T.C.

ISTANBUL ALTINBAS UNIVERSITESI

GRADUATE SCHOOL OF SCIENCES ENGINEERING

INTRUSION DETECTION MODEL BASED ON DATA MINING AND MACHINE LEARNING

Khalid Abdulwahid Kadhim

Master of Information Technology

Thesis Supervisor

Asst. Prof. Dr. Oğuz Ata

Istanbul (2018)

INTRUSION DETECTION MODEL BASED ON DATA MINING AND MACHINE LEARNING

by

Khalid Abdulwahid Kadhim KADHIM

Information Technology

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2018

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Oğuz ATA

Supervisor

Examining Committee Members (first name belongs to the chairperson of the jury and the second name belongs to supervisor)

Assoc. Prof. Dr. Metin ZONTUL	Software Engineering, Istanbul Aydin University	
Asst. Prof. Dr. Oğuz ATA	Software Engineering, Istanbul Altinbaş University	
Assoc. Prof. Dr. Oğuz BAYAT	Electrical Engineering, Istanbul Altinbaş University	

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Oğuz ATA

Head of Department

Assoc. Prof. Dr. Oğuz BAYAT

Director

Approval Date of Graduate School of

Science and Engineering: ____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Khalid Abdulwahid Kadhim KADHIM

Signature

DEDICATION

I would like to dedicate this work to my very first teacher, my mother, my first supporter and role model, my father and my companion throughout the journey, my wife. Without you, this dream would never come true and my brothers and my sisters.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to all the instructors that have taught me more than just science, especially my supervisor, Asst. Prof. Dr. Oğuz Ata, for all the time, support and guidance provided to me along the journey to accomplish this work. Thank you all for all the knowledge and advice that made me overcome all the difficulties that I have faces.

ABSTRACT

INTRUSION DETECTION MODEL BASED ON DATA MINING AND MACHINE LEARNING

KADHIM, Khalid Abdulwahid Kadhim,

M.S., Information Technology, Altınbaş University

Supervisor: Asst. Prof. Dr. Oğuz Ata

Date: May/2018

Pages: 71

Recently, the use of online services has grown rapidly, which imposes the need to protect servers that provide these services without affecting the quality of these services. Traditional network protection techniques are no longer applicable, according to the development of the intrusion techniques being used by intruders. Thus, more complex techniques are being used to provide better protection to these networks. Data mining is one of the machine learning fields that can be used to extract relations between packets information and the labels given to them. Thus, in this study, three different data mining classification techniques, which are the Support Vector Machine, Random Forest and Feed-Forward Neural Networks are evaluated to detect anomalies in the packets incoming to the network. Then, the type of attack being executed is also detected by these classifiers, in case an intrusion is detected.

The results show that the feed-forward deep neural network classifier, with only three hidden layers of 32 neurons each, has the best overall performance with a predictions accuracy of 99.27% in binary classification with an average prediction time of 0.7 *u*Sec per each prediction, while the Random forest classifier, with 100 trees in the forest, has scored an accuracy of 99.60% but consumes an average of 8.54 *u*Sec per each prediction, which is extremely high time compared to the deep learning model. Moreover, the support vector machine classifier has scored an accuracy of 98.70% and an average execution time of 218.3 *u*Sec per each prediction.

Moreover, in multi-class classification, the deep learning model with the same hidden layers has shown the best prediction accuracy and time with 90.82% accuracy and 0.89 *u*Sec average prediction time, while the random forest classifier achieved an accuracy of only 87.92% consuming an average of 17.28 *u*Sec per prediction and the support vector machine classifier has a prediction accuracy of 70.43% and consumes an average of 709.65 *u*Sec per prediction. These results show that the feed-forward deep neural network is the best choice to be employed in an intrusion detection system.

Keywords: Network Security; Intrusion Detection System; Data Mining; Anomaly Detection.

ÖZET

VERİ MADENCİLİĞİ VE MAKİNE ÖĞRENME TEMELLİ SALDIRI TESPİT MODELİ

Khalid Abdulwahid Kadhim, KADHIM,
M.S., Bilgi Teknolojisi, Altınbaş Üniversitesi
Danışman: Yrd. Dr. Oğuz Ata
Tarih: Mayıs / 2018
Sayfalar: 71

Son zamanlarda, çevrimiçi hizmetlerin kullanımı hızla artmıştır, bu da bu hizmetlerin kalitesini etkilemeden bu hizmetleri sağlayan sunucuları koruma ihtiyacını doğurmaktadır.
İzinsiz kullanıcıların kullandıkları saldırı tekniklerinin gelişim göstermesinden sonra, geleneksel ağ koruma teknikleri artık yeterli olmamaktadır. Böylece, ağlarda daha iyi koruma sağlamak için daha karmaşık teknikler kullanılması gerekmektedir. Veri madenciliği (Data mining), paket bilgileri ve bunlara verilen etiketler arasındaki ilişkileri çıkarmak için kullanılabilecek makine öğrenme alanlarından biridir.
Bu yüzden, bu çalışmada, Destek Vektör Makinesi(Support Vector Machine), Rastgele Orman(Random Forest) ve İleri Beslemeli Sinir Ağları(Feed-forward Neural Networks) olmak üzere üç farklı veri madenciliği sınıflandırma tekniği, ağa gelen paketlerdeki anormallikleri tespit etmek için kullanılmaktadır. Ayrıca, yapılan saldırı türü de bir saldırı tespit edildiğinde bu sınıflandırıcılar tarafından da algılanır.
Sonuçlar, her bir gizli katmanında 32 nöron bulunan İleri Beslemeli Sinir Ağları'nın, her bir tahmin için ortalama tahmin süresi olarak 0.7 uSec harcayarak %99.27'lik bir tahmin doğruluğu ile en iyi genel performansa sahip olduğunu göstermektedir. Aynı zamanda 100 ağaçlı Rastgele Orman sınıflandırıcısı, % 99.60'lık bir doğruluk elde ederken, her bir tahmin başına ortalama 8.54 uSec tüketir ve bu da derin öğrenme modeline kıyasla çok yüksek bir zamandır. Ayrıca, Destek Vektör Makine sınıflandırıcısı, her bir tahmin için %98.70'lik bir doğruluk ve 218.3 uSec'lik bir ortalama yürütme süresine sahip olmuştur.

Son olarak, Çok Sınıflı Sınıflandırmada, aynı gizli katmanlara sahip derin öğrenme modeli, en iyi tahmin doğruluğunu ve zamanını % 90.82 doğruluk ve 0.89 uSec ortalama tahmin süresi ile gösterirken, Rastgele orman sınıflandırıcısı sadece %87.92 başarı oranı ve tahmin başına ortalama 17.28 uSec ve Destek Vektör Makine sınıflandırıcısı %70.43'lük bir doğruluk oranı ve tahmin başına ortalama 709,65 uSec tüketmektedir.

Bu sonuçlar, İleri Beslemeli Derin Sinir Ağının bir saldırı tespit sisteminde kullanılacak en iyi seçim olduğunu göstermektedir.

Anahtar Kelimeler: Ağ Güvenliği; Saldırı tespit sistemi; Veri madenciliği; Anomali tespiti.

TABLE OF CONTENTS

Pages

LIST OF ABBREVIATIONS

DoS	: Denial of Service
U2R	: User to Root
R2L	: Remote to Local
IDS	: Intrusion Detection System
HIDS	: Host Based Intrusion Detection System
NSM	: Network Security Monitor
DIDS	: Distributed Intrusion Detection System
NADIR	: Network Anomaly Detection and Intrusion Reporter
CIDF	: Common Intrusion Detection Framework
SVM	: Support Vector Machine
RF	: Random Forests
FAR	: False Alarm Rate
DL	: Deep Learning
GPU	: Graphical Processing Unit

LIST OF TABLES

Pages

LIST OF FIGURES

Pages

1. INTRODUCTION

Network security is one of the main concern that has been raising recently, according to the importance of the services being provided over these networks and the development in the techniques used to attack these networks [1]. Traditional techniques that are used to protect networks, such as IP and port filtering, are no longer applicable because of this development, where it has become difficult to distinguish between packets coming from attack traffic and those coming from normal traffic coming from legitimate users trying to access the services provided on the network [2]. Thus, it has become important to use more sophisticated techniques to protect these networks from any intrusion attempts to protect information stored in the servers in the networks and maintain the quality of the services provided over these networks.

There are different types of intrusions that have different goals on networks, where some of these intrusions aim to compromise the security of the information stored on a server, or being transferred from one point on the network to another, and other intrusions aim to degrade the quality of the services provided over the network, to affect the reputation of the facility providing these techniques. To achieve such intrusions, attacks of two main categories are used, which are the passive and active attacks. In a passive attack, the intruder does not interact with any of the partner exchanging information, however, the intruder monitors and analyzes the data flowing in the network in order to extract the information being exchanged by the communicating hosts. Moreover, active attacks use techniques that interact with the connections, by sending, receiving or modifying the information being sent over the connection, in order to gain this information, or access to services that the intruder is not entitled to gain [3].

Machine learning is widely used in different application, where computers are used to interact with the environment by extracting the required knowledge from sample inputs in order to use this knowledge interacting with new inputs, depending on the application that the machine learning technique is used for. Data mining is one of the most important machine learning fields, where knowledge is extracted from datasets by investigating relations among the objects and attributes in the dataset. Data mining techniques extract knowledge that may be difficult for humans to detect, according to the enormous number of values in the dataset, or because of the complex relation that joins these values, which makes it impossible to be detected without the aid of data mining techniques [4].

As any other machine learning techniques, data mining techniques are divided into two main categories, which are the supervised and unsupervised techniques. The data used by data mining learning techniques must include extra information added by an expert human, which are known as labels, while the unsupervised data mining techniques require no additional knowledge to be added to the dataset. Classification techniques are supervised machine learning techniques that are widely used in different applications, where these techniques extract the relation between the values the characterize each object and the label given to that object, so that, this knowledge can be used to predict labels for future unlabeled objects without the need of human expertise. These predictions can be used to estimate the behavior of the object depending on the behavior that objects share the same label [5].

To detect an anomaly in the network traffic incoming to the network, a classifier can be used to predict the behavior of each packet based on the information of that packet, so that, a decision can be made whether to allow that packet into the protected network or deny it. Thus, it is possible to use a classifier to predict the state of the packet depending on the values extracted from the packet's information. However, a labeled dataset is required to train the classifier on detecting such packets, where different dataset are collected from network traffic that include normal and intrusion packets, so that, these datasets can be used to train classifiers to detect attack packets in order to filter them out of the network [6].

As the classifier makes a prediction for each packet, this prediction must be used to execute the filtering operation on that packet, where normal packets are allowed to the network, while attack packets are not. Among all the devices used to connect hosts in a network, such as switches and routers, firewalls are the network devices responsible for examining the specifications of each packet to a set of rules, configured in the firewall, in order to allow or deny the access of that packet into the network. However, as the intrusion techniques are generating traffic quite similar to that incoming form normal traffic, the use of static rules configures in the firewalls is no longer applicable, as it is impossible to detect an anomaly in the network traffic using these rules. Thus, predictions provided by the classifier are forwarded to the firewall to use these predictions, instead of the static rules, to make the appropriate decision for each packet incoming to the network. Moreover, these firewalls are also used to analyze the packets to extract required information needed by the classifier to make these predictions [7].

As the classifiers have different approaches to create models based on the extracted knowledge that is used to predict classes for these packets, where these approaches do not affect the accuracy of the predictions provided for the packets only, they affect the time required to compute each prediction. Moreover, to maintain the quality of the services provided on the network, it is important to provide rapid predictions, in addition to the accuracy of these predictions. Thus, the time required by each classifier to predict a label for a packet is also important to measure, where larger networks have an enormous number of packets flowing through, which may increase the latency of the network in cases where classifiers with slower predictions are used in the intrusion detection system. Thus, it is important to select a classifier that has a balanced performance, with respect to the accuracy of the predictions and the average time consumed by the classifier to make that prediction [8].

1.1 PROBLEM DEFINITION

Protecting networks against intrusions is becoming more and more difficult, as the techniques used to execute these intrusions are developing rapidly, so that, it is becoming difficult to distinguish packets incoming from legitimate users trying to access services on the network, and those of attacks used to gain authority by unauthorized users or degrading the quality of the services provided on that network. Traditional techniques that use static rules in the firewalls, to filter out any unwanted or suspicious traffic, are becoming inefficient against such intrusions. Thus, more efficient techniques are required to protect these networks against such intrusions.

The use of machine learning techniques is growing rapidly, according to the capabilities of these techniques in detecting relations among input values that may be difficult, or impossible, for humans to detect, because of the complexity of these relations or the enormous number of value in the input. Different applications employ these techniques to interact with external domains, by training these techniques using sample values, so that, the knowledge extracted during training is used with future inputs. However, the performance of these techniques is different from each other, and depends on the inputs of the system. Thus, the performance of different machine learning classification techniques must be tested in order to select the most appropriate classifier that has the ability to provide accurate and rapid predictions, so that, the security of the network is improved without affecting the quality of the services provided on that network.

1.2 AIM OF THE STUDY

As the performance of the intrusion detection system is affected by the accuracy of the predictions provided by the classifier and the time required to provide these predictions, the aim of this study is to evaluate the performance of different classifiers, so that, the classifier with the best performance is selected to implement such system. Thus, classifiers with topologies simpler than other classifiers, proposed in earlier studies, are evaluated in this study, so that, faster performance is assured while better predictions accuracies are provided by these classifiers. Using such classifiers, the intrusion detection system can provide better security for the network without affecting the quality of the services provided by the network. The classifier with the balanced performance, with high accuracy and low prediction time, is selected to provide binary predictions for the firewall to allow or deny packets access to the network, while a multi-class classification is used to predict the type of intrusion being executed against the network.

1.3 THESIS LAYOUT

The layout of the remainder of this thesis is as follows:

- Chapter two reviews background knowledge and related work.
- Chapter three illustrates the devices used to connect hosts in a network, the security of networks, types of most recent attacks being executed against such networks and the machine learning techniques that can be used to detect an anomaly in the network traffic.
- Chapter four describes the proposed methodology of the proposed system, the preprocessing techniques that are used to transform the data into a more suitable form for the classifiers and the performance measures that are used to evaluate the performance of classifiers.

- Chapter five illustrates the experiments conducted to evaluate the performance of these classifiers, the results of these experiments and discusses these results and compares them to results from earlier studies.
- Chapter six shows the conclusions of this study.

2. LITERATURE REVIEW

The KDD Cup [9] dataset is one of the popular datasets that are used for intrusion analysis and detection. This dataset consists of 41 attributes that describe each packet in the network, where each label is labeled as a normal packet or an attack packet. Attacks in this dataset are divided into four categories, which are Denial of Service (DOS), User to Root (U2R), Remote to Local (R2L), and Probing attacks. Two major issues exist in this dataset, the first issue is the enormous number of duplicate records in both the training and testing parts of the dataset, where about 78% of the training part, and 75% of the testing part, records have duplicates. These duplicates may result in biased training toward the duplicate records. The second issue is the difficulty level analysis of the dataset, where even the simplest classifiers are able to come up with good predictions accuracy for the records in both the training and testing datasets.

An intrusion detection system (IDS) is proposed by Wei-Chao Lin, et al. [10], which is based on the *k* Nearest Neighbors (k-NN) classifier that is trained using the KDD CUP'99 dataset. The accuracy of the proposed method based on the k-NN classifier is 99.89%. The k-NN classifier is a lazy classifier, which means that no prior training or knowledge extraction is accomplished until a prediction is required for a new record. When a prediction is required, the features values of the record being classified are compared to all the records that are in the training dataset, then, the *k* most similar records are used to predict a class for the new record. Although the k-NN classifier may result in accurate predictions, it consumes, relatively, more time than other types of classifiers for predictions, as it has to compute the similarity with all the records in the dataset, per each prediction.

The method proposed by Sakchi Jaiswal, et al. [11] has accelerated the prediction process by reducing the number of attributes used by the classifier to find the nearest neighbors of the new record, which would simplify the computations required for that task. The attributes selected by this method are selected based on the information gain of each attribute. Information gain is computed based on the information entropy, which is computed using equation 2.1, where H(X) is the entropy and p_i is the probability of a single attribute. Using the entropy value, information gain is computed using equation 2.2. The overall accuracy of the proposed method is 94.77%, which is less than that presented earlier, according to the reduced number of attributes.

$$H(X) = -\sum_{i=1}^{m} p_i \, log_2 p_i \tag{2.1}$$

$$H(X) = H(X) - H(X|Y) \tag{2.2}$$

Another approach is presented by Neha G Relan and Dharmaraj R Patil [12], which implements an intrusion detection system based on the decision tree classifier. The accuracy of the proposed system has scored a maximum of 95.09%, using the KDDCUP'99 dataset for both training and testing phases. The decision tree classifier creates sets of IF/THEN rules that can be applied to the attributes' values of each record, in order to predict a class for that record. These sets are generated based on the attributes values of the records in the training dataset, and the class that each record belongs to, where the sets are distributed in levels, and the condition to be investigated in the next level is selected depending on the result of the condition being applied in the current level.

The model proposed by Vrushali D Mane and SN Pawar [13] uses a neural network to classify the KDDCUP'99 dataset into binary classes, which means that each record is classified to be either a part of normal traffic or an attack. To simplify the neural network, required to achieve this task, only 17, out of the original 41, attributes are selected as inputs to the dataset, and the network includes only one hidden layer with 10 neurons. Moreover, to reduce the training time required by the network, only 10% of the data is used in the training process. However, the accuracy of the predictions made by this network has an accuracy of 98.0%.

John McHugh [14] illustrates the issues in the newer version of the KDD CUP'99 dataset, which is known as the NSL-KDD CUP'99. This dataset includes the same attacks that are included in the previous KDD CUP'99 dataset but has fixed the issue of duplicated records. However, the environment setup is considered to be questionable and more concerns are raised about the suitability of synthetically generated dataset to be applicable in real-life applications. Different studies are conducted to propose intrusion detection systems, based on this dataset.

The method proposed by Muhammad Shakil Pervez and Dewan Md Farid [15] uses the Support Vector Machine (SVM) classifier for intrusion detection purposes based on the NSL-KDD CUP'99 dataset. The best classification accuracy of the proposed method has scored 82.37% when all attributes in the dataset are fed to the SVM classifier. The SVM classifiers creates a multidimensional space, where the number of dimensions in the space is equal to the number of attributes in the dataset, then, the boundaries that split classes' records are represented mathematically, so that, when new unlabeled records are required to be classified, the SVM predicts a class for them depending on their position in the domain and the concluded boundaries as shown in Figure 2.1.

