model," *IEEE Trans. Softw. Eng.*, 1987, doi: 10.1109/TSE.1987.232894.

[11] H. Tun, S. Lupin, H. H. Linn, and K. N. Z. Lin, "Selection the perimeter protection equipment in security systems," in *Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus*

2018, 2018, doi: 10.1109/EIConRus.2018.8317383.

[12] "Defense in Depth." [Online]. Available: https://apps.nsa.gov/iaarchive/library/ia-guidance/archive/defense-in-depth.cfm. [Accessed: 17-May-2020].

[13] A. Niemelä, "Traffic Analysis for Intrusion Detection in Telecommunications Networks," no. December, 2010.

[14] "SANS Institute: Reading Room - Intrusion Detection." [Online]. Available: https://www.sans.org/reading-room/whitepapers/detection/paper/366. [Accessed: 18-May-2020].

[15] M. Roesch and S. Telecommunications, "Snort - LightWeight Instrution Detection For Networks," *Proc. LISA '99 13th Syst. Adm. Conf.*, 1999, doi: 10.1.1.105.6212.

[16] "Exploring Snort - Cisco Meraki Blog." [Online]. Available: https://meraki.cisco.com/blog/2019/06/exploring-snort/. [Accessed: 18-May-2020].

[17] H. N. Tran, "A Dynamic Scalable Parallel Network-based Intrusion Detection System using Intelligent Rule Ordering," 2017.

[18] "Introduction — Bro 2.5.5 documentation." [Online]. Available: https://old.zeek.org/manual/2.5.5/intro/index.html. [Accessed: 18-May-2020].

[19] E. Alpaydin, "Introduction to Machine Learning Ethem Alpaydin.," *Introd. to Mach. Learn. Third Ed.*, 2014.

[20] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. 2019.

[21] B. Cronin, *Henrik Brink Joseph W. Richards Mark Fetherolf - Machine Learning*. 2017.

[22] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences," *Atmos. Environ.*, 1998, doi: 10.1016/S1352-2310(97)00447-0.

[23] R. Kohavi, "Scaling Up the Accuracy of Naïve -Bayes Classifiers: A Decision-Tree Hybrid," *Proc. Second Int. Conf. Knowl. Discov. Data Min.*, 1996, doi:

citeulike-article-id:3157868.

[24] M. F. A. Saputra, T. Widiyaningtyas, and A. P. Wibawa, “Illiteracy Classification Using K Means-Naïve Bayes Algorithm,” *JOIV Int. J. Informatics Vis.*, 2018, doi: 10.30630/joiv.2.3.129.

[25] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, “Supervised Machine Learning : A Review of Classification Techniques General Issues of Supervised Learning Algorithms,” *Inform.*, 2007.

[26] H. Ming, N. Wenying, and L. Xu, “An improved decision tree classification algorithm based on ID3 and the application in score analysis,” in *2009 Chinese Control and Decision Conference, CCDC 2009*, 2009, doi: 10.1109/CCDC.2009.5192865.

[27] T. K. Ho, “The random subspace method for constructing decision forests,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 1998, doi: 10.1109/34.709601.

[28] A. Natekin and A. Knoll, “Gradient boosting machines, a tutorial,” *Front. Neurorobot.*, vol. 7, no. DEC, p. 21, Dec. 2013, doi: 10.3389/fnbot.2013.00021.

[29] L. Breiman, “Random forests,” *Mach. Learn.*, 2001, doi: 10.1023/A:1010933404324.

[30] G. Bonaccorso, *Machine Learning Algorithms: Reference guide for popular algorithms for data science and machine learning*. 2017.

[31] S. Marsland, *Machine learning: An algorithmic perspective*. 2014.

[32] “ML Project.” [Online]. Available: http://people.irisa.fr/Elisa.Fromont/ML/MLProject.html. [Accessed: 12-May-2020].

[33] “1.2. Linear and Quadratic Discriminant Analysis — scikit-learn 0.22.2 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/lda_qda.html. [Accessed: 12-May-2020].

[34] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “An Evaluation Framework for Intrusion Detection Dataset,” in *ICISS 2016 - 2016 International Conference on Information Science and Security*, 2017, doi:

10.1109/ICISSEC.2016.7885840.

[35] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: techniques, datasets and challenges,” *Cybersecurity*, 2019, doi: 10.1186/s42400-019-0038-7.

[36] G. Creech and J. Hu, “A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns,” *IEEE Trans. Comput.*, 2014, doi: 10.1109/TC.2013.13.

[37] “KDD-CUP-99 Task Description.” [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/task.html. [Accessed: 30-Apr-2020].

[38] A. Alazab, M. Hobbs, J. Abawajy, and M. Alazab, “Using feature selection for intrusion detection system,” in *2012 International Symposium on Communications and Information Technologies, ISCIT 2012*, 2012, doi: 10.1109/ISCIT.2012.6380910.

[39] 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, CISDA 2009*, 2009, doi: 10.1109/CISDA.2009.5356528.

[40] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” *Comput. Secur.*, 2012, doi: 10.1016/j.cose.2011.12.012.