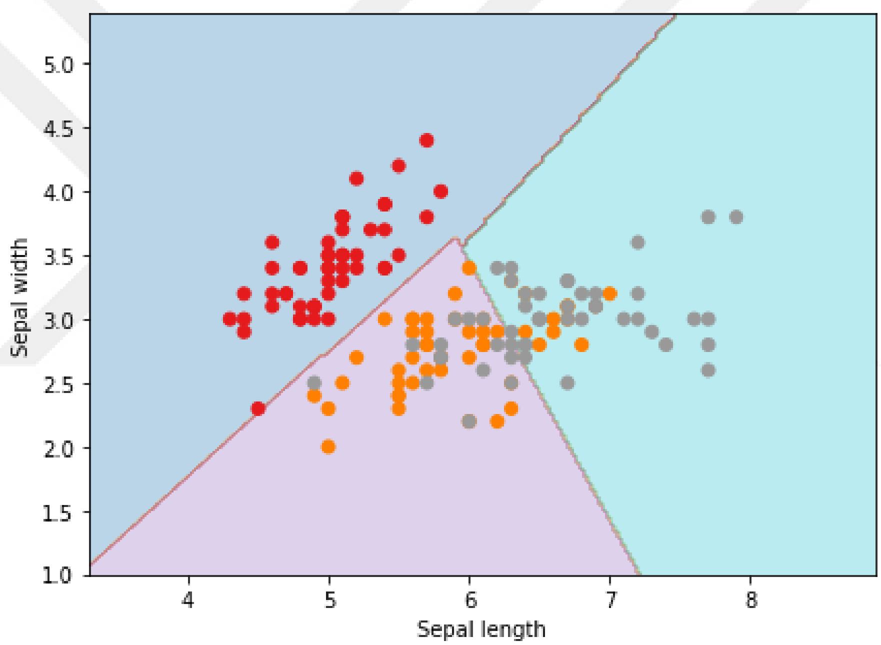

Figure 2.1: Sample SVM boundaries for classification.

The study proposed by L Dhanabal and SP Shantharajah [16] implements and compares three intrusion detection systems, each system is based on a different classifier that is trained using the NSL-KDD dataset. These classifiers are the decision tree classifier, SVM, and Naïve Bayes. The classification accuracies of these classifiers are 98.88%, 95.2%, and 73.32% sequentially. The Naïve Bayes classifier predicts a class for a record based on the probabilities computed per each attribute values with respect to the classes existing in the dataset. These probabilities are then computed based on the new attributes values of the new record being classified, with respect to every class in the dataset, then, the class with the highest probability that the record belongs to it is selected as a prediction for the record.

Two models, based on neural network classifier, are proposed by Bhupendra Ingre and Anamika Yadav [17]. One of the models classifies each record into one of the five classes of the dataset, which include one class for normal packets and four classes for five types of attacks, while the other model implements binary classification, whether the record is predicted to be normal or attack, regardless to the type of the attack. A total of 29 attributes are selected, from the 41 total attributes in the dataset, as inputs to the neural network, based on their roles in the classification process. The highest accuracy of classifying records into five classes is 79.9%, while the highest accuracy achieved using binary classification is 81.2%. The operation of neural networks classifiers is discussed in details in the next chapter of this study.

A more recent dataset is proposed by Nour Moustafa and Jill Slay [18], which is known as the UNSW-NB15 dataset. This dataset includes real-life network traffic with both normal and abnormal packets in a synthetic environment. The packets in this dataset are labeled into ten classes, one for the packets generated by the normal traffic, and nine attacks that appear in the traffic. These attacks are the Fuzzers, Analysis, Backdoor, Dos, Exploit, Generic, Reconnaissance, Shellcode and Worms.

A multi-stage decision tree classifier based intrusion detection system is proposed by Mustapha Belouch, et al. [19]. This system is trained and tested using both the NSL-KDD and the UNSW-NB15 dataset. The system consists of two stages, the first stage predicts whether the packet is a part of a normal traffic, or is an intrusion attempt, then, in case that the packet is predicted to be an intrusion attempt, the next stage is triggered in order to predict the type of attack being executed for the intrusion attempt. The classification accuracy of the proposed method is 88.95% for the UNSW-NB15 dataset, and 89.85% for the NSL-KDD dataset.

Hossein Gharaee and Hamid Hosseinvand [20] implemented an IDS that combines the benefits of a genetic algorithm to reduce the number of attributes used for classification, and the SVM classifier, to extract knowledge from the dataset, and predict classes for the new packets. The implemented system is tested on both the KDDCUP'99 and UNSW-NB15 dataset, but, besides the normal class, only six out of the nine attacks that are included in the UNSW-NB15 dataset. The implemented method has an average accuracy of 99.26% for the KDDCUP'99 dataset, and 93.25% for the UNSW-NB15.

Rana Aamir Raza Ashfaq, et al. [21] proposes a semi-supervised learning approach based on fuzziness to make use of unlabeled data alongside with the labeled ones, to improve the quality of the data used for classifier's training. The semi-supervised approach uses fuzziness vector to cluster labeled and unlabeled data, so that, the labels of the unlabeled data can be predicted. This reduces the need for domain experts to label all the data in the dataset, which makes it more convenient to include more data in training by providing labels to those unlabeled data. The results of this approach are fed to a feed-forward neural network, with only one hidden layer. The results of this approach show significant improvement in the classification results, compared to other classifiers, such as the J48, Naïve Bayes tree, and SVM, where the approach based on semi-supervised learning has scored an accuracy of 84.12% while the Naïve Bayes tree, J48 and SVM have scored accuracies of 81.59%, 81.05% and 69.52% when used to classify the NSL-KDD dataset.

Partha Sarathi Bhattacharjee, et al. [22] implements an intrusion detection system based on Genetic Algorithm that employs Weighted Vectorized Fitness functions with Fuzzy membership function. The Fuzzy membership function computes the probability of a value to be in a certain category, instead of login predictions where values are classified to be in a certain category among many as shown in Figure 2.2. The results of the study show that the employment of the Fuzzy Vectorized Genetic Algorithm has improved the accuracy of the classification results up to 99.18% using the NSL-KDD dataset.

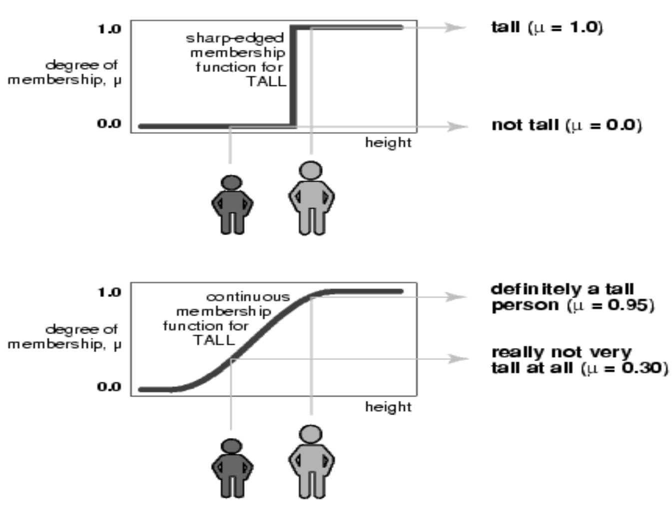

Figure 2.2: Illustration of Fuzzy membership function.

Moustafa and Slay [23] present an intrusion detection system that employs linear regression to compute the probability of incoming packets to be a part of normal traffic or of a specific kind of an attack. Linear regression generates a distribution of probabilities of tuples to be in a specific class, depending on the corresponding values of the data in the training dataset that belong to the class. Then, when a new data comes in, the attributes' values are projected on the distribution to compute the probability of the incoming data to be in any of the existing classes. The proposed method is tested using both the UNSW-NB15 and NSL-KDD dataset. The results show that the implementation has a relatively higher accuracy than the Expectation-Maximization and Naïve Bayes methods, where the proposed method has an accuracy of 83% when tested with the UNSW-NB15 and 82.1% with the NSL-KDD, while the Expectation-Maximization method has an accuracy of 77.2% and 74.4% for the same datasets, and the Naïve Bayes has 79.5% and 28.9% accuracy for the same datasets.

Rifkie Primartha and Bayu Adhi Tama [24] compares the performance of the Random Forest (RF) classifier with other classifiers proposed in different studies, such as decision tree, random tree and multi-layer perceptron. The results show that the random forest classifier with 800 trees in the forest has a better average performance than the other classifiers using the KDD'99, NSL-KDD and the UNSW-NB15 network traffic datasets. The classification accuracy of the random forest classifier with the UNSW-NB15 is 95.5% with False Alarm Rate (FAR) of 7.22%. False alarm rate is the ratio between the number of normal packets that are rejected by the classifier to the total number of packets predicted to be intrusion packets by the classifier.

Malek Al-Zewairi, et al. [25] proposes an intrusion detection system based on feed-forward deep learning neural network that consists of five hidden layers with ten neurons in each layer. The deeper the neural network, the more complex features can be detected based on the input data, while increasing the number of neurons in a layer increases the number of features that the layer can detect. The performance of the deep learning model is compared to other classifiers, such as decision tree, logistic regression, Naïve Bayes and neural network, where the experimental results show that the deep learning model outperforms the other model tested in the study with 98.99% accuracy.

3. DATA MINING AND INTRUSION DETECTION SYSTEMS

3.1 INTRODUCTION

This chapter discusses the basic concepts of the machine learning, especially data mining, and their applications regarding intrusion detection systems by providing an overview of networks and their security, intruders and how types on existing intrusions, as well as the data mining techniques and how they can be employed for intrusion detection.

3.2 NETWORK DEVICES

Regardless of the type of media used to connect network devices to each other, three types of devices are usually used to implement a network. These devices are the switches, routers and firewalls. Switches are used to connect devices to the same subnetwork, where all the devices in that subnetwork communicate with each other directly. Before switches, hubs are used to connect devices in the same subnetwork, where a packet coming from one device connected to that hub is reflected on all the ports of the hub, where all the devices on the subnetwork receive that packet and are expected to neglect it if not directed to them. Thus, the bandwidth of a hub is shared between all the devices in the network and the security of the information being exchanged between two devices is in higher risk to be sniffed by other devices that are on the same subnetwork. However, switches keep a list of devices connected to each port, so that, when a packet is direct toward one device is reflected on the corresponding port only, while a packet directed to an unknown host is reflected on all ports. The entries in the hosts list are created whenever a new device sends information through the switch, as the switch is able to detect the port that the packet has come from and the address that sent this packet. This behavior provides more security to the information being exchanged through the switch as well as dedicated bandwidth per each port on the switch [26].

Routers are used to connect subnetworks to each other, so that, devices on different networks can interchange information among them. Usually, routers have multiple ports, where each port has a different network configuration and is connected to a different network. Each port on a network can directly be reached from all the devices in that subnetwork, and information directed to devices on other networks are sent to the router in order to deliver them to that device. Routers have special tables that contain information about the reachable networks and how to reach them. This information can be hard-coded by the network administrator, or dynamically generated by exchanging information between every two connected routes, telling each other about the networks that they can reach using Routing Information Protocol (RIP). The only piece of information that routers are concerned about in a packet is the address of the destination device, which is compared against the information in routing table in order to decide the next hop, where the packet should be sent to in order to receive its destination [27].

Firewalls are hardware devices that are connected to a network to monitor and analyze packets going to or from the network in order to protect any unauthorized type of communication. Unlike routers, firewalls analyze different parts of the packet, such as the source address, sources port, destination address and destination port and compare then to a set of rules in order to decide whether the packet is allowed to pass the firewall of should be denied. Firewalls are used to manage access to services exist on different networks, as well as protecting networks by denying packets that are against the rules of the firewall to pass through. Thus, network protection should always be implemented in the firewalls [28].

3.3 NETWORK SECURITY

Practices and policies that control the operation of a network in order to prevent any unauthorized access to that network, which intends to misuse the resources on the network, or attempt to deny services from being accessed by, or from, other networks. An enormous number of network attacks are executed nowadays using different techniques, which may target different devices on the network and cause different types of damage [29]. Moreover, network security may be involved in private networks as well as public networks, where private networks may be local networks in a business, government or larger interconnected organizations [30].

The increased demand for online services forces most of the companies and services providers to catch up these demands, by making any possible online service available for their clients. This phenomenon has imposed the need to maintain these services in order to maintain the reputation of the company or organization, where even governmental services are being provided online. To do so, it is mandatory for these organization to store clients' information on servers that are reachable from the internet, so that, the requires services are provided to these clients. Thus, these organizations have the obligations of maintaining the quality of the provided services, and the confidentiality of the clients' information being stored on their servers [31].

3.3.1 Intruder

An intruder is a person that has no authority to use specific services, or access certain data, but attempts to access these services or data using a network connected device, or more, which can reach the network where the intrusion is intended to be executed [32]. There are three main types of intruders, which are misfeasor, masquerader, and clandestine user. A misfeasor is a user on the network, who has access to certain data and services, but attempts to access information that has no authority to access, while the masquerader is the unauthorized person who has no access to any of the services or information on the network and usually attacks from outside the network. Moreover, clandestine users attempt to make use of the privileges they have over the network devices to acquire information about clients who are using the system. Such attacks may be executed locally, on the same network, or from outside the network, using the same privileges they have over the system.

3.3.2 Intrusion

An intrusion is defined as the act of gaining access to a service or data that the intruder does not have legitimate access to them, or an attempt to affect the quality of the services provided over this network [33]. These intrusions can be performed using different approaches, where some intrusions are executed from the same network where the victim device is located, or from another network, which is also connected to the victim's network. Moreover, some intrusions are executed using a single device to perform the attack, while other techniques use multiple computers to perform the intended intrusion [34].

3.4 HISTORY OF INTRUSION DETECTION SYSTEMS

The seminal paper proposed by [35] is the first known attempt to monitor the packets being transferred in a certain network in order to analyze them, so that, the normal user behavior is understood in order to distinguish any other behavior that may be a threat to the network. That work is considered as the base that launched all other Host-Based Intrusion Detection Systems (HIDS) to detect and prevent any intrusion attempts to the network. Based on that study, an intrusion detection model is proposed by [36] that analyzes the user behavior on a governmental mainframe in order to generate a profile for the legitimate users of the system, then, block users who have different profiles from accessing the mainframe.

In November 1988, the Morris internet worm had been released, which is the first known intrusive program that has the ability to spread automatically over the internet connection. That worm has affected the internet, at that time, so bad that it has disabled servers of two of the major corporations, which are the DEC VAX and Sun-3. The worm has the ability to infect a single computer multiple time, causing them to go extremely slow by executing each infection separately, which causes these computers to crash multiple times. This behavior of the worm is not implemented intentionally, but it is because of a lack of experience in intrusions, where the code did not check whether the worn exists on the computer or not before attacking it. Thus, it eventually has attacked the same computer from other computers, whenever that computer is reachable [37].

Investments in networks security models have gained significant attention after the intrusion detection system that is proposed by [38], which is known as Network Security Monitor (NSM) at that time. Massive amount of network traffic information is analyzed in that method in order to distinguish suspicious behavior of the network users, which is based on the use of hybrid methods to detect and block malicious users on the network. Moreover, on the same year, an Automated Security Incident Measurement (ASIM) system is proposed by the United States Air Force through their Cryptologic Support Center, where this system is the first known system that implements intrusion detection system using standalone hardware with a specially implemented software that runs on it.

Evaluation measures for the intrusion detection systems have started to appear in the literature, where the IDS maintainability, scalability and efficiency are the most common measures that are used in IDS evaluations. Moreover, these systems are categorized into two main categories, based on the implementation concepts, which are the Distributed Intrusion Detection System (DIDS), and the Network Anomaly Detection and Intrusion Reporter (NADIR). These systems are designed to protect multiple hosts from attacks, by collecting and analyzing data collected from the network traffic [38].

Ever since, different commercial intrusion detection systems are proposed to protect networks from network attacks. NetRanger by Cisco, OmniGuard Intruder Alert by Axent, and RealSecure by ISS are examples of the commercial intrusion detection system, which is based on signatures of the attacks, therefore, they are required to be updated in whenever a new intrusion type is proposed. Many attempts have been made to come up with a Common Intrusion Detection Framework (CIDF), which attempt to provide a common specification language for intrusions, but such framework is difficult to provide, as there are no IDS that can detect all types of intrusions flawlessly [39].

3.5 TYPES OF NETWORK ATTACKS

Although there is an enormous number of network attacks that already exist and is dramatically growing, so that, it is quite difficult, if not impossible, to illustrate all of them, these attacks can be categorized into two main categories, which are the passive and active attacks. However, these attacks may be similar to each other, with only a few changes that provide them the ability to go through the existing protection schemes [40]. Thus, in this section the main categories are illustrated with few of the most popular attacks in each category.

3.5.1 Passive Attacks

In passive attacks, the intruder monitors and analyze the packets being transferred in the network, without the knowledge of any of the legitimate partners who are interchanging these data. By doing so, the intruder gains knowledge of the information being transferred between two authenticated partners, without having any credentials to access this information. This type of intrusion is difficult to detect, as the intruder may not leave any traces that may indicate the existence of a third party monitoring the information being interchanged [41]. Some of the known passive attacks are the Wiretapping Attack, Traffic Analysis Attack, and the Release of Message Contents Attack.