[41] “Applications | Research | Canadian Institute for Cybersecurity | UNB.” [Online]. Available: https://www.unb.ca/cic/research/applications.html#CICFlowMeter. [Accessed: 25-Apr-2020].

[42] “IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB.” [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html. [Accessed: 01-May-2020].

[43] “IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB.” [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html. [Accessed: 27-Feb-2020].

[44] “What Is a Brute Force Attack? | Kaspersky.” [Online]. Available: https://www.kaspersky.com/resource-center/definitions/brute-force-attack. [Accessed: 18-Apr-2020].

[45] M. M. Najafabadi, T. M. Khoshgoftaar, C. Kemp, N. Seliya, and R. Zuech, “Machine learning for detecting brute force attacks at the network level,” in *Proceedings - IEEE 14th International Conference on Bioinformatics and Bioengineering, BIBE 2014*, 2014, doi: 10.1109/BIBE.2014.73.

[46] J. Mirkovic and P. Reiher, “A taxonomy of DDoS attack and DDoS defense mechanisms,” *Computer Communication Review*. 2004, doi: 10.1145/997150.997156.

[47] M. Kumagai, Y. Musashi, D. A. L. Romaña, K. Takemori, S. Kubota, and K. Sugitani, “SSH dictionary attack and DNS reverse resolution traffic in campus network,” in *Proceedings - 3rd International Conference on Intelligent Networks and Intelligent Systems, ICINIS 2010*, 2010, doi: 10.1109/ICINIS.2010.9.

[48] “GitHub - lanjelot/patator: Patator is a multi-purpose brute-forcer, with a modular design and a flexible usage.” [Online]. Available: https://github.com/lanjelot/patator. [Accessed: 19-Apr-2020].

[49] D. J. Barrett and R. Silverman, *SSH, The Secure Shell - The Definitive Guide*, First Edit. O’Reilly Media, 2001.

[50] R. Alshammari and A. Nur Zincir-Heywood, “A flow based approach for SSH traffic detection,” in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, 2007, doi: 10.1109/ICSMC.2007.4414006.

[51] M. M. Najafabadi, T. M. Khoshgoftaar, C. Calvert, and C. Kemp, “Detection of SSH brute force attacks using aggregated netflow data,” in *Proceedings - 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA 2015*, 2016, doi: 10.1109/ICMLA.2015.20.

[52] P. Likarish, E. Jung, and I. Jo, “Obfuscated malicious javascript detection using classification techniques,” in *2009 4th International Conference on Malicious and Unwanted Software, MALWARE 2009*, 2009, doi: 10.1109/MALWARE.2009.5403020.

[53] “CWE - Vulnerability Type Distributions in CVE.” [Online]. Available: http://cwe.mitre.org/documents/vuln-trends/index.html. [Accessed: 22-Apr-2020].

[54] E. Alomari, S. Manickam, B. B. Gupta, S. Karuppayah, and R. Alfaris, “Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers: Classification and Art,” *Int. J. Comput. Appl.*, 2012, doi: 10.5120/7640-0724.

[55] M. Feily, A. Shahrestani, and S. Ramadass, “A survey of botnet and botnet detection,” in *Proceedings - 2009 3rd International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2009*, 2009, doi: 10.1109/SECURWARE.2009.48.

[56] A. H. Lashkari, G. D. Gil, J. E. Keenan, K. F. Mbah, and A. A. Ghorbani, “A survey leading to a new evaluation framework for networkbased botnet detection,” in *ACM International Conference Proceeding Series*, 2017, doi: 10.1145/3163058.3163059.

[57] A. Sperotto and A. Pras, “Flow-based intrusion detection,” in *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management, IM 2011*, 2011, doi: 10.1109/INM.2011.5990529.

[58] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai, “Denial-of-service attack-detection techniques,” *IEEE Internet Comput.*, 2006, doi: 10.1109/MIC.2006.5.

[59] S. Behal and K. Kumar, “Characterization and comparison of DDoS attack tools and traffic generators - a review,” *Int. J. Netw. Secur.*, 2017, doi: 10.6633/IJNS.201703.19(3).07.

[60] “GitHub - jseidl/GoldenEye: GoldenEye Layer 7 (KeepAlive+NoCache) DoS Test Tool.” [Online]. Available: https://github.com/jseidl/GoldenEye. [Accessed: 21-Apr-2020].

[61] S. T. Zargar, J. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks,” *IEEE Commun. Surv. Tutorials*, 2013, doi: 10.1109/SURV.2013.031413.00127.

[62] “How Every Cyber Attack Works - A Full List.” [Online]. Available:

https://heimdalsecurity.com/blog/cyber-attack/#. [Accessed: 02-May-2020].

[63] "GitHub - llaera/slowloris.pl: A new DOS Perl Programm." [Online]. Available: https://github.com/llaera/slowloris.pl. [Accessed: 21-Apr-2020].

[64] "Tag: slow http attack | Qualys Blog." [Online]. Available: https://blog.qualys.com/tag/slow-http-attack. [Accessed: 22-Apr-2020].