3.5.2 Active Attacks

When intruders execute an active attack, they may send, receive, modify, or reply network messages, in order to gain access to information or services that they have to authority to access. Some active attacks may gain no access to any information, they only affect the performance of the services provided by the network, by slowing, or shutting, down these services. As the intruders in active attacks so interact with the servers, and interchange network traffic with them, this type of attacks is easier to detect, however, it may severely harm the performance of the network [42]. Some of the active network attacks are:

1. **Denial of Service (DoS):** These attacks aim to block legitimate users from accessing the services provided by the network, or, denying access to the network resources, such as servers and other hardware devices. The main concept behind this kind of attacks is to flood the network with a lot of information, that may look like initiated from legitimate users, in order to consume the available resources on the network, such as the bandwidth or processing power. In such an attack, intruders do not gain any access to any information, and intend to reduce the quality of the services provided by these networks.
2. **Spoofing Attack:** In such attacks, intruders send information pretending to be someone else, who is a trusted or legitimate user or service provider. For example, an intruder sends an email to a client from a fake email address pretending to be the owner of that email address, or, sending network packets to a server pretending to be a legitimate user who is trying to access information that the user has the authority to access. In both scenarios, the intruder may gain access to confidential information, or make use of services that are not intended to be provided to the intruder.
3. **The Man in the Middle Attack:** In this kind of attacks, the intruder takes place in the middle of the communications between two legitimate partners, so that, all the data being exchanged goes through the intruder before reaching the other partner. In this case, not only the information is disclosed to the intruder, but the intruder may also modify, insert or delete some of the messages being exchanged between these partners.
4. **ARP Poisoning Attack:** Address Resolution Protocol (ARP) is used in the lower levels of the Transmission Control Protocol (TCP) to resolve the physical addresses of the network interfaces that information is targeted to. In this type of attacks, the intruder replies to all, or a specific, resolve requests as the owner of the required address, thus, all information that is intended to be sent to that address are redirected to the intruders, which may reveal this information to the intruder, or just denies users from accessing the required services, as the sent messages are not reaching their destinations.

5. **Buffer Overflow Attack:** Network interfaces have buffers that hold the incoming data before forwarding them to the next step. These buffers are of limited size, so that, sending too much communication toward that network interface, faster than the data retrieval capacity of the device, causes these data to overflow the buffers, which means that the buffer needs to get rid of some of the existing data in order to fit the incoming ones. This results in losing the information coming from legitimate users to store data coming from the intruder.

3.6 MACHINE LEARNING AND DATA MINING

Providing computers with the ability to gain knowledge or making decisions with the external world without any interaction from humans is known as machine learning. In machine learning the same algorithms may have different outcomes depending on the inputs of the systems, where these inputs may have never been through the system before but the system still has the ability to process them. Data mining is one of the machine learning fields of study that Machine learning techniques can be categorized into two categories, which are the supervised and unsupervised machine learning [43].

3.6.1 Unsupervised Machine Learning

Unsupervised machine learning is used to extract knowledge from datasets as they are, where the extracted knowledge represents relations between the attributes values of the records in the dataset. This type of machine learning required no labeling to the records in the dataset, as these algorithms tend to find the relations among the records themselves, depending on the values that characterize each record. Clustering is one of the most popular unsupervised data mining techniques, where records in the dataset are distributed, into groups, based on their attributed values. In these groups, each record is more similar to the other records in that groups than any other records in the other groups. Thus, clustering generates groups of homogeneous records [44].

Moreover, the number of groups is the main factor that affects the performance of the clustering process, where a larger number of clusters increases the time required to process the records in the dataset, without any actual benefits of these extra clusters, while clustering records into a smaller number of groups generate meaningless clusters. Thus, it is important to cluster the records into the optimal number of clusters, depending on the distribution of the attributes values in the dataset. This optimal number may be provided by humans to the clustering algorithm, or by using some optimal number of clusters selection techniques, wherein these techniques different number of clusters are tested in order to select the optimal number of cluster for that dataset, depending on specific factor per each number of clusters selection technique [45].

3.6.2 Supervised Machine Learning

In supervised machine learning, the inputs of the systems are required to be labeled in order to extract knowledge from these inputs. The relations between the inputs and the labels given to them are investigated in the supervised machine learning techniques. Classification is one of the most widely used supervised data mining techniques, where the label given for each record represents the class that this record belongs to. Then, the classifiers extract the relations between the attributes' values that characterize that record, and the class that the record is labeled to be a member of. This knowledge is then applied to new records that are not classified in order to predict a class for them. This prediction can assist estimating the future behavior of that new record, depending on the general characteristics of the records on that class [46].

For knowledge extraction, classifiers need labeled dataset, so that, this dataset is used to train the classifier. This dataset is known as the training dataset. However, as the classifiers are used for predictions, it is not possible to evaluate the performance of the classifier using unlabeled dataset, while using the same training dataset is not a good method to evaluate their performance because the classes of these records are known to the classifier during the training, and this evaluation does not measure the prediction performance. Thus, in order to provide more accurate measures, the labeled dataset is split into two parts. The first part is used for the training phase, and the other is used for testing the classifier. Using such approach, the data used for evaluation is not included in the training, but the actual classes of the records in that dataset are known, so that, the testing dataset is fed to the classifier and the classes predicted by the classifier for the records in the testing dataset are compared to the actual classes that they belong to, in order to produce accurate evaluation measures [47]. There are different classifiers used for extracting knowledge from a dataset. These classifiers have different approached of knowledge extraction. However, the classifiers' performance may vary from one dataset to another, where a certain classifier may have better performance than another when applied on a certain dataset, while the other classifier may outperform it on another dataset. Thus, it is important to test the performance of more than one classifier on a dataset, to select the classifier of the best performance. Moreover, there are classifiers that show a better overall performance than others, such as the Support Vector Machine (SVM), Random Forests (RF) and Deep Learning (DL) classifiers.

3.6.2.1 Support Vector Machine Classifier

To classify the records in the dataset, the SVM classifier creates a domain space for all the records in the dataset, where the number of dimensions in the domain space is equal to the number of attributes in the dataset, and the records are represented as points according to their attributes' values. Then, the SVM classifier splits the domains according to the number of classes in that domain. To achieve this approach, the SVM extract equations for the boundaries that slip these regions, then, when prediction is required for a new record, the position of that new record is examined against these boundaries in order to find the region that the record falls in, hence, a class is predicted for that record [48].

Figure 3.1 shows a simple example of a two-dimensional space with records distributed in the space and are labeled with two different labels. The figure illustrates the possibility of finding more than one boundary to split the space into two regions, each region contains records of one label. Thus, it is important to find the best boundary that splits the regions, so that, better predictions are provided later.

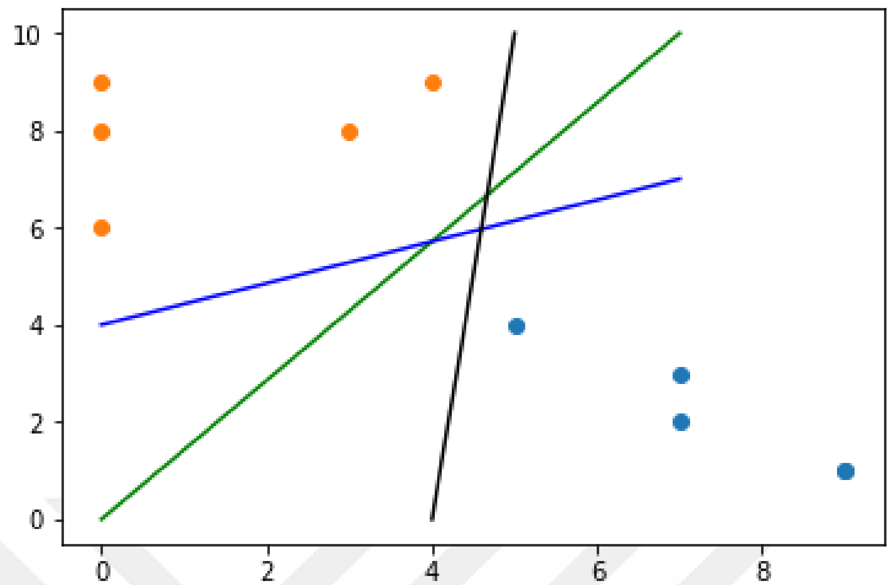

Figure 3.1: Possible boundaries that split records in a two-dimensional space.

In SVM classifier, the confidence of a prediction is computed based on the distance between the record and the boundaries, where the larger is the distance, the more confident is the prediction. Thus, to maximize the predictions confidence and reduce the errors, margins are set around the boundaries set for the space, which represent the minimum distances between the boundary created and the nearest point of each class. Then, the classifier optimizes the boundaries by maximizing the margins. Figure 3.2 illustrated the margins of the boundaries for the above example, where the green line has the farthest boundaries, therefore, this line is selected for classes predictions of any new records.

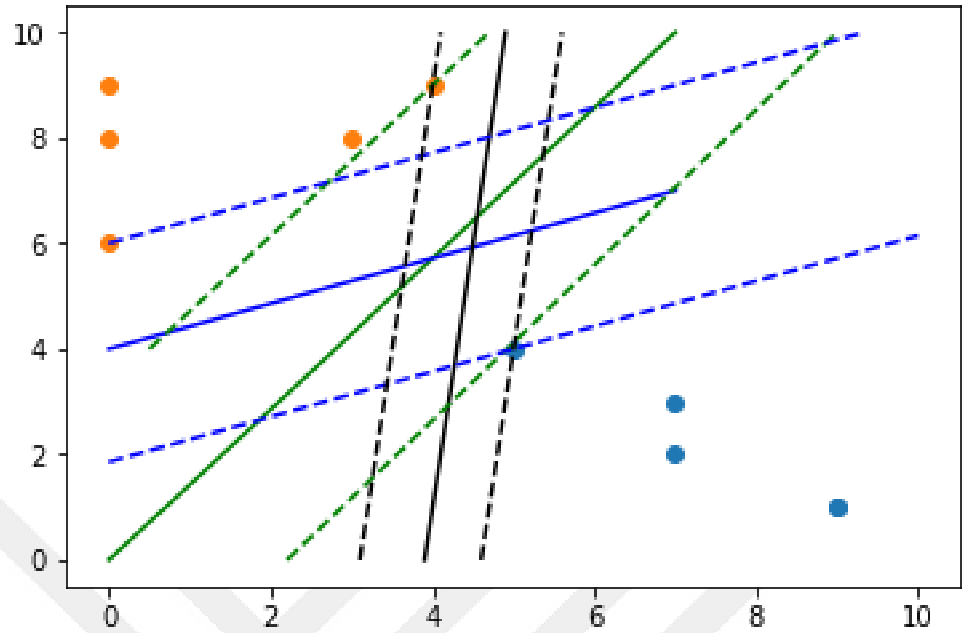

Figure 3.2: Margins around the boundaries that split the domain space.

3.6.2.2 Random Forests Classifier

Random forests classifier is based on decision trees, where the extracted knowledge from the training dataset is represented using a set of IF/THEN conditional statements. These statements are arranged in a tree-like topology, where the root is on the top with one conditional statement, while the remaining statements are distributed in levels. The decision of each level decides the direction that the comparison goes to, in the next level. Each comparison in a certain level may lead to another comparison in the next level or a decision for the prediction, which are known as leaves [49].

In a Random Forests classifier, the training dataset is slip into batches that are equal to the number of trees in the forest. Then, different decision trees are generated, one per each data batch. A tree in the forest may, or may not, be similar to other trees in the forest. This approach minimizes the dependency of a single attribute value, hence, provide more flexible and accurate predictions. Depending on one attribute values than other may provide better accuracy when this attribute value is dominant on the class, however, it lacks the ability to predict the correct class for less frequent attribute values in that class. Thus, the Random Forest provide more accurate predictions by depending on multiple paths to predict a class for the incoming record, by providing multiple prediction, one per each tree, then selecting the dominant class among these predictions. However, this approach requires more processing time, as multiple predictions are computed per each class in order to find the most appropriate one [50].

A sample dataset is shown in Table 3.1 for weather condition and the status of a player to play on that day or not. The decision tree created for that dataset in order to predict the play status for any new weather conditions, is shown in Figure 3.3.

Table 3.1: Sample weather dataset for decision tree classification.

Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

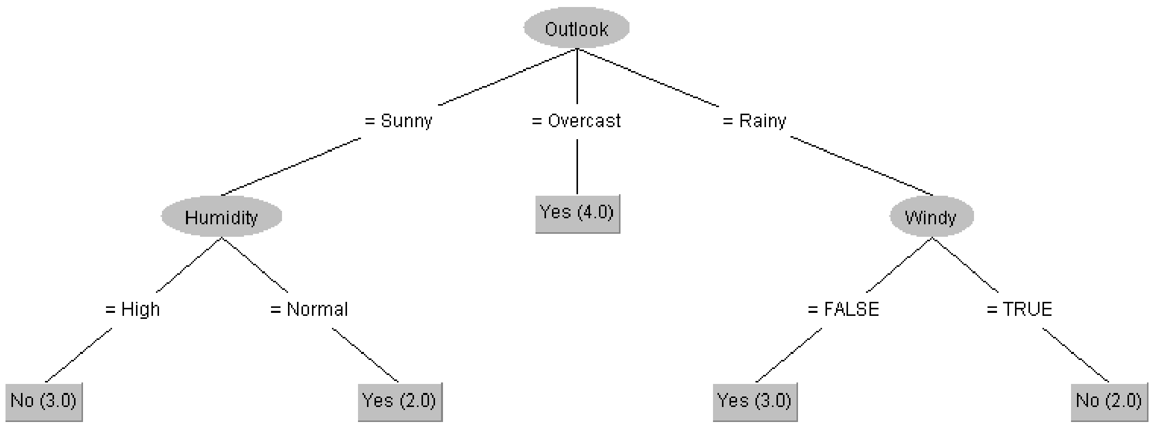

Figure 3.3: Decision tree for the weather sample dataset.

3.6.2.3 Deep Learning Classifier

Huge emphasis on the applications based on neural networks has been shown recently, according to the good performance of these networks, especially in data classification. Neural networks are mathematical representations of the way that signals are processed in the human brain, where neurons in a human brain are connected together in order to create the neural networks. Some of the neurons are loosely connected to other neurons, while may be rigidly connected to other, so that, different combinations exist to trigger different neurons based on their input [51].

A typical neural network consists of many neurons distributed in layers. Each layer may be an input, output, or hidden layer. The first layer that the inputs are fed to is known as the input layer, while the layer that generates the required output is known as the output layer. However, to improve the performance of the neural network, one or more layers are added in between the input and the output layers. These layers are known as the hidden layers as they are not visible to the external world. The number of neurons in the input layer is controlled by the number of inputs to the neural network, i.e. the number of the attributes, while the number of neurons in the output layer is controlled by the number of outputs required from the neural network. However, more neurons can result in better results, as increasing the number of neurons increases the possible combinations to trigger different actions, but it is not possible to add these neurons to the input or output layers. Thus, these neurons are added in the hidden layers[52] .

Moreover, the more the hidden layers, the more complex features can be examined in order to reach a more confident decision. However, the larger the number of neurons in the network, or the number of hidden layers, increases the computational complexity of the network, which increases the time required to process an input in order to compute the output. Thus, it is important to start with a smaller network, then, neurons and layers are added to avoid increasing the complexity without any benefits at the output. When there is more than one hidden layer in the neural network, such network is known as a deep learning network, as the features that trigger the neurons in the second hidden layer, and above, are more complex than those computed at the first hidden layer [53]. A sample deep learning neural network is shown in Figure 3.4.

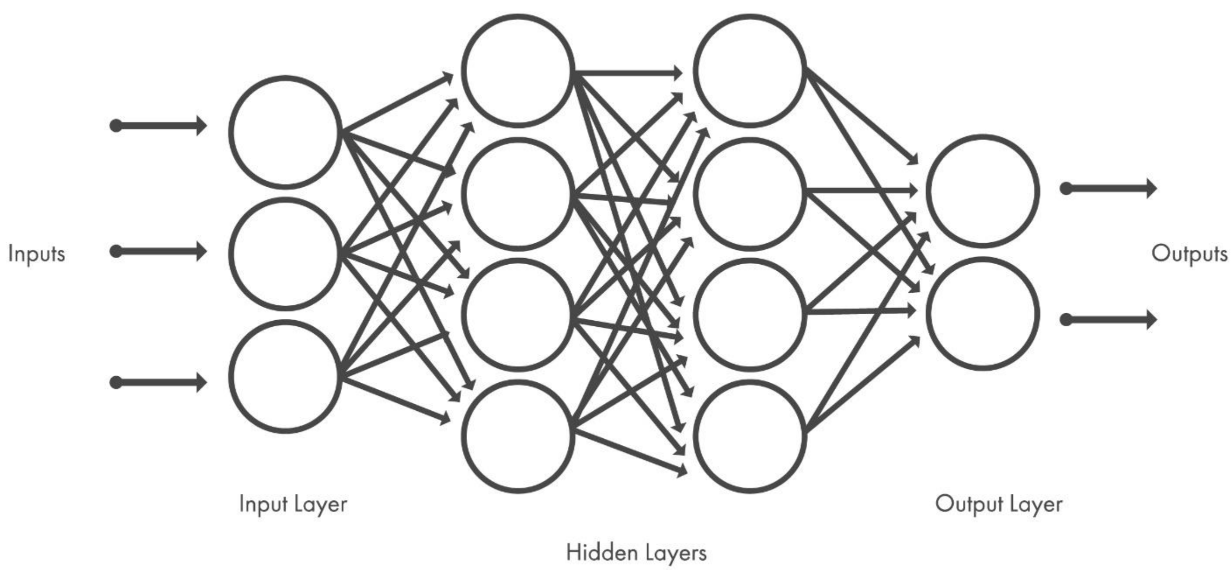

Figure 3.4: Sample deep learning neural network

The arrows that connect neurons from one layer to the neurons in the next layer are known as weights, where the larger the effect of the source neuron on the destination neuron, the larger is the weight between them. In a fully connected neural network, each neuron in a certain layer receives the outputs of all neurons in the previous layer, each one multiplied by the corresponding weight. The neuron, then, sums all these inputs and passes the summation results into an activation function. Activation functions provide non-linearity to the output of the neuron, where there are different types of application function, which are illustrated in Figure 3.5. Moreover, to provide further flexibility to the neural network, each neuron adds a bias value to the inputs that are incoming from the previous layer, which is also multiplied by a weight value. All the weights of the neural network are initialized using different random techniques but are updated during the training phase of the neural network using backpropagation, by measuring the error between the output values and the required values [54].

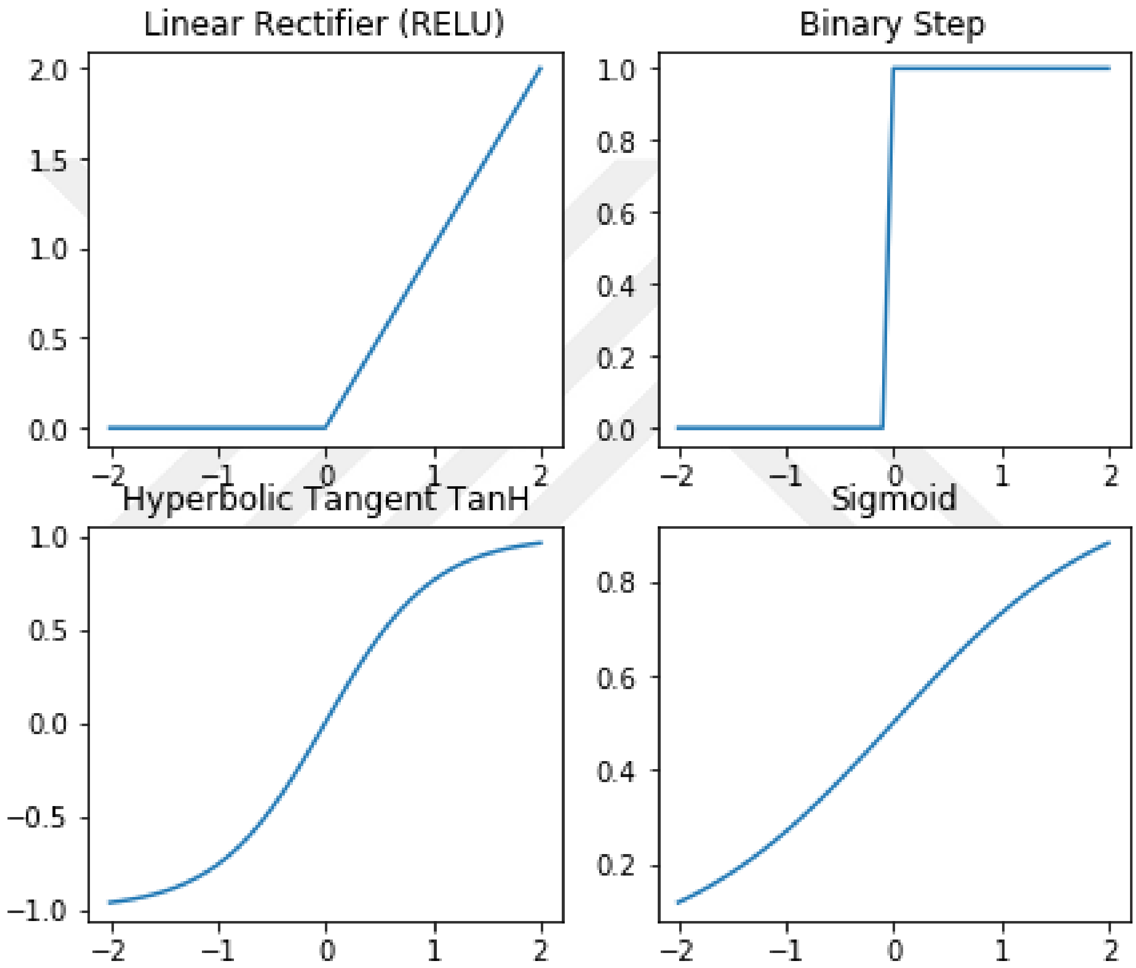

Figure 3.5: Activation functions of neural networks.

4. METHODOLOGY

4.1 MODEL ARCHITECTURE

As described in section 3.2, firewalls are the network devices that are responsible for protecting the network devices for attacks, by analyzing the packets going in and out of the network. Thus, an intrusion detection system should work together with the firewall, so that, the packet information is retrieved by the firewall and sent to the IDS, then, the IDS process this information in order to predict a category for that information and pass a decision to the firewall in order to allow or block that packet. An overview of the model architecture is shown in Figure 4.1.

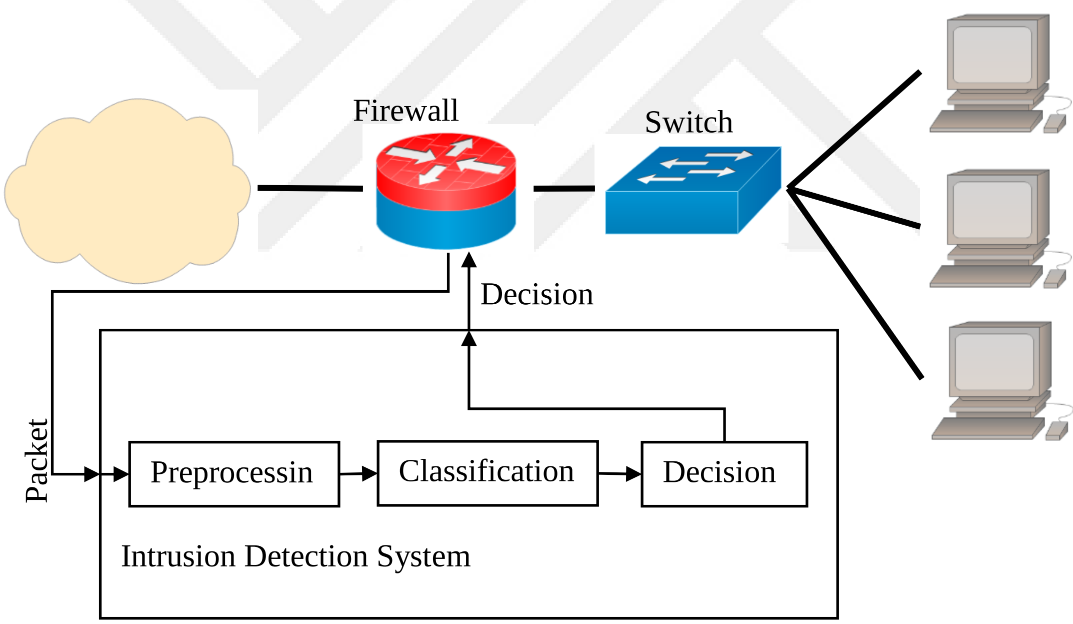

Figure 4.1: Model architecture overview.

4.2 DATA PREPROCESSING

Real-life datasets include different types of data, such as nominal, ordinal or numerical values of different ranges. Moreover, some databases may include some missing values and wrong values, which are values out of the normal ranges, such as a value of 200 in human age, or wrong combination of values, such as a record that has male in gender attribute and yes in the pregnant attribute. These values must be adjusted, so that, it is possible to train the data mining techniques, as well as providing accurate knowledge to come up with accurate predictions [55].

4.2.1 Label Encoding

Values in a database may be numerical or discrete values, where discrete values may be ordinal or nominal value. However, some data mining techniques do not have the ability to process discrete values, as these techniques include computations based on the input values. Thus, sometimes it is important to encode the discrete values into numbers, so that, it becomes possible to process these databases using these classifiers. Ordinal values are the discrete values that each unique value has a certain position when all unique values of that attribute are ordered, which means that ordering these values reflects a meaning of these discrete values. For example, discrete values that represent different age groups may be ordered in a certain position, where infants are smaller than babies, and babies are closer to adults than seniors. Moreover, nominal values have no meaningful order, such as the gender, where it is not possible to tell which nominal value goes where.

Label encoding converts these discrete values into numerical form, where if the values in that attribute are ordinal values, then they may be assigned with numbers according to their order and distances between one discrete value and another. Moreover, when the discrete values are nominal, numerical values can be assigned randomly, or by any selected order, such as the alphabetic order of the values in the attribute. This preprocess allows processing these values using classifiers that accept only numerical values, such as the SVM and DL classifiers [56]. Sample labels and their encoding are shown in Table 4.1.

Table 4.1: Sample labels and their encoded values.

Label	Encoded Label
B	2
A	1
C	3
B	2
C	3
A	1

4.2.2 One-Hot Encoding

In neural networks, it is not possible to predict a class for an input that can be classified into one of multiple classes using only one neuron. Thus, the number of neurons in the output layer is equal to the number of classes in the training dataset, where the neuron corresponding to the predicted class has the highest value among all other neurons in the output layer. In order to train the neural network to achieve such behavior, it is important to convert the labels in the training dataset into the appropriate shape, which should have a size equal to the number of neurons in the output layer of the neural network. To convert the labels of the dataset into the required shape, a vector is generated for each record with a width equal to the number of classes in the training dataset, i.e. the number of neurons in the output layer. Each class in the database is assigned with a position in that vector, where the value is set to one for records that belong to that class, while all other values are set to zero. This produces a vector with only one hot value, which is one, while all the remaining values are zeros, which is the reason behind naming this technique a one-hot encoding [57].

Table 4.2: Sample encoded labels and the corresponding one-hot encoded values.

Encoded Label	One-Hot Encoded Values		
2	0	1	0
1	1	0	0
3	0	0	1
2	0	1	0
3	0	0	1
1	1	0	0

4.2.3 Data Normalization

Numerical attributes in a single database may have different ranges, where a range is defined by the minimum and maximum values that appear in the records' values of that attribute. The attributes values may fall anywhere between the minimum and the maximum values of that attribute, which is also to describe and implement in a classifier. However, when two, or more, numerical values are inputted to a classifier that applies computation on these inputs, such as the SVM and DL classifiers, it becomes difficult for these classifiers to extract mathematical representation for the output based on these values. For example, if one of the input attributes have a range [0,5], while another has a range of [500,1000], then the record with input values of 2 and 700, respectively, is actually having the same values compared to the ranges of each attributes, which is 20% of the range. But mathematically it is difficult for the classifiers to adjust their parameters to adopt these range, which also gets more complicated when the number of attributes is increased. Thus, data normalization makes computations much easier and more relative to the classifiers, which produces better results and simplifies the computation inside the classifiers [58]. For an attribute that has a maximum value m and minimum value n, then each value o in that attribute is replaced with a new value v computed using Equation 4.1:

$$v = \frac{o - n}{m - n} \tag{4.1}$$

Sample attributes' values are shown in Figure 4.2. This figure illustrated how data normalization maintains the relativity among the values of the same attribute, however, the effect of these values is equalized when the values are normalized.

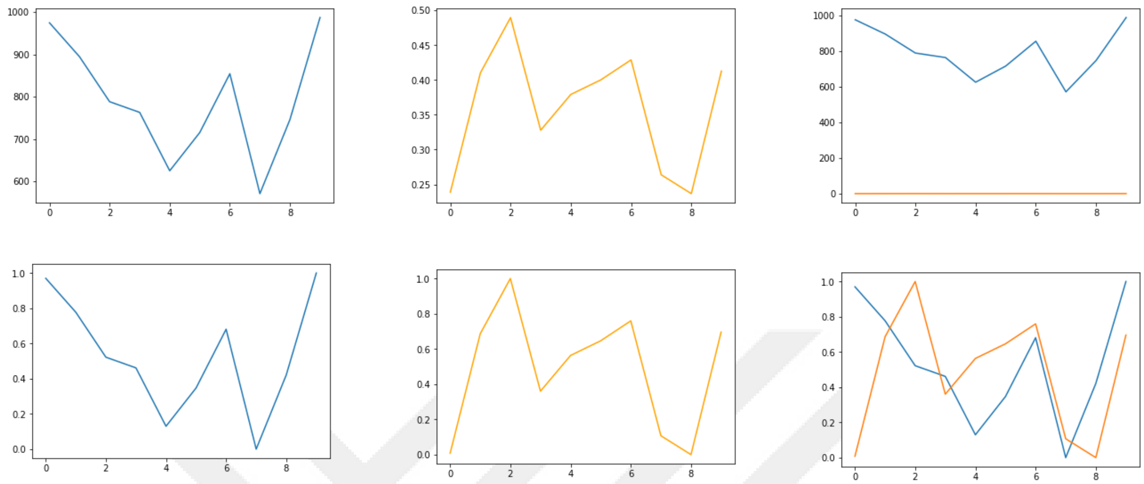

Figure 4.2: Sample attribute's values; Top: Original values; Bottom: Normalized Values; Left: Sample values 1; Middle: Sample values 2; Right: Both attributes' values.

The employment of these techniques to preprocess the network traffic dataset for intrusion detection is shown in Algorithm 4.1.

Algorithm 4.1: Data preprocessing algorithm.

Algorithm: Data Preprocessing
Input: Raw Data **Output:** Preprocessed Data
Step1: Read the entire input data. Step2: Split attributes from labels. Step3: Remove socket information. Step4: Replace missing attributes' values with 0. Step5: Encode attributes' categorical data. Step6: Normalize the values per each attribute. Step7: Remove white spaces from labels Step8: Replace missing labels with 'Normal'. Step8: One-hot encode label values (for deep learning only).

The socket information includes the source and destination IP addresses and port numbers. This information is removed to ensure unbiased training toward a specific host or service, so that, if a new device is added to the network or the IP address of a host is changes, the classifier still able of detecting the attacks in order to block their traffic. The output data from the preprocessing phase have no missing values and all the attributes' values are numerical with identical ranges, from zero to one. This enables better knowledge extraction from the dataset using classifiers, as in some classifiers the prediction is made by applying mathematical operations on the inputs, which makes it difficult to provide accurate predictions when the values' ranges among attributes are different. However, data normalization preserves the relevance among the values in the same attribute, which makes this formation of data more suitable for the classifiers. Moreover, in neural networks, the use of one neuron to provide more than one class is quite inaccurate, thus, the labels are encoded using the one-hot encoder, so that, it is possible to use one neuron per each class to produce more accurate results.

4.3 CLASSIFICATION

Classification uses data mining techniques to examine the relations between the attributes' values of each tuple and the label given to that tuple. Different classifiers use different representations of these relations, so that, when new tuples are fed to the classifier, it is possible to apply the extracted knowledge to predict a label for that tuple depending on the attributes values that characterize the tuple. These relations are different from one dataset to another; thus, it is important to train the classifiers using a labeled training dataset [59]. In an intrusion detection system, classifiers are trained using a labeled dataset, so that, the classifiers learn the characteristics of packets that belong to an attack traffic, and those of the normal traffic. Then, this knowledge is used to classify new packets in order to detect attacks packets in order to block them from passing through the network, in order to protect that network from external intrusions.

According to the importance of fast decision provided to the firewall, to reduce the time required to process each packet and maintain the performance of the services provided on the network, each classifier is used to provide two types of decisions, the first decision uses binary predictions to instruct the firewall whether to allow the packet to access the network or deny it. The other decision is the type of attack, in case the first prediction is an attack packet. An illustration of the hierarchy of the proposed system is shown in Figure 4.3.

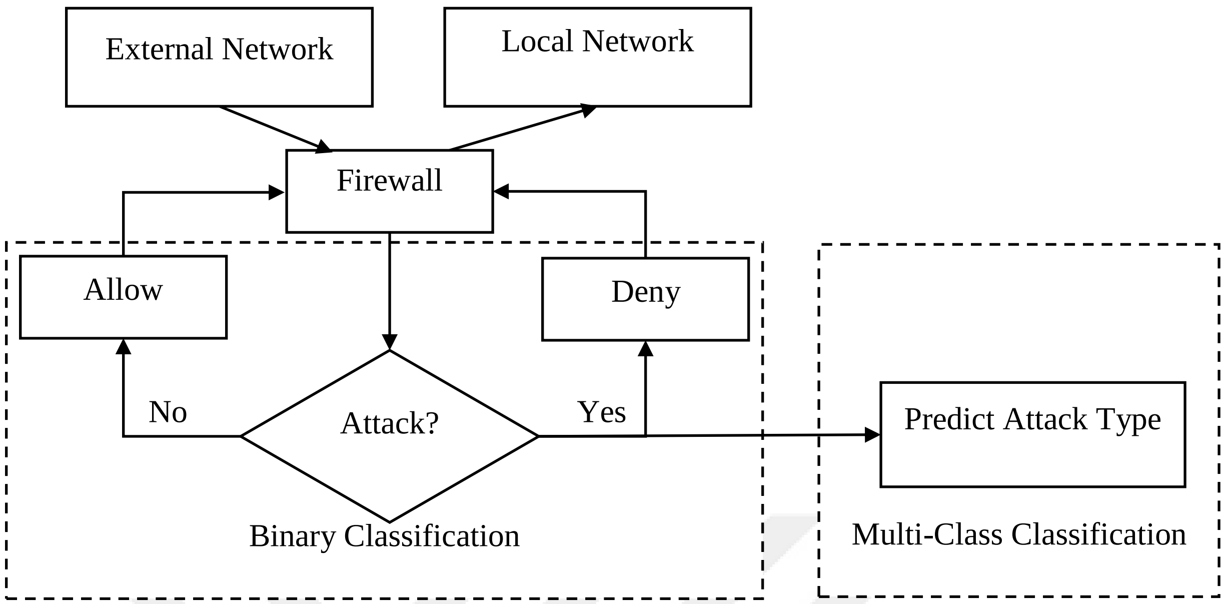

Figure 4.3: Hierarchy of the proposed intrusion detection system.

4.3.1 Using Support Vector Machine Classifier

The preprocessed data of the network traffic are fed to the SVM classifier that distributes the tuples in a multi-dimensional space, where the coordinates of these points are the attributes' values of the tuples corresponding to that point, as described in section 3.6.2.1. As the values in the dataset are normalized per each attribute, the values vary from zero to one. Thus, each axis in the space has a length of one unit that extends from zero to one. After the distribution of these points, the SVM classifier finds the optimal hyperplanes that split these points into groups, trying to keep the points of the same label in a sperate group. However, according to the close attributes' values of the tuples in different classes, it is quite difficult to achieve such goal. The attributes' values are close according to the methodologies used behind different types of attacks, which attempt to keep the traffic packets as similar to normal packet as possible using different approaches. Moreover, when a label is required for a new datum, the SVM classifier projects the corresponding point to the multi-dimensional space that is split with the hyperplanes concluded during the training phase of the classifier and examines the position that the point belongs to, in order to predict a label for that datum. The confidence of the prediction depends on how far the projected point is from the hyperplanes that split the domain space [60]. Based on these predictions, a decision is made for each packet, depending on the packet information received from the firewall. These decisions of whether to allow each of the packets of accessing the network or denying it. These decisions are forwarded back to the firewall in order to execute that decision.

4.3.2 Using Random Forest Classifier

As illustrated in section 3.6.2.2, random forest classifier is based on predictions made by multiple decision trees. Each decision tree is a group of IF/THEN clauses that are distributed in levels, where the top level consists of one condition and is known as the root of the tree. These conditions are generated during the training of the classifier, so that, when a new prediction is required for a new datum, the attributes' values of that datum are compared against these rules in order to predict a label for it. Each tree in the forest is trained using a different set of tuples from the training dataset, so that, different paths are concluded for each label in order to provide more accurate classification. To classify a packet, the packet information is sent to the random forest classifier after preprocessing it, so that the computed values are passed through the forest generated during the training. The final prediction from the random forest is based on the prediction of all the trees in that forest, where the dominant label in the predictions is selected as the predicted label for that packet [61]. Based on that prediction, a decision is made for that packer, whether to allow or deny access to the network behind the firewall. This decision is sent back to the firewall for execution.