[65] "GitHub - NewEraCracker/LOIC: Low Orbit Ion Cannon - An open source network stress tool, written in C#. Based on Praetox's LOIC project. USE ON YOUR OWN RISK. WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES." [Online]. Available: https://github.com/NewEraCracker/LOIC/. [Accessed: 23-Apr-2020].

[66] R. Papadie and I. Apostol, "Analyzing websites protection mechanisms against DDoS attacks," in *Proceedings of the 9th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2017*, 2017, doi: 10.1109/ECAI.2017.8166454.

[67] J. Fonseca, M. Vieira, and H. Madeira, "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks," in *Proceedings - 13th Pacific Rim International Symposium on Dependable Computing, PRDC 2007*, 2007, doi: 10.1109/PRDC.2007.63.

[68] "Perform a Web Security Audit | Acunetix." [Online]. Available: https://www.acunetix.com/security-audit/. [Accessed: 22-Apr-2020].

[69] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, doi: 10.5220/0006639801080116.

[70] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Comput. Secur.*, 2005, doi: 10.1016/j.cose.2004.09.008.

[71] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *VLDB J.*, 2007, doi:

10.1007/s00778-006-0002-5.

[72] W. Hu, W. Hu, and S. Maybank, “AdaBoost-based algorithm for network intrusion detection,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, 2008, doi: 10.1109/TSMCB.2007.914695.

[73] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, “An efficient intrusion detection system based on support vector machines and gradually feature removal method,” *Expert Syst. Appl.*, 2012, doi: 10.1016/j.eswa.2011.07.032.

[74] W. Yassin, N. I. Udzir, and Z. Muda, “Anomaly-based Intrusion Detection Through K- Means Clustering and Naïve Bayes Classification,” in *Proceedings of the 4th International Conference on Computing and Informatics, ICOCI 2013*, 2013.

[75] V. Kanimozhi and T. P. Jacob, “Artificial Intelligence based Network Intrusion Detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing,” *ICT Express*, 2019, doi: 10.1016/j.icte.2019.03.003.

[76] “[1905.03685] Evaluation of Machine Learning Classifiers for Zero-Day Intrusion Detection -- An Analysis on CIC-AWS-2018 dataset.” [Online]. Available: https://arxiv.org/abs/1905.03685. [Accessed: 16-May-2020].

[77] T. Chadza, K. G. Kyriakopoulos, and S. Lambotharan, “Contemporary Sequential Network Attacks Prediction using Hidden Markov Model,” in *2019 17th International Conference on Privacy, Security and Trust, PST 2019 - Proceedings*, 2019, doi: 10.1109/PST47121.2019.8949035.

[78] P. Lin, K. Ye, and C. Z. Xu, “Dynamic network anomaly detection system by using deep learning techniques,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, doi: 10.1007/978-3-030-23502-4_12.

[79] “What is Python? Executive Summary | Python.org.” [Online]. Available: https://www.python.org/doc/essays/blurb/. [Accessed: 20-May-2020].

[80] “scikit-learn: machine learning in Python — scikit-learn 0.23.1 documentation.”

[Online]. Available: https://scikit-learn.org/stable/. [Accessed: 20-May-2020].

[81] “pandas - Python Data Analysis Library.” [Online]. Available: https://pandas.pydata.org/. [Accessed: 20-May-2020].

[82] “Overview — Matplotlib 3.2.1 documentation.” [Online]. Available: https://matplotlib.org/3.2.1/contents.html. [Accessed: 20-May-2020].

[83] “NumPy — NumPy.” [Online]. Available: https://numpy.org/. [Accessed: 20-May-2020].

[84] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Network anomaly detection: Methods, systems and tools,” *IEEE Commun. Surv. Tutorials*, 2014, doi: 10.1109/SURV.2013.052213.00046.

[85] S. Raschka, *Python Machine Learning Unlock*, Illustrate. BIRMINGHAM: Packt Publishing, 2014.

[86] J. VanderPlas, *Python Data Science Handbook*. 2016.

[87] R. B. Basnet, R. Shash, C. Johnson, L. Walgren, and T. Doleck, “Towards detecting and classifying network intrusion traffic using deep learning frameworks,” *J. Internet Serv. Inf. Secur.*, 2019, doi: 10.22667/JISIS.2019.11.30.001.

[88] “sklearn.model_selection.train_test_split — scikit-learn 0.22.1 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. [Accessed: 28-Feb-2020].

[89] C. F. Tsai, Y. F. Hsu, C. Y. Lin, and W. Y. Lin, “Intrusion detection by machine learning: A review,” *Expert Systems with Applications*. 2009, doi: 10.1016/j.eswa.2009.05.029.

[90] “Feature Selection Using Random forest - Towards Data Science.” [Online]. Available: https://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f. [Accessed: 28-Feb-2020].

[91] “3.2.4.3.2. sklearn.ensemble.RandomForestRegressor — scikit-learn 0.22.1 documentation.” [Online]. Available: https://scikit