4.3.3 Using Deep Learning Classifier

The use of more than one hidden layer in neural networks generates what is known as deep neural networks, where the multiple hidden layers allow the detection of more complex features by combining features from the previous layer. The number of neurons in the input layer is equal to the number of attributes that characterize each tuple, while the number of neurons in the output layer is equal to number of labels in the training dataset. The labels are one-hot encoded, so that, the label of each tuple consists of more than values, where all these values are zeros except one of these values is set to one, which corresponds the label given to that tuple. The output value of each neuron in a layer is computed by passing the summation of weighted inputs into an activation function as shown in section 3.6.2.3.

When tuples are labeled with individual labels, i.e. one label per each tuple, the SoftMax activation function is used at the output layer, where this activation function uses equation 4.2 to compute the probability *σ(z)* of the tuple being in a certain class out of *K* classes. Using SoftMax activation function, the summation of the probabilities computed at the output of the neural network is one [62].

$$\sigma(z) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{4.2}$$

4.4 PERFORMANCE EVALUATION

As there are many data mining algorithms that can be used to classify tuples, and as these algorithms have different performance on different data, it is important to evaluate their performance in order to find the best classifier for the IDS. A classifier may outperform another classifier on a certain dataset, however, the other classifier may outperform that classifier when used on another dataset. The classifier’s performance may be described by the accuracy of the predictions provided by the classifier and time taken to predict a class for a tuple [63]. The accuracy of a classifier is the number of correct prediction to the total number of predictions made by the classifier. Moreover, in IDS it is important to measure the rate of normal packets that are blocked by wrong predictions, which is known as the False Alarm Rate (FAR). Denying legitimate users from accessing degrades the quality of service provided by the servers on the network, which is against the main aim of the IDS. To compute the accuracy of a classifier, it is important to use data that are not included in the training dataset. However, it is also mandatory to use labeled data, so that, the classifier’s predictions are compared to the actual labels in order to compute the accuracy of that classifier [64]. For this reason, the dataset is split into two parts, one for the training purpose and the other is for testing the performance of the classifier, so that, the data used for testing are labeled data that are not included in the training, for accurate performance evaluation.

Another important performance measure is the F1 score, which is computed based in the weighted average of the precision and the recall of each class in the dataset. The F1 score for a specific class is computed using Equation 4.3, while the overall weighted F1 score is computed using Equation 4.4, where S_c is the support of that class, which is the number of tuples that belong to that class in the dataset. The recall represents the ratio of the correctly classified inputs to the total number of inputs that originally belong to that class in the dataset. Precision, on the other hand, represents the ratio of the correctly classified inputs of a certain class to the total number of inputs that are predicted by the classifier to be in that class.

$$F_c = 2 * \frac{precision * recall}{precision + recall} \tag{4.3}$$

$$F = \frac{\sum_c F_c * S_c}{Total\ Tuples\ Count} \tag{4.4}$$

4.5 DATA DESCRIPTION

The generation of the UNSW-NB15 dataset utilizes the IXIA PerfectStorm tool to collect a combination of normal and attack packets on the network. The traffic information is captured using Pcap files by a tcpdump tool, where 100 GB of raw traffic is collected. The dataset consists of 49 attributed that are collected by the parallel execution of Argus and Bro-IDS techniques, in addition to other twelve procedures. These attributes can be categorized into the following six categories:

1. **Flow Attributes:** This category includes the attributes that characterize and identify the connections established in the network, such as client to server and server to client connections.
2. **Basic Attributes:** This category includes the attributes that describe the connections, regarding the specifications of the protocol used for the communications.
3. **Content Attributes:** In this group, the attributes of the transmission control protocol, internet protocol, and some hypertext transfer protocol information are stored.
4. **Time Attributes:** This group holds timing information, such as packets' start and end time, packets' arrival time, and TCP protocol's round-trip time.

5. **Additional Attributes:** This category includes few extra attributes that are collected for protocols' service protection and summary attributes that summarize the flow of the most recent 100 connections, based on the time sequence.
6. **Label Attributes:** Two attributes in the dataset hold labels of each record. The first attribute holds the type of the packet, to be a normal packet, or of a specific type of attacks, while the other attributed includes one of two values, one value represents normal traffic and the other represents attacks.

In addition to the normal packets, the UNSW-NB15 dataset includes nine types of attacks. These attacks are:

1. **Analysis:** Includes different attacks, which abuse the services provided by the webserver using the legitimate ports that these services are provided through. Examples are spam emails and HTML files scripts.
2. **Backdoor:** These attacks authenticate unauthorized remote host by bypassing the server's authentication. The access in this type of attacks is not achieved using a different point of entry than that dedicated for the legitimate users to authenticate.
3. **DoS:** This attack aims to keep the server as busy as possible in order to deny the legitimate users of accessing the intended services, as the resources of the server are consumed by the services requests being sent by the attackers.
4. **Exploit:** This type of attacks makes use of glitches, vulnerabilities or bugs on the server to execute a series of commands to cause unexpected behavior by the host of the server under attack.
5. **Fuzzers:** The intruders, in this attack, send massive amount of random data to the server, in an attempt to discover any loopholes in the network or the operating systems of the servers, as well as the applications that provide the intended services.
6. **Generic:** In this attack, the intruders attempt to attack every black-cipher by causing collisions by using a hash function, regardless of the clock-cipher's configurations.
7. **Reconnaissance:** Information is gathered about the network and the computers in the network, so that, the attacker gains the ability to evade the security controls of the network.
8. **Shellcode:** A slight piece of code is injected into the server starting from a shell, this code is then used to control the attacked computer.

9. **Worm:** In this attack, the attacker uses some transportation media, usually the network, to replicate itself on other devices. The aim of attacks in this category may be different, however, the method used to spread these attacks defines this category.

5. RESULTS AND DISCUSSION

5.1 INTRODUCTION

In order to evaluate the performance of the classifiers used for intrusion detection, these classifiers are tested using the UNSW-NB15 dataset. These experiments are conducted using Python programming language [65] using a computer with Intel® Core™ i7-4500HQ CPU running at 1.80GHz with a memory of 16GB and 4GB of memory in the Graphical Processing Unit (GPU) running windows 10© operating system.

5.2 EXPERIMENTAL RESULTS

The proposed system is tested using three classifiers, which are the SVM, Random Forest and feed-forward deep learning classifier. Each packet is classified using the binary classification first, where packets predicted as attacks are classified using the multi-class classifier, in order to predict the type of attack that the packet belongs to. The predictions of the binary classification are used to make a decision per each packet, whether to allow access to the network, or deny it. The evaluation is performed using five folds cross-validation for both binary and multi-class classification for the UNSW-NB15 network traffic dataset.

5.2.1 Support Vector Machine Classifier

The performance of the SVM classifier in an intrusion detection system is tested to provide binary and multi-class predictions. Table 5.1 shows the confusion matrix of the classification results. The time required by the SVM classifier to predict a class per each packet is 218.27 *u*Sec.

Table 5.1: Confusion matrix of binary classification results for the SVM classifier.

		Predicted	
		Attack	**Normal**
Actual	**Attack**	319892	1391
	Normal	31700	2187064

The summary of the performance measures for the SVM classifier is shown in Table 5.2, where the accuracy of the predictions provided by this classifier in binary classification is 98.70%. These measures are also illustrated in Figure 5.1.

Table 5.2: Summary of the performance measures for the SVM classifier in binary classification.

	Precision	Recall	F1-Score	Accuracy	Support
Attack	0.9098	0.9957	0.9508	0.9957	321283
Normal	0.9994	0.9857	0.9925	0.9857	2218764
Avg/Total	**0.9880**	**0.9870**	**0.9872**	**0.9870**	**2540047**

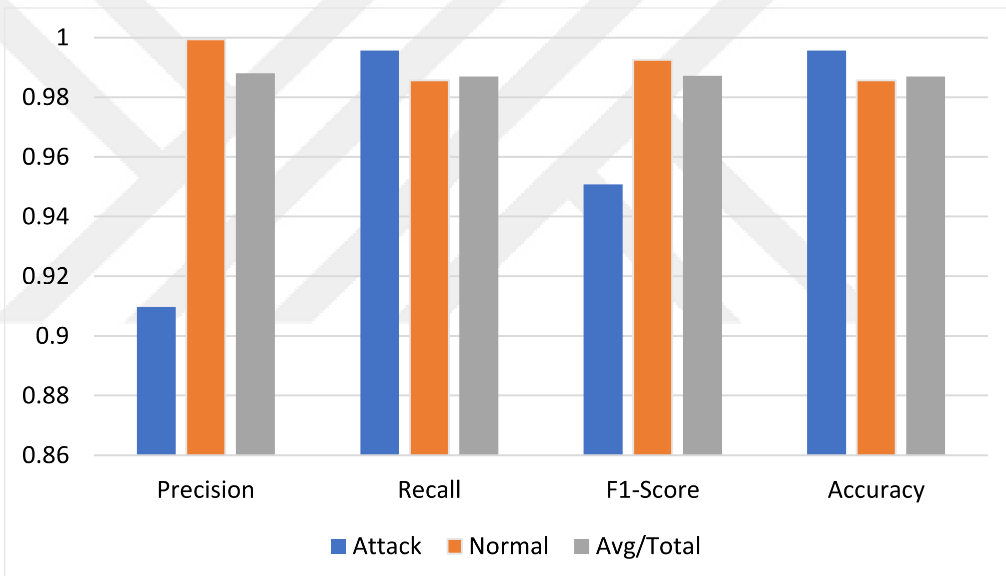

Figure 5.1: Illustration of the SVM classifier performance in binary classification.

The attack packets are extracted from the training dataset per each fold and as used to train a multi-class SVM classifier, so that, the packets that are predicted as attacks by the binary classifier are fed to the multi-class classifier in order to predict the type of the attack. The confusion matrix of the multi-class classification results is shown in Table 5.3.

Table 5.3: Confusion matrix of multi-class classification results for the SVM classifier.

		Predicted								
		Analysis	Backdoors	DoS	Exploits	Fuzzers	Generic	Reconnaissance	Shellcode	Worms
Actual	Analysis	2090	0	0	511	38	16	0	0	0
	Backdoors	2109	0	0	76	67	20	24	0	0
	DoS	13194	0	13	2413	298	129	45	0	0
	Exploits	21414	0	1	20787	1407	179	86	0	0
	Fuzzers	9468	0	0	313	13925	150	69	0	0
	Generic	3061	0	1	1920	430	209969	23	0	0
	Normal	8323	0	3	4714	18435	97	128	0	0
	Reconnaissance	11743	0	2	370	886	104	856	0	0
	Shellcode	1274	0	0	1	152	0	84	0	0
	Worms	91	0	0	44	10	28	1	0	0

The performance measures of the SVM classifier when used in multi-class classification are summarized in Table 5.4 and illustrated visually in Figure 5.2. The accuracy of the predictions provided by the SVM classifier is 70.43%, while the average time required to classify each packet is 709.65 *u*Sec.

Table 5.4: Summary of the performance measures for the SVM classifier in multi-class classification.

	Precision	Recall	F1-Score	Accuracy	Support
Analysis	0.0287	0.7872	0.0554	0.7872	2655
Backdoors	0	0	0	0	2296
DoS	0.65	0.0008	0.0016	0.0008	16092
Exploits	0.6673	0.4738	0.5542	0.4738	43874
Fuzzers	0.3906	0.582	0.4675	0.582	23925
Generic	0.9966	0.9748	0.9855	0.9748	215404
Normal	0	0	0	0	31700
Reconnaissance	0.6505	0.0613	0.1121	0.0613	13961
Shellcode	0	0	0	0	1511
Worms	0	0	0	0	174
Avg/Total	**0.7762**	**0.7043**	**0.7097**	**0.7043**	**351592**

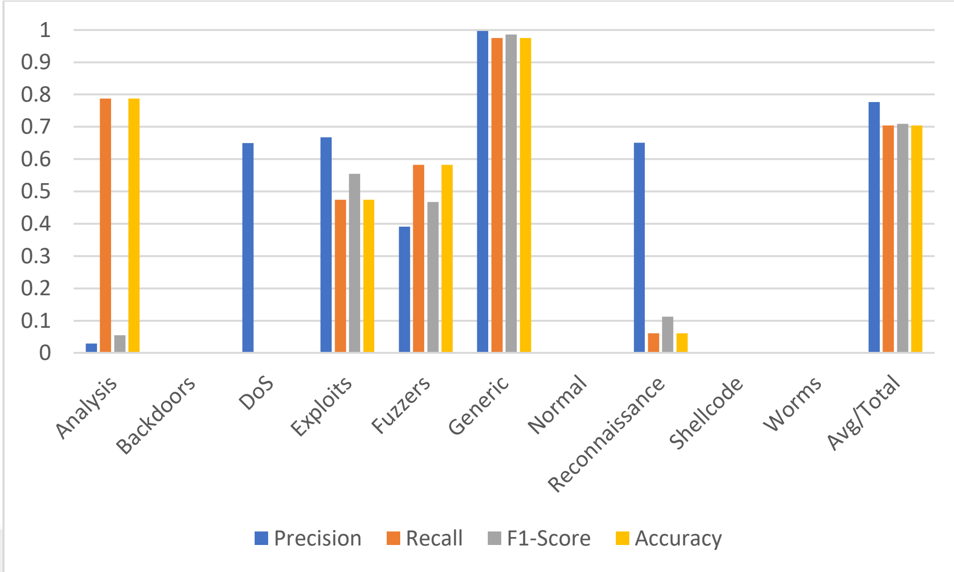

Figure 5.2: Illustration of the SVM classifier performance in multi-class classification.

5.2.2 Random Forest Classifier

In this experiment, the performance of the Random Forest classifier in an intrusion detection system is tested. First, the classifier provides a binary classification for the packets, in order to allow or deny them, then, the attack packets are classified in order to predict the type of attack being executed. Both classifiers use 100 trees in the forest, where each tree provides a prediction and the dominant class predicted by these trees is selected as the predicted class by the forest. The binary classification results are summarized in the confusion matrix shown in Table 5.5.

Table 5.5: Confusion matrix of binary classification results for the Random Forest classifier.

		Predicted	
		Attack	**Normal**
Actual	**Attack**	316041	5242
	Normal	4966	2213798

The performance measures of the Random Forest classifier are shown in Table 5.6. and illustrated in Figure 5.3, where the accuracy of the classifier in binary classification is 99.60% and each packet required an average of 8.54 *u*Sec.

Table 5.6: Summary of the performance measures for the Random Forest classifier in binary classification.

	Precision	Recall	F1-Score	Accuracy	Support
Attack	0.9845	0.9837	0.9841	0.9837	321283
Normal	0.9976	0.9978	0.9977	0.9978	2218764
Avg/Total	**0.9960**	**0.9960**	**0.9960**	**0.9960**	**2540047**

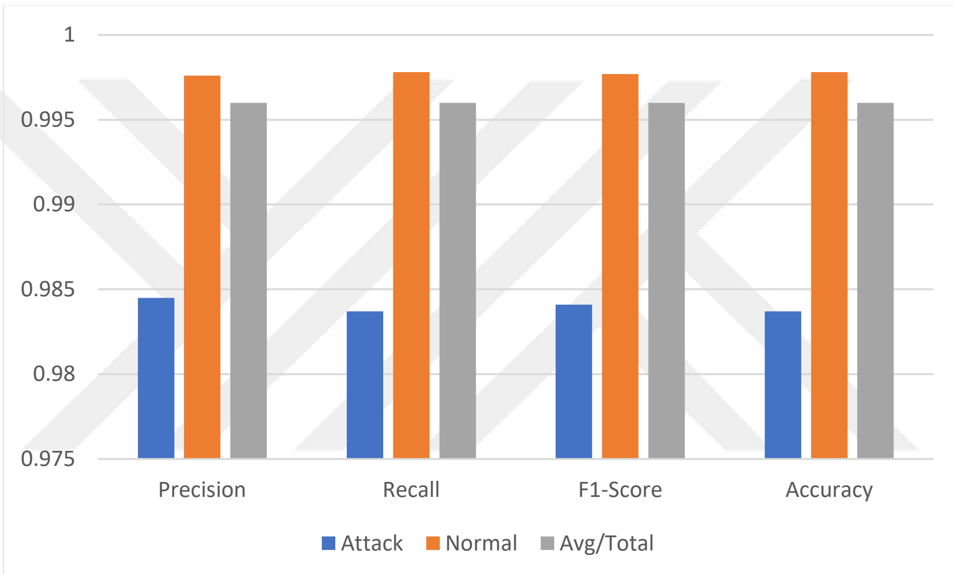

Figure 5.3: Illustration of the Random Forest classifier performance in binary classification.

The packets predicted by the binary classifier to be attack packets are classified by the multi-class classifier to predict the type of attack that each packet is a part of. The training dataset of the multiclass classifier is the part of the entire dataset that includes actual attack packets. The confusion matrix of the binary classification results is shown in Table 5.7.

Table 5.7: Confusion matrix of multi-class classification results for the Random Forest classifier.

		Predicted								
		Analysis	**Backdoors**	**DoS**	**Exploits**	**Fuzzers**	**Generic**	**Reconnaissance**	**Shellcode**	**Worms**
Actual	**Analysis**	311	3	549	1327	229	25	1	0	0
	Backdoors	3	225	551	1280	238	7	10	15	0
	DoS	6	7	4121	11447	378	113	89	142	2
	Exploits	9	22	5442	36399	1108	291	715	158	17
	Fuzzers	7	15	586	2178	16801	39	24	100	0
	Generic	26	20	653	1933	162	212560	18	72	3
	Normal	22	7	31	649	4098	20	32	106	1
	Reconnaissance	1	8	731	2417	39	12	10752	14	0
	Shellcode	0	0	20	190	178	23	10	1035	0
	Worms	0	0	2	119	8	8	0	3	34

The performance measures of the Random Forest classifier, when used to predict the type of the attack that a packet comes from, are shown in Table 5.8. The classification accuracy of the multi-classification for the Random Forest classifier is 87.92% and the average time required to predict an attack type for each packet is 17.28 *u*Sec. Figure 5.4 illustrates the performance of the Random Forest classifier visually.

Table 5.8: Summary of the performance measures for the Random Forest classifier in multi-class classification.

	Precision	Recall	F1-Score	Accuracy	Support
Analysis	0.8078	0.1272	0.2198	0.1272	2445
Backdoors	0.7329	0.0966	0.1707	0.0966	2329
DoS	0.3248	0.2527	0.2843	0.2527	16305
Exploits	0.6282	0.8242	0.713	0.8242	44161
Fuzzers	0.723	0.8507	0.7816	0.8507	19750
Generic	0.9975	0.9866	0.992	0.9866	215447
Normal	0	0	0	0	4966
Reconnaissance	0.9228	0.7694	0.8392	0.7694	13974
Shellcode	0.6292	0.7109	0.6675	0.7109	1456
Worms	0.5965	0.1954	0.2944	0.1954	174
Avg/Total	**0.8717**	**0.8792**	**0.869**	**0.8792**	**321007**

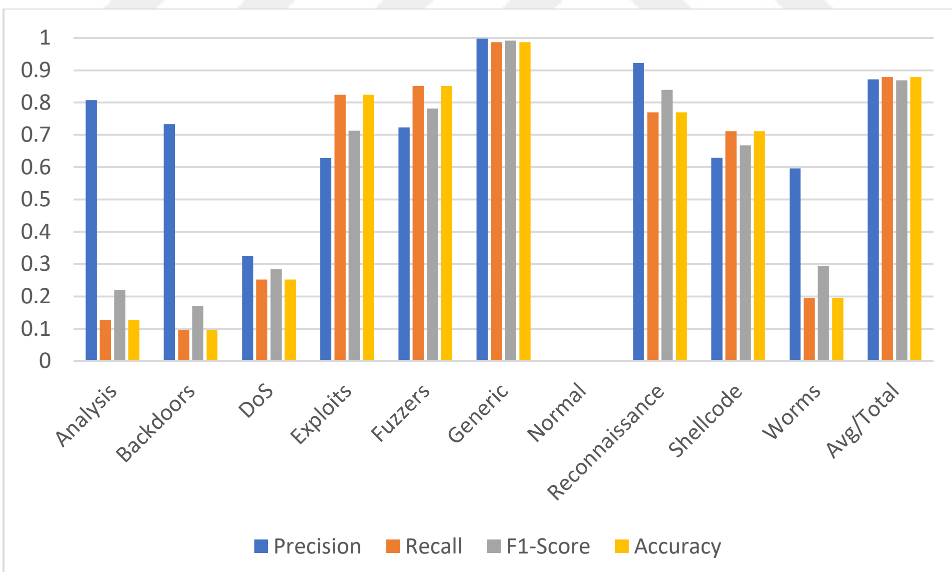

Figure 5.4: Illustration of the Random Forest classifier performance in multi-class classification.

5.2.3 Deep Learning Artificial Neural Network Classifier

A simple, yet effective, deep artificial neural network is implemented to classify the packets into two classes, in binary classification. The packets classified as attacks are forwarded to another deep artificial neural network to predict the type of attack that the packet belongs to. The model implemented for binary classification consists of five layers, which are one input layer with 43 neurons, three hidden layers with 128,64 and 32 neurons sequentially, while the output layer has one neuron, where a prediction of one for normal packets and zero for the attack ones. The neuron in the output layer has a sigmoid activation function, which is the function normally used with binary classification, while all neurons in the other layers use ReLU activation function. The confusion matrix shown in Table 5.9 summarizes the classification results of the deep feed-forward artificial neural network used in binary classification.

Table 5.9: Confusion matrix of binary classification results for the deep artificial neural network classifier.

		Predicted	
		Attack	**Normal**
Actual	**Attack**	311220	10063
	Normal	8593	2210171

The performance of the deep feed-forward artificial neural network in binary classification is illustrated by the measure in Table 5.10 and visually in Figure 5.5. The measures show that the accuracy of the binary predictions of this classifier is 99.27%. The average time required by the classifier to produce a prediction for each packet is 0.70 *u*Sec.

Table 5.10: Summary of the performance measures for the deep artificial neural network classifier in binary classification.

	Precision	Recall	F1-Score	Accuracy	Support
Attack	0.9731	0.9687	0.9709	0.9687	321283
Normal	0.9955	0.9961	0.9958	0.9961	2218764
Avg/Total	**0.9926**	**0.9927**	**0.9926**	**0.9927**	**2540047**

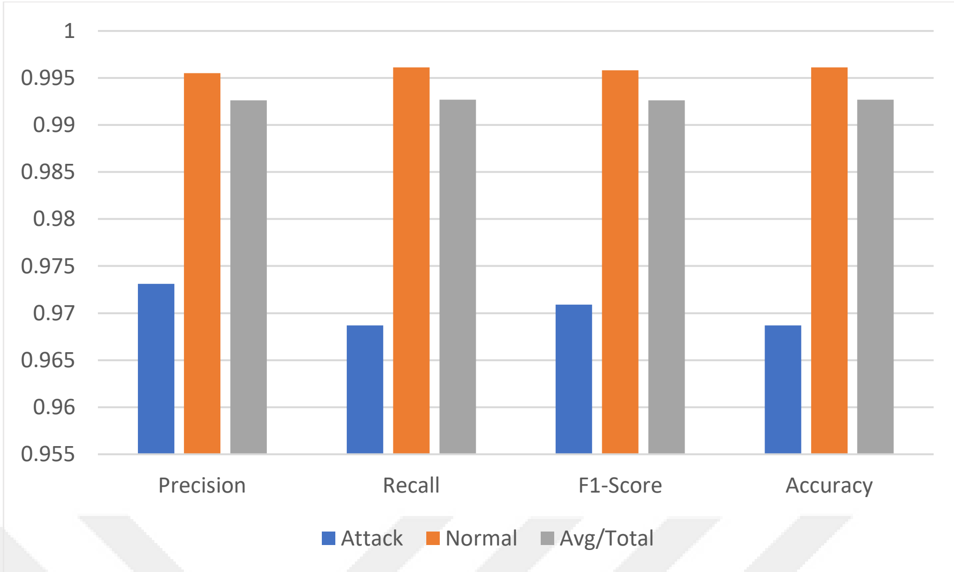

Figure 5.5: Illustration of the artificial neural network classifier performance in binary classification.

Packets predicted to by the binary classifier as attack packets are fed to another deep feed-forward neural network that predicts the type of the attack that the packet comes from. Beside the output layer, the multi-class network is identical to the one used in binary classification, where the output layer consists of ten neurons with softmax activation function, which is the activation function normally used for multi-class predictions. The results are summarized in the confusion matrix shown in Table 5.11.

Table 5.11: Confusion matrix of multi-class classification results for the artificial neural network classifier.

		Predicted								
		Analysis	Backdoors	DoS	Exploits	Fuzzers	Generic	Reconnaissance	Shellcode	Worms
Actual	Analysis	2119	0	0	113	6	20	0	0	0
	Backdoors	2075	39	9	151	19	7	8	7	0
	DoS	662	5	12574	2513	171	185	73	68	0
	Exploits	654	3	148	41562	691	449	412	111	3
	Fuzzers	3116	0	9	335	11846	25	182	60	0
	Generic	2222	3	67	919	220	211930	37	26	2
	Normal	830	1	10	1081	6186	26	381	77	1
	Reconnaissance	2395	5	19	1362	181	13	9731	14	0
	Shellcode	401	0	1	120	156	13	140	640	0
	Worms	12	0	1	123	10	5	0	0	22

The performance measures of the multi-class results are shown in Table 5.12 and illustrated, visually, in Figure 5.6. The accuracy of the prediction in this experiment is 90.82% and the average time per each prediction is 0.89 *u*Sec.

Table 5.12: Summary of the performance measures for the artificial neural network classifier in multi-class classification.

	Precision	Recall	F1-Score	Accuracy	Support
Analysis	0.1463	0.9384	0.2531	0.9384	2258
Backdoors	0.6964	0.0168	0.0329	0.0168	2315
DoS	0.9794	0.7737	0.8645	0.7737	16251
Exploits	0.8609	0.9439	0.9005	0.9439	44033
Fuzzers	0.6079	0.7607	0.6758	0.7607	15573
Generic	0.9965	0.9838	0.9901	0.9838	215426
Normal	0	0	0	0	8593
Reconnaissance	0.8875	0.7093	0.7884	0.7093	13720
Shellcode	0.6381	0.4351	0.5174	0.4351	1471
Worms	0.7857	0.1272	0.2189	0.1272	173
Avg/Total	**0.9167**	**0.9082**	**0.9061**	**0.9082**	**319813**

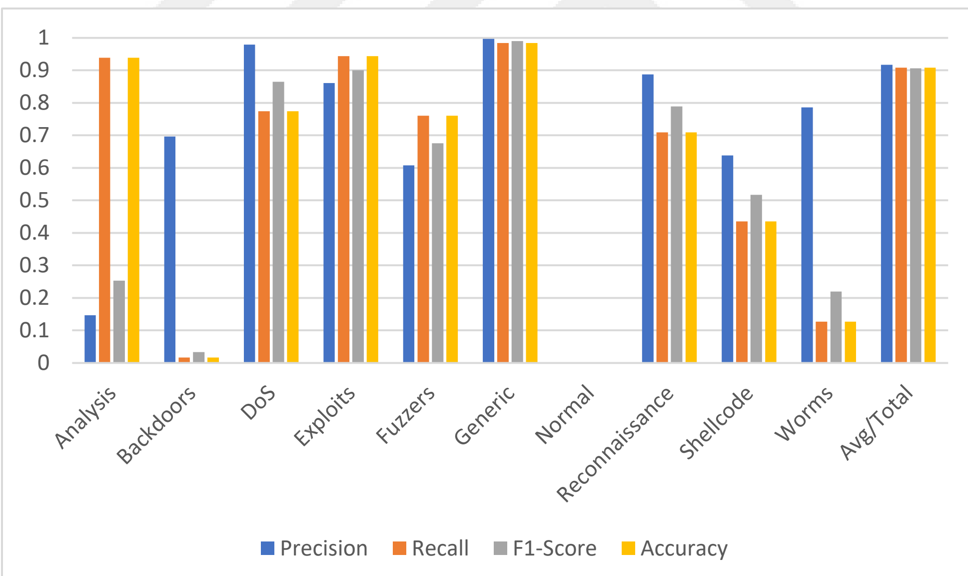

Figure 5.6: Illustration of the artificial neural network classifier performance in multi-class classification.

5.3 RESULTS SUMMARY AND DISCUSSION

The summary of the performance evaluation results is shown in Table 5.13. The results show that the Random Forest classifier has the highest accuracy among the tested classifiers in binary classification, while the SVM has the least. Moreover, the difference between the Deep Learning neural network and the Random Forest is marginal, regarding the accuracy of the binary predictions made by these classifiers, with 99.60% for the Random Forest and 99.27% for the deep learning model, which shows that the Random Forests classifier is only 0.33% more accurate. Moreover, the false alarm rate of the Random Forest classifier, which is 0.24%, is only 0.21% less than that of the Deep Learning neural network classifier, which is 0.45%. However, the average time taken by the Random Forest classifier to predict a class for a tuple is 8.54 *u*Sec, which is relatively high compared to that taken by the Deep Learning neural network classifier, which is 0.7 *u*Sec. Thus, the Deep Learning neural network classifier is 1220% faster than the Random Forest classifier. Figure 5.7 also illustrates the summary of the classifiers' performance through all the experiments conducted during the study. For better illustration, the values are normalized per each category, according to the maximum value in that category.

Table 5.13: Summary of the experimental results.

	Accuracy (%)			**F1 Score (%)**			**FAR (%)**	**Prediction Time (*u*Sec)**		
	Binary	**Multi-Class**	**Average**	**Binary**	**Multi-Class**	**Average**		**Binary**	**Multi-Class**	**Average**
SVM	98.70	70.43	84.57	98.72	70.43	84.58	**0.06**	218.27	709.65	463.96
RF	**99.60**	87.92	93.76	**99.60**	86.90	93.25	0.24	8.54	17.28	12.91
DL	99.27	**90.82**	**95.05**	99.26	**90.61**	**94.94**	0.45	**0.70**	**0.89**	**0.80**

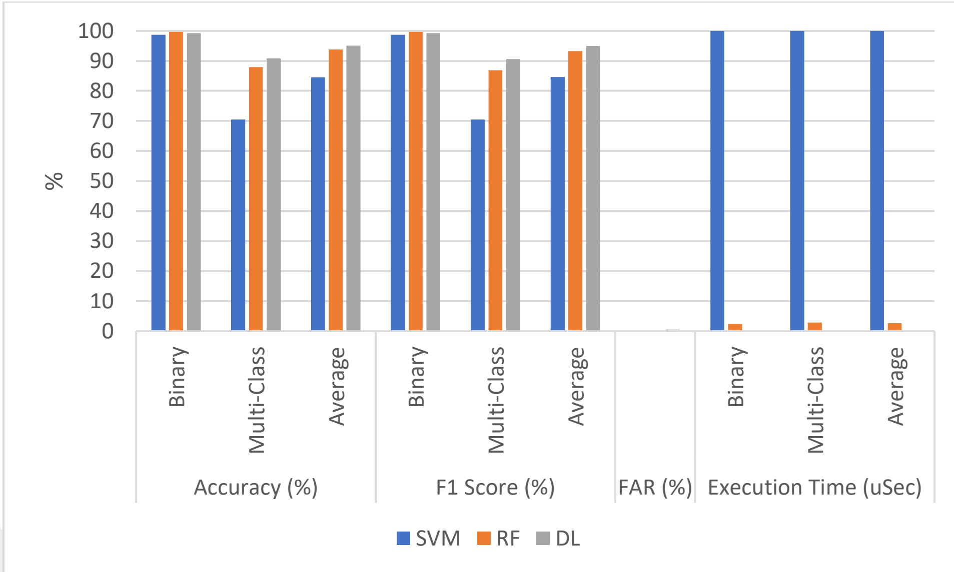

Figure 5.7: Illustration of the performance summary for the classifiers evaluated in the study.

Table 5.14 summarizes the results from earlier studies that use binary classification to detect intrusion packets from normal, where different techniques are tested using the UNSW-NB15 networks traffic dataset. The comparison shows that the proposed method has a better performance than those in the earlier studies. The table shows the techniques used in the proposed IDS systems as well as the methods evaluated in this study, where the comparison shows that the Deep Learning neural network and the Random Forest classifiers have outperformed the techniques proposed in earlier studies for binary classification, where the highest accuracy achieved in the earlier studies is achieved by Malek Al-Zewairi, et al.[25], which is 98.99%.

Table 5.14: Summary of the earlier binary classification studies' results.

Study	Technique	Accuracy (%)
Mustapha Belouch, et al. [19]	Decision Tree	86.13
	Artificial Neural Network	86.31
	Naïve Bayes	80.04
	Random Tree	86.59
	RepTree	87.80
Nour Mustafa and Jill Slay [23]	Expectation-Maximization	77.20
	Linear Regression	83.00
	Naïve Bayes	79.50
Primartha and Tama [24]	Random Forest	95.5
	Multilayer Perceptron	83.50
Malek Al-Zewairi, et al. [25]	Deep Learning	**98.99**
This Study	SVM	98.70
	Random Forest	**99.60**
	Deep Learning	99.27

Moreover, Table 5.15 summarizes the earlier studies that use multi-class classification to predict the type of intrusion that the network is being attacked with and the results of the methods tested in this study. The results show that the Random Forest and Deep Learning classifiers have outperformed other classifiers used for multi-class classification is earlier studies. Although the method proposed in Gharaee and Hosseinvand [20] have a higher accuracy than the methods tested in this study, their method classifies attacks into only seven classes, instead of using all the classes in the dataset. However, the earlier studies that use all the classed in the dataset to classify the attacks has less accuracy measures than those evaluates in this study, where the highest multi-class classification accuracy is achieved by the ReTree method proposed by Mustapha Belouch, et al. [19], with an accuracy of 79.20%, while the Random Forest and Deep Learning methods evaluated in this study have scored accuracies of 87.92% and 90.82%, respectively.

Table 5.15: Summary of the earlier multi-class classification studies' results.

Study	Technique	Accuracy (%)
Mustapha Belouch, et al. [19]	Artificial Neural Network	78.14
	Naïve Bayes	73.86
	Random Tree	76.21
	RepTree	**79.20**
Gharaee and Hosseinvand [20]	Genetic + SVM[1]	**93.25**
This Study	SVM	70.43
	Random Forest	87.92
	Deep Learning	**90.82**

The results of the tested methods shows high performance compared to those tested by Nour Mustafa and Jill Slay [23], where the proposed hybrid method uses linear regression to compute the probability of incoming packets to be a part of normal traffic or of a specific kind of an attack, which has an accuracy of 83% when tested with the UNSW-NB15 network traffic dataset. The linear regression hybrid method is also compared to two other methods, which are the Expectation-Maximization and Naïve Bayes classifiers, and has shown relatively higher performance, where the Expectation-Maximization has scored an accuracy of 77.20% and the Naïve Bayes has scored 79.50% accuracy when tested on the same dataset.

[1] Tuples are classified into only seven classes, one normal and six attack types.

Moreover, the performance of the Random Forest classifier in this study has outperformed the performance of the same classifier tested by Rifke Primartha and Bayu Adhi Tama [24], which uses a random forest classifier with 800 decision trees and has scored an accuracy of 95.5% and FAR of 7.22%. This comparison illustrates the importance of data preprocessing, where the results are improved despite the fact the implemented model has only 100 decision trees in the Random Forest.

The performance of the Deep Learning neural network classifier has also shown relatively better performance than the model implemented by Malek Al-Zewairi, et al. [25], which has five hidden layers and has achieved a prediction accuracy of 98.99%, where the Deep Learning neural network classifier in this study has scored an accuracy of 99.27% in the binary classification, using only three hidden layers. Al-Zewairi et al. have also compared the Deep Learning neural network classifier to many other classifiers, where all other classifiers have shown less prediction accuracy than the Deep Learning model.

6. CONCLUSION

The rapid growth of internet usage to access different services provided online has emerged the need to protect the servers that provide these services from any attempts to compromise the quality of the services provided by these servers and the security of the information stored on them. The evolution of the techniques used by the intruders to attack these servers are getting more complicated, so that, it is becoming more and more difficult to distinguish packet coming from normal traffic or attacks. Thus, machine learning techniques are employed in the implementation of intrusion detection systems that have the ability to detect the anomaly in the network traffic and deny access to such traffic.

Classification techniques are supervised machine learning techniques that investigate the relations between the values that characterize an object and the label given to that object. These relations are, then, used to predict labels for new objects in order to estimate their future behavior depending on the label predict for each object and the general behavior of objects that have similar behavior. Moreover, there are many classification techniques that have different approaches in extracting these relations and creating models that represent the extracted knowledge, to be used for predictions. Thus, different classifiers may have different performance depending on the dataset used to train them and it is important to evaluate the performance of the classifiers in order to select the one with the best performance when used with the required dataset.

As the classifiers are used to provide predictions, and it is important to evaluate their performances, labeled data is required to evaluate the performance of the classifier by comparing the predictions made by the classifier for that dataset to the actual labels of the object in that dataset. Splitting the dataset into training and testing data may result in biased evaluation, where that split may be suitable for one classifier rather than the other. Thus, cross-validation is used to provide more realistic performance measure, where the dataset is divided into bins and the classifier is used to iterate through all these bins. Per each iteration, one of the bins is used as the testing dataset, while the remaining bins are used for training. By the end of the iterations, the average performance measures are used to describe the performance of the classifier.

Many network traffic data are collected to generate databases that can be used to train classifiers to be used in intrusion detection systems. However, most of these datasets have drawbacks that restrict their use to train classifiers to be used in such system. Moreover, the UNSW-NB15 dataset is a recent dataset that has packet information for different types of network traffic. Some of these packets are of normal traffic, while other are of intrusion packets. Nine types of intrusions are included in this dataset in addition to the normal traffic packets. Thus, this dataset is most appropriate dataset for that purpose.

In this study, the performance of three classifiers, which are the Support Vector Machine, Random Forest and Feed-Forward Deep Neural Network, to be used in an intrusion detection system is evaluated using the UNSW-NB15 network traffic dataset and five-fold cross-validation. The proposed intrusion detection system uses a classifier for two purposes. First, the classifier is used to provide binary predictions for the packets in the dataset, where each packet is classified to be a normal or an attack packet. A decision is made based on this prediction whether to grant the packet access to the network or deny. Then, if the packet is predicted to be an attack, the type of the intrusion being executed against the server is predicted using a multi-class classifier.

In binary classification, the Random forest classifier has scored the highest performance measures, with 99.60% predictions accuracy, while the SVM and deep learning classifiers have scored 98.7% and 99.27%, respectively. However, as these predictions are used to grand, or deny, access to the network, it is important to provide faster decisions to reduce the network latency and maintain the quality of the services provided on the network, where the deep learning classifier has consumed significantly less time to provide predictions for the incoming traffic, which only an average of 0.7 *u*Sec per each prediction, while the Random Forest and SVM classifier have consumed an average of 8.54 and 218.3 *u*Sec, respectively. Thus, as the deep learning classifier has a marginal difference in accuracy and significant difference in time consumption, it is recommended to be used in the intrusion detection system.

Moreover, in intrusion type detection , the deep learning multi-class classifier have shown the highest predictions accuracy with 90.82%, compared to the Random Forest, which has scored 87.92%, and the SVM classifier that scored 70.43% accuracy. The deep learning has also provided faster predictions with an average of only 0.89 *u*Sec per each packet, while the Random Forest has consumed 17.28 *u*Sec and the SVM classifier has consumed 709.65 *u*Sec. Thus, the deep learning classifier has shown the best performance regarding attack type prediction in both accuracy and speed of prediction.

In future work, techniques to embed the deep learning classifier in the firewall hardware are tested, so that, a standalone device can be used to filter out intrusion packets without the need of external computers, to increase the efficiency and reduce power consumption.

REFERENCES

[1] D. Acemoglu, A. Malekian, and A. Ozdaglar, "Network security and contagion," *Journal of Economic Theory*, vol. 166, pp. 536-585, 2016.

[2] D. Yu, Y. Jin, Y. Zhang, and X. Zheng, "A survey on security issues in services communication of Microservices-enabled fog applications," *Concurrency and Computation: Practice and Experience*, p. e4436.

[3] S. Henningsen, S. Dietzel, and B. Scheuermann, "Challenges of Misbehavior Detection in Industrial Wireless Networks," in *Ad Hoc Networks*, ed: Springer, 2018, pp. 37-46.

[4] V. C. Storey and I.-Y. Song, "Big data technologies and Management: What conceptual modeling can do," *Data & Knowledge Engineering*, vol. 108, pp. 50-67, 2017.

[5] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*: Morgan Kaufmann, 2016.

[6] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19-31, 2016.

[7] N. Pandeeswari and G. Kumar, "Anomaly detection system in cloud environment using fuzzy clustering based ANN," *Mobile Networks and Applications*, vol. 21, pp. 494-505, 2016.

[8] G. P. Gupta and M. Kulariya, "A framework for fast and efficient cyber security network intrusion detection using apache spark," *Procedia Computer Science*, vol. 93, pp. 824-831, 2016.

[9] K. Cup, "Dataset," *available at the following website http://kdd. ics. uci. edu/databases/kddcup99/kddcup99. html*, vol. 72, 1999.

[10] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-based systems*, vol. 78, pp. 13-21, 2015.

[11] S. Jaiswal, K. Saxena, A. Mishra, and S. K. Sahu, "A KNN-ACO approach for intrusion detection using KDDCUP'99 dataset," in *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on*, 2016, pp. 628-633.

[12] N. G. Relan and D. R. Patil, "Implementation of network intrusion detection system using variant of decision tree algorithm," in *Nascent Technologies in the Engineering Field (ICNTE), 2015 International Conference on*, 2015, pp. 1-5.

[13] V. D. Mane and S. Pawar, "Anomaly based ids using backpropagation neural network," *International Journal of Computer Applications,* vol. 136, 2016.

[14] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security (TISSEC),* vol. 3, pp. 262-294, 2000.

[15] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," in *Software, Knowledge, Information Management and Applications (SKIMA), 2014 8th International Conference on*, 2014, pp. 1-6.

[16] L. Dhanabal and S. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 4, pp. 446-452, 2015.

[17] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Signal Processing And Communication Engineering Systems (SPACES), 2015 International Conference on*, 2015, pp. 92-96.

[18] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Military Communications and Information Systems Conference (MilCIS), 2015*, 2015, pp. 1-6.

[19] M. Belouch, S. El Hadaj, and M. Idhammad, "A Two-Stage Classifier Approach using RepTree Algorithm for Network Intrusion Detection," *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS,* vol. 8, pp. 389-394, 2017.

[20] H. Gharaee and H. Hosseinvand, "A new feature selection IDS based on genetic algorithm and SVM," in *Telecommunications (IST), 2016 8th International Symposium on*, 2016, pp. 139-144.

[21] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences,* vol. 378, pp. 484-497, 2017.

[22] P. S. Bhattacharjee, A. K. M. Fujail, and S. A. Begum, "Intrusion detection system for NSL-KDD data set using vectorised fitness function in genetic algorithm," *Adv. Comput. Sci. Technol.*, vol. 10, pp. 235-246, 2017.

[23] N. Moustafa and J. Slay, "A hybrid feature selection for network intrusion detection systems: Central points," *arXiv preprint arXiv:1707.05505*, 2017.

[24] R. Primartha and B. A. Tama, "Anomaly detection using random forest: A performance revisited," in *Data and Software Engineering (ICoDSE), 2017 International Conference on*, 2017, pp. 1-6.

[25] M. Al-Zewairi, S. Almajali, and A. Awajan, "Experimental Evaluation of a Multi-layer Feed-Forward Artificial Neural Network Classifier for Network Intrusion Detection System," in *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, 2017, pp. 167-172.

[26] I. U. Din, S. Mahooz, and M. Adnan, "Performance evaluation of different Ethernet LANs connected by Switches and Hubs," *European Journal of Scientific Research*, vol. 37, pp. 461-470, 2009.

[27] A. E. Retana, A. C. Lindem III, and R. White, "Dynamically configuring and verifying routing information of broadcast networks using link state protocols in a computer network," ed: Google Patents, 2015.

[28] K. Gould and A. Danforth, "Method to block unauthorized network traffic in a cable data network," ed: Google Patents, 2016.

[29] S. Khan, A. Gani, A. W. A. Wahab, M. Shiraz, and I. Ahmad, "Network forensics: Review, taxonomy, and open challenges," *Journal of Network and Computer Applications*, vol. 66, pp. 214-235, 2016.

[30] J. M. Kizza, *Guide to computer network security*: Springer, 2017.

[31] R. Dabestani, A. Shahin, and M. Saljoughian, "Evaluation and prioritization of service quality dimensions based on gap analysis with analytic network process," *International journal of quality & reliability management*, vol. 34, pp. 530-548, 2017.

[32] T. P. Fries, "Classification of Network Traffic Using Fuzzy Clustering for Network Security," in *Industrial Conference on Data Mining*, 2017, pp. 278-285.

[33] A. Sahasrabuddhe, S. Naikade, A. Ramaswamy, B. Sadliwala, and P. Futane, "Survey on Intrusion Detection System using Data Mining Techniques," 2017.

[34] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Information Networking (ICOIN), 2016 International Conference on*, 2016, pp. 63-68.

[35] J. P. Anderson, "Computer security threat monitoring and surveillance," *Technical Report, James P. Anderson Company,* 1980.

[36] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering,* pp. 222-232, 1987.

[37] H. Orman, "The Morris worm: A fifteen-year perspective," *IEEE Security & Privacy,* vol. 99, pp. 35-43, 2003.

[38] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor," in *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, 1990, pp. 296-304.

[39] D. Dasgupta, "Immunity-based intrusion detection system: a general framework," in *Proc. of the 22nd NISSC*, 1999, pp. 147-160.

[40] M. Hayashi, M. Owari, G. Kato, and N. Cai, "Secrecy and robustness for active attack in secure network coding," in *Information Theory (ISIT), 2017 IEEE International Symposium on*, 2017, pp. 1172-1176.

[41] P. Sengar and N. Bhardwaj, "A Survey on Security and Various Attacks in Wireless Sensor Network," *International Journal of Computer Sciences and Engineering,* vol. 5, pp. 78-84, 2017.

[42] S. Latha and S. J. Prakash, "A survey on network attacks and Intrusion detection systems," in *Advanced Computing and Communication Systems (ICACCS), 2017 4th International Conference on*, 2017, pp. 1-7.

[43] E. M. Tzanakou, *Supervised and unsupervised pattern recognition: feature extraction and computational intelligence*: CRC Press, 2017.

[44] S. Liu and M. d'Aquin, "Unsupervised learning for understanding student achievement in a distance learning setting," in *Global Engineering Education Conference (EDUCON), 2017 IEEE*, 2017, pp. 1373-1377.

[45] E. Hancer and D. Karaboga, "A comprehensive survey of traditional, merge-split and evolutionary approaches proposed for determination of cluster number," *Swarm and Evolutionary Computation,* vol. 32, pp. 49-67, 2017.

[46] S. Suthaharan, "Big data classification: Problems and challenges in network intrusion prediction with machine learning," *ACM SIGMETRICS Performance Evaluation Review,* vol. 41, pp. 70-73, 2014.

[47] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 59-68.

[48] S. Suthaharan, "Support vector machine," in *Machine learning models and algorithms for big data classification,* ed: Springer, 2016, pp. 207-235.

[49] R. Lior, *Data mining with decision trees: theory and applications* vol. 81: World scientific, 2014.

[50] H.-R. Zhang and F. Min, "Three-way recommender systems based on random forests," *Knowledge-Based Systems,* vol. 91, pp. 275-286, 2016.

[51] R. J. Schalkoff, *Artificial neural networks* vol. 1: McGraw-Hill New York, 1997.

[52] M. Valipour, "Optimization of neural networks for precipitation analysis in a humid region to detect drought and wet year alarms," *Meteorological Applications,* vol. 23, pp. 91-100, 2016.

[53] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature,* vol. 521, p. 436, 2015.

[54] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE access,* vol. 2, pp. 514-525, 2014.

[55] M. Tiwari and R. Dixit, *Data Mining Principles, Process Model and Applications*: Educreation Publishing, 2017.

[56] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "Revision: Automated classification, analysis and redesign of chart images," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 393-402.

[57] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, and A. Graves, "Conditional image generation with pixelcnn decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790-4798.

[58] J. Quackenbush, "Microarray data normalization and transformation," *Nature genetics,* vol. 32, p. 496, 2002.

[59] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708-713, 2015.

[60] S. Dugad, V. Puliyadi, H. Palod, N. Johnson, S. Rajput, and S. Johnny, "Ship intrusion detection security system using image processing & SVM," in *Nascent Technologies in Engineering (ICNTE), 2017 International Conference on*, 2017, pp. 1-7.

[61] A. Mellor, S. Boukir, A. Haywood, and S. Jones, "Exploring issues of training data imbalance and mislabelling on random forest performance for large area land cover classification using the ensemble margin," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 105, pp. 155-168, 2015.

[62] Z. Yuan, J. Li, Z. Li, C. Ding, A. Ren, B. Yuan*, et al.*, "Softmax Regression Design for Stochastic Computing Based Deep Convolutional Neural Networks," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, 2017, pp. 467-470.

[63] J. A. Sáez, J. Luengo, and F. Herrera, "Evaluating the classifier behavior with noisy data considering performance and robustness: The Equalized Loss of Accuracy measure," *Neurocomputing*, vol. 176, pp. 26-35, 2016.

[64] H. Lu, R. Setiono, and H. Liu, "Neurorule: A connectionist approach to data mining," *arXiv preprint arXiv:1701.01358*, 2017.

[65] M. F. Sanner, "Python: a programming language for software integration and development," *J Mol Graph Model*, vol. 17, pp. 57-61, 1999.

APPENDIX-A
SAMPLE OF THE INPUT DATASET

dstip	dsport	proto	state	dur	sbytes	dbytes	sttl	dttl	sloss	dloss	service	Sload	Dload
149.171.126.6	53	udp	CON	0.001055	132	164	31	29	0	0	dns	500473.938	621800.938
149.171.126.9	1024	udp	CON	0.036133	528	304	31	29	0	0	-	87676.0859	50480.1719
149.171.126.7	53	udp	CON	0.001119	146	178	31	29	0	0	dns	521894.531	636282.375
149.171.126.5	53	udp	CON	0.001209	132	164	31	29	0	0	dns	436724.563	542597.188
149.171.126.0	53	udp	CON	0.001169	146	178	31	29	0	0	dns	499572.25	609067.563
149.171.126.9	111	udp	CON	0.078339	568	312	31	29	0	0	-	43503.2344	23896.1426
149.171.126.4	53	udp	CON	0.001134	132	164	31	29	0	0	dns	465608.469	578483.25
10.40.182.3	0	arp	INT	0	46	0	0	0	0	0	-	0	0
149.171.126.6	53	udp	CON	0.001126	146	178	31	29	0	0	dns	518650.094	632326.813
149.171.126.4	53	udp	CON	0.001167	132	164	31	29	0	0	dns	452442.156	562125.063
10.40.170.2	0	arp	INT	0	46	0	0	0	0	0	-	0	0
10.40.170.2	0	arp	INT	0	46	0	0	0	0	0	-	0	0
10.40.182.3	0	arp	INT	0	46	0	0	0	0	0	-	0	0
149.171.126.5	53	udp	CON	0.001093	132	164	31	29	0	0	dns	483074.094	600182.938
149.171.126.9	41049	udp	CON	0.001851	528	304	31	29	0	0	-	1711507.25	985413.313
149.171.126.6	44307	udp	CON	0.001749	528	304	31	29	0	0	-	1811320.75	1042881.63
149.171.126.4	53	udp	CON	0.001128	132	164	31	29	0	0	dns	468085.125	581560.313
149.171.126.9	111	udp	CON	0.005153	568	312	31	29	0	0	-	661362.313	363283.531
149.171.126.6	111	udp	CON	0.004898	568	312	31	29	0	0	-	695794.188	382196.813
149.171.126.5	53	udp	CON	0.001111	132	164	31	29	0	0	dns	475247.531	590459.063
149.171.126.18	32780	udp	INT	0.000021	728	0	254	0	0	0	-	138666672	0
149.171.126.16	80	tcp	FIN	0.240139	918	25552	62	252	2	10	http	28050.4219	815794.188
149.171.126.16	80	tcp	FIN	2.39039	1362	268	254	252	6	1	http	4233.61914	749.668518
149.171.126.9	53	udp	CON	0.001101	132	164	31	29	0	0	dns	479564.031	595822
149.171.126.8	53	udp	CON	0.001082	132	164	31	29	0	0	dns	487985.219	606284.688
149.171.126.2	53	udp	CON	0.001122	132	164	31	29	0	0	dns	470588.219	584670.188
149.171.126.6	53	udp	CON	0.001141	146	178	31	29	0	0	dns	511831.75	624014.063
149.171.126.1	53	udp	CON	0.001164	146	178	31	29	0	0	dns	501718.219	611683.813
149.171.126.9	53	udp	CON	0.001127	132	164	31	29	0	0	dns	468500.469	582076.313
149.171.126.4	53	udp	CON	0.001073	132	164	31	29	0	0	dns	492078.281	611370
149.171.126.9	53	udp	CON	0.001196	146	178	31	29	0	0	dns	488294.313	595317.688
149.171.126.1	53	udp	CON	0.001101	132	164	31	29	0	0	dns	479564.031	595822
149.171.126.6	53	udp	CON	0.001058	146	178	31	29	0	0	dns	551984.875	672967.875
149.171.126.8	53	udp	CON	0.0011	132	164	31	29	0	0	dns	480000	596363.625
149.171.126.0	53	udp	CON	0.001126	146	178	31	29	0	0	dns	518650.094	632326.813
149.171.126.8	53	udp	CON	0.001017	146	178	31	29	0	0	dns	574238	700098.375
149.171.126.7	53	udp	CON	0.001094	132	164	31	29	0	0	dns	482632.563	599634.375
149.171.126.8	53	udp	CON	0.001092	132	164	31	29	0	0	dns	483516.469	600732.563
149.171.126.1	53	udp	CON	0.001082	132	164	31	29	0	0	dns	487985.219	606284.688
149.171.126.16	5555	tcp	FIN	0.17519	8168	268	254	252	4	1	-	346366.813	10228.8945
149.171.126.10	80	tcp	FIN	0.1906	844	268	254	252	2	1	http	31899.2676	9401.88867
149.171.126.4	53	udp	CON	0.001089	132	164	31	29	0	0	dns	484848.469	602387.5
149.171.126.1	53	udp	CON	0.001126	132	164	31	29	0	0	dns	468916.531	582593.25
149.171.126.1	23202	udp	CON	0.001859	528	304	31	29	0	0	-	1704142	981172.688
149.171.126.9	53	udp	CON	0.001152	132	164	31	29	0	0	dns	458333.313	569444.438
149.171.126.1	111	udp	CON	0.004996	568	312	31	29	0	0	-	682145.75	374699.781
149.171.126.5	53	udp	CON	0.001044	146	178	31	29	0	0	dns	559386.938	681992.313
149.171.126.3	53	udp	CON	0.001089	132	164	31	29	0	0	dns	484848.469	602387.5
149.171.126.6	53	udp	CON	0.001122	146	178	31	29	0	0	dns	520499.094	634581.063
149.171.126.2	53	udp	CON	0.001201	130	162	31	29	0	0	dns	432972.531	539550.375
149.171.126.4	53	udp	CON	0.001079	132	164	31	29	0	0	dns	489342	607970.375
149.171.126.3	38340	udp	CON	0.001864	528	304	31	29	0	0	-	1699570.88	978540.75
149.171.126.3	111	udp	CON	0.0051	568	312	31	29	0	0	-	668235.25	367058.813
149.171.126.8	53	udp	CON	0.001123	132	164	31	29	0	0	dns	470169.156	584149.563
149.171.126.7	53	udp	CON	0.001112	132	164	31	29	0	0	dns	474820.156	589928.063
149.171.126.5	53	udp	CON	0.001066	132	164	31	29	0	0	dns	495309.594	615384.625
149.171.126.5	53	udp	CON	0.001108	130	162	31	29	0	0	dns	469314.063	584837.5
149.171.126.15	80	tcp	FIN	0.177449	1214	268	254	252	2	1	http	49276.1289	10098.6758
149.171.126.14	3000	tcp	FIN	0.19491	844	268	254	252	2	1	-	31193.8828	9193.98633
149.171.126.8	55173	udp	CON	0.011205	528	304	31	29	0	0	-	282730.938	162784.469
149.171.126.2	53	udp	CON	0.001091	130	162	31	29	0	0	dns	476626.938	593950.5
149.171.126.5	53	udp	CON	0.001091	130	162	31	29	0	0	dns	476626.938	593950.5
149.171.126.7	53	udp	CON	0.001135	132	164	31	29	0	0	dns	465198.219	577973.563
149.171.126.1	53	udp	CON	0.001105	132	164	31	29	0	0	dns	477828.031	593665.125
149.171.126.1	53	udp	CON	0.001087	146	178	31	29	0	0	dns	537258.5	655013.813
149.171.126.8	111	udp	CON	0.040337	568	312	31	29	0	0	-	84488.1875	46409.0039
149.171.126.2	32859	udp	CON	0.0314	520	304	31	29	0	0	-	99363.0625	58089.1758
149.171.126.2	111	udp	CON	0.060192	568	304	31	29	0	0	-	56618.8203	30303.0293

Spkts	Dpkts	swin	dwin	stcpb	dtcpb	smeansz	dmeansz	trans_depth	res_bdy_len	Sjit	Djit
2	2	0	0	0	0	66	82	0	0	0	0
4	4	0	0	0	0	132	76	0	0	9.89101	10.682733
2	2	0	0	0	0	73	89	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	73	89	0	0	0	0
4	4	0	0	0	0	142	78	0	0	29.682221	34.37034
2	2	0	0	0	0	66	82	0	0	0	0
1	0	0	0	0	0	46	0	0	0	0	0
2	2	0	0	0	0	73	89	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
1	0	0	0	0	0	46	0	0	0	0	0
1	0	0	0	0	0	46	0	0	0	0	0
1	0	0	0	0	0	46	0	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
4	4	0	0	0	0	132	76	0	0	0.656903	0.328339
4	4	0	0	0	0	132	76	0	0	0.640403	0.280968
2	2	0	0	0	0	66	82	0	0	0	0
4	4	0	0	0	0	142	78	0	0	1.890104	1.610554
4	4	0	0	0	0	142	78	0	0	1.780739	1.549507
2	2	0	0	0	0	66	82	0	0	0	0
2	0	0	0	0	0	364	0	0	0	0	0
12	24	255	255	1708297952	1939490744	77	1065	1	12026	1170.48167	1144.38336
14	6	255	255	3897219059	2466816006	97	45	1	0	18786.7114	941.724938
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	73	89	0	0	0	0
2	2	0	0	0	0	73	89	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	73	89	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	73	89	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	73	89	0	0	0	0
2	2	0	0	0	0	73	89	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
14	6	255	255	2505143795	3592239707	583	45	0	0	774.788316	47.765387
10	6	255	255	3006332195	1452987536	84	45	1	0	996.632407	59.532129
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	4	0	0	0	0	132	76	0	0	0.662323	0.337295
2	2	0	0	0	0	66	82	0	0	0	0
4	4	0	0	0	0	142	78	0	0	1.838719	1.588406
2	2	0	0	0	0	73	89	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	73	89	0	0	0	0
2	2	0	0	0	0	65	81	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
4	4	0	0	0	0	132	76	0	0	0.661617	0.338476
4	4	0	0	0	0	142	78	0	0	1.869125	1.620926
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	65	81	0	0	0	0
10	6	255	255	366102997	2277661750	121	45	1	0	1020.23676	62.002961
10	6	255	255	257622786	1677629526	84	45	0	0	1071.49701	61.076766
4	4	0	0	0	0	132	76	0	0	0.625554	4.74233
2	2	0	0	0	0	65	81	0	0	0	0
2	2	0	0	0	0	65	81	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	66	82	0	0	0	0
2	2	0	0	0	0	73	89	0	0	0	0
4	4	0	0	0	0	142	78	0	0	11.786999	18.234869
4	4	0	0	0	0	130	76	0	0	7.678237	10.358879
4	4	0	0	0	0	142	76	0	0	21.212496	27.670975

Stime	Ltime	Sintpkt	Dintpkt	tcprtt	synack	ackdat	is_sm_ips_ports	ct_state_ttl	ct_flw_http_mthd
1421927414	1421927414	0.017	0.013	0	0	0	0	0	0
1421927414	1421927414	7.005	7.564333	0	0	0	0	0	0
1421927414	1421927414	0.017	0.013	0	0	0	0	0	0
1421927414	1421927414	0.043	0.014	0	0	0	0	0	0
1421927414	1421927414	0.005	0.003	0	0	0	0	0	0
1421927414	1421927414	21.003	24.315	0	0	0	0	0	0
1421927414	1421927414	0.017	0.013	0	0	0	0	0	0
1421927415	1421927415	0	0	0	0	0	1	2	0
1421927415	1421927415	0.018	0.013	0	0	0	0	0	0
1421927415	1421927415	0.018	0.013	0	0	0	0	0	0
1421927415	1421927415	0	0	0	0	0	1	2	0
1421927415	1421927415	0	0	0	0	0	1	2	0
1421927415	1421927415	0	0	0	0	0	1	2	0
1421927415	1421927415	0.018	0.013	0	0	0	0	0	0
1421927415	1421927415	0.471	0.237667	0	0	0	0	0	0
1421927415	1421927415	0.458333	0.203667	0	0	0	0	0	0
1421927415	1421927415	0.017	0.015	0	0	0	0	0	0
1421927415	1421927415	1.348	1.149333	0	0	0	0	0	0
1421927415	1421927415	1.269667	1.103667	0	0	0	0	0	0
1421927415	1421927415	0.018	0.013	0	0	0	0	0	0
1421927415	1421927415	0.021	0	0	0	0	0	2	0
1421927416	1421927416	21.830818	9.570304	0.051475	0.006528	0.044947	0	1	1
1421927414	1421927416	183.579303	474.259406	0.066088	0.017959	0.048129	0	1	1
1421927416	1421927416	0.017	0.012	0	0	0	0	0	0
1421927416	1421927416	0.011	0.009	0	0	0	0	0	0
1421927416	1421927416	0.018	0.014	0	0	0	0	0	0
1421927416	1421927416	0.017	0.015	0	0	0	0	0	0
1421927416	1421927416	0.018	0.013	0	0	0	0	0	0
1421927416	1421927416	0.017	0.015	0	0	0	0	0	0
1421927416	1421927416	0.011	0.008	0	0	0	0	0	0
1421927416	1421927416	0.017	0.015	0	0	0	0	0	0
1421927416	1421927416	0.017	0.013	0	0	0	0	0	0
1421927417	1421927417	0.017	0.013	0	0	0	0	0	0
1421927417	1421927417	0.018	0.013	0	0	0	0	0	0
1421927417	1421927417	0.011	0.008	0	0	0	0	0	0
1421927417	1421927417	0.011	0.012	0	0	0	0	0	0
1421927417	1421927417	0.011	0.009	0	0	0	0	0	0
1421927417	1421927417	0.017	0.012	0	0	0	0	0	0
1421927417	1421927417	0.017	0.012	0	0	0	0	0	0
1421927417	1421927417	11.837692	33.287	0.054878	0.008744	0.046134	0	1	0
1421927418	1421927418	18.573778	36.845602	0.050675	0.006354	0.044321	0	1	1
1421927418	1421927418	0.011	0.009	0	0	0	0	0	0
1421927418	1421927418	0.017	0.012	0	0	0	0	0	0
1421927418	1421927418	0.475333	0.244	0	0	0	0	0	0
1421927418	1421927418	0.011	0.009	0	0	0	0	0	0
1421927418	1421927418	1.312667	1.137667	0	0	0	0	0	0
1421927418	1421927418	0.018	0.011	0	0	0	0	0	0
1421927418	1421927418	0.018	0.013	0	0	0	0	0	0
1421927418	1421927418	0.017	0.012	0	0	0	0	0	0
1421927418	1421927418	0.019	0.016	0	0	0	0	0	0
1421927419	1421927419	0.011	0.008	0	0	0	0	0	0
1421927419	1421927419	0.474333	0.244333	0	0	0	0	0	0
1421927419	1421927419	1.332667	1.156667	0	0	0	0	0	0
1421927419	1421927419	0.012	0.008	0	0	0	0	0	0
1421927419	1421927419	0.018	0.016	0	0	0	0	0	0
1421927419	1421927419	0.011	0.008	0	0	0	0	0	0
1421927419	1421927419	0.017	0.012	0	0	0	0	0	0
1421927419	1421927419	19.203667	34.071199	0.05198	0.007076	0.044904	0	1	1
1421927420	1421927420	19.932667	37.800801	0.050128	0.005888	0.04424	0	1	0
1421927420	1421927420	0.449333	3.361333	0	0	0	0	0	0
1421927420	1421927420	0.011	0.008	0	0	0	0	0	0
1421927420	1421927420	0.011	0.008	0	0	0	0	0	0
1421927420	1421927420	0.011	0.008	0	0	0	0	0	0
1421927420	1421927420	0.011	0.008	0	0	0	0	0	0
1421927420	1421927420	0.011	0.008	0	0	0	0	0	0
1421927420	1421927420	8.349667	12.905	0	0	0	0	0	0
1421927420	1421927420	5.437333	7.331333	0	0	0	0	0	0
1421927420	1421927420	15.011	19.576334	0	0	0	0	0	0

is_ftp_login	ct_ftp_cmd	ct_srv_src	ct_srv_dst	ct_dst_ltm	ct_src_ ltm	ct_src_dport_ltm	ct_dst_sport_ltm	ct_dst_src_ltm	attack_cat	Label
0	0	3	7	1	3	1	1	1		0
0	0	2	4	2	3	1	1	2		0
0	0	12	8	1	2	2	1	1		0
0	0	6	9	1	1	1	1	1		0
0	0	7	9	1	1	1	1	1		0
0	0	2	4	2	3	1	1	2		0
0	0	12	7	1	2	2	1	1		0
0	0	2	2	2	2	2	2	2		0
0	0	6	7	3	1	1	1	1		0
0	0	6	7	2	1	1	1	1		0
0	0	2	2	2	2	2	2	2		0
0	0	2	2	2	2	2	2	2		0
0	0	2	2	2	2	2	2	2		0
0	0	6	9	2	5	1	1	1		0
0	0	8	4	2	5	1	1	2		0
0	0	8	2	3	5	1	1	2		0
0	0	7	7	2	1	1	1	1		0
0	0	8	4	2	5	2	1	2		0
0	0	8	2	3	5	2	1	2		0
0	0	5	9	2	1	1	1	1		0
0	0	1	1	1	1	1	1	1	Exploits	1
0	0	3	2	2	1	1	1	1	Exploits	1
0	0	5	2	2	1	1	1	1	Reconnaissance	1
0	0	7	4	3	1	1	1	1		0
0	0	7	6	1	1	1	1	1		0
0	0	8	3	1	2	2	1	1		0
0	0	5	7	1	1	1	1	1		0
0	0	3	8	2	1	1	1	1		0
0	0	6	4	3	1	1	1	1		0
0	0	6	7	1	1	1	1	1		0
0	0	8	4	3	2	2	1	1		0
0	0	6	8	2	1	1	1	1		0
0	0	6	7	1	2	2	1	1		0
0	0	6	6	3	2	2	1	1		0
0	0	8	9	1	1	1	1	1		0
0	0	5	6	3	2	2	1	1		0
0	0	5	8	1	2	2	1	1		0
0	0	3	6	3	1	1	1	1		0
0	0	7	8	1	1	1	1	1		0
0	0	1	1	1	1	1	1	1	Exploits	1
0	0	3	1	1	1	1	1	1	Exploits	1
0	0	7	7	1	4	2	1	1		0
0	0	8	8	3	1	1	1	1		0
0	0	5	3	3	4	1	1	2		0
0	0	6	4	1	1	1	1	1		0
0	0	5	3	3	4	1	1	2		0
0	0	7	9	1	4	2	1	1		0
0	0	7	4	1	1	1	1	1		0
0	0	5	7	1	1	1	1	1		0
0	0	12	3	1	1	1	1	1		0
0	0	12	7	1	2	2	1	1		0
0	0	8	2	2	3	1	1	2		0
0	0	8	2	2	3	1	1	2		0
0	0	7	6	1	1	1	1	1		0
0	0	6	8	1	1	1	1	1		0
0	0	6	9	2	3	1	1	1		0
0	0	12	9	2	2	2	1	1		0
0	0	5	2	1	1	1	1	1	DoS	1
0	0	1	1	1	1	1	1	1	Generic	1
0	0	8	2	2	2	1	1	2		0
0	0	8	3	3	2	2	1	1		0
0	0	5	9	2	1	1	1	1		0
0	0	7	8	1	1	1	1	1		0
0	0	7	8	2	1	1	1	1		0
0	0	8	8	2	2	2	1	1		0
0	0	8	2	2	2	1	1	2		0
0	0	2	2	3	3	1	1	2		0
0	0	2	2	3	3	1	1	2		0

APPENDIX-B
DEEP LEARNING PYTHON CODE

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn import metrics
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import OneHotEncoder
import numpy as np
import time
TD=np.load('ED_Crop_Intif_MV_Norm.npy')
TL=np.load('EDLabels.npy')
OHE=OneHotEncoder(sparse=False).fit(TL.reshape(-1,1))
kfolds=StratifiedKFold(n_splits=5,shuffle=True)
c=1
epochs=100
BMA=[]
BPRD=np.empty((0,1))
BACT=np.empty((0,1))
BPRT=[]
MCMA=[]
MCPRD=np.empty((0,10))
MCACT=np.empty((0,10))
MCPRT=[]
OD=np.empty((0,43))
OL=np.empty((0,1))
for train,test in kfolds.split(TD,TL):
    print('Step: ',c)
    c=c+1
    TrD=TD[train]
    TrL=TL[train]
    TrBL=(TrL==6).astype(int)
    TsD=TD[test]
    TsL=TL[test]
```

```
TsBL=(TsL==6).astype(int)
BACT=np.vstack((BACT,TsBL.reshape(-1,1)))
OD=np.vstack((OD,TrD))
Bmodel = Sequential()
Bmodel.add(Dense(128,input_dim=43,activation='relu',use_bias=True))
Bmodel.add(Dropout(0.5))
Bmodel.add(Dense(64,activation='relu',use_bias=True))
Bmodel.add(Dropout(0.2))
Bmodel.add(Dense(32,activation='relu',use_bias=True))
Bmodel.add(Dropout(0.2))
Bmodel.add(Dense(1,activation='sigmoid',use_bias=True))
Bmodel.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
Bmodel.fit(TrD,TrBL,epochs=2*epochs, batch_size=100000,verbose=2)
st=time.time()
TsY = Bmodel.predict(TsD, batch_size=100000)
pt=time.time()-st
BPRT.append(pt)
Bpred = (TsY>0.5).astype(int)
OL=np.vstack((OL,TsL.reshape(-1,1)[Bpred==0].reshape(-1,1)))
BMA.append(metrics.accuracy_score(TsBL,Bpred))
BPRD=np.vstack((BPRD,Bpred))
MCTrD=TrD[TrL!=6]
MCTrL=OHE.transform(TrL[TrL!=6].reshape(-1,1))
MCTsD=TsD[(Bpred==0).reshape(-1),:]
MCTsL=OHE.transform(TsL[(Bpred==0).reshape(-1)].reshape(-1,1))
MCACT=np.vstack((MCACT,MCTsL))
MCmodel = Sequential()
MCmodel.add(Dense(128,input_dim=43,activation='relu',use_bias=True))
Bmodel.add(Dropout(0.2))
MCmodel.add(Dense(64,activation='relu',use_bias=True))
Bmodel.add(Dropout(0.2))
MCmodel.add(Dense(32,activation='relu',use_bias=True))
Bmodel.add(Dropout(0.2))
MCmodel.add(Dense(10,activation='softmax',use_bias=True))
```

```
    MCmodel.compile(loss='categorical_crossentropy',optimizer='adam',
metrics=['categorical_accuracy'])
    MCmodel.fit(MCTrD,MCTrL,epochs=5*epochs,batch_size=100000,verbose=2)
    st=time.time()
    MCY=MCmodel.predict(MCTsD,batch_size=100000)
    pred=(MCY>0.5).astype(int)
    pt=time.time()-st
    MCPRT.append(pt)
    MCMA.append(metrics.accuracy_score(MCTsL,pred))
    MCPRD=np.vstack((MCPRD,pred))
metrics.accuracy_score(MCACT,MCPRD)
metrics.accuracy_score(BACT,BPRD)
```