Chapter One

Introduction

1.1 Background to the Study

Nowadays, computer network is useful in every aspect of human life and production which takes place daily. The use of desktop computers, palmtop, laptop or some of the other internet-enabled devices which makes for easy access to information that is available on the internet. The network comprises of various companies such as schools and organization who share information among themselves. According to statistics from the National Telecommunications and Information Administration, by November 2017, more than 240 million people in the United States used the Internet, accounting for about 77% of the US population. The line chart from National Telecommunications and Information Administration in Figure 1.1 (***See Appendix 1***) shows that the number of Internet use in the U.S. fluctuated up and down for no more than 10% of the American population from 2010 to 2014, but it is steadily increasing at the whole.

However, the wide spreading and developing of computer network accompanies with the security challenges for people's privacy and assets that are stored or transmitted through the computer network. In the past few years, several critical data breach incidents have been reported, and caused sensations in the world. The latest incident of data breach that also called publics a great attention was the database breach of Capital One on March 2019. Forbes reported that Capital One misconfigured the Amazon server making it a vulnerable target. As a former employee of Amazon, the hacker in this case got the access to the login information of the company's computer and server, and then stole the data. Over 100 million credit card users of

Capital One were influenced. And the hacking incident costed Capital One 100 million to 150 million US dollars[1]. From this case, it is not difficult to find that the network attack could cause extremely critical damage to people and organization's privacy and economic property. Thus, most people and organization have become more concern about the network security issues during the usage of Internet-enabled devices and computers to gain access to information and storing of important data.

Generally, one of the powerful tools used in the protecting of network security is the Intrusion Detection Systems which is able to detect network attacks from orchestrated by internet uses who are suspicious. Examination of the network traffic is done by the systems which scrutinize the packet at the various layers of communication model[2]. In recent years, the use of machine learning by many researchers to improve Intrusion Detection System and proffer solution to issues of network security, inclusive of detection and classification of abnormal activities on the network through the learning of the different attacks by the systems using the sample data and detection inclusive[3]. The consistent advancement of Artificial Intelligent and its usage in different technological fields. Been a component of Artificial Intelligent, machine learning also has huge prospect in the area of cybersecurity.

Overall machine learning process could be able to be understood and upfront, however, the limitation is not restricted to the content of this research. When machine learning approaches are applied by researchers, identification of sample data which is the summary pattern of the data of the sample in the network traffic's packet. For example, port's number, IP address and protocol, etc. Models are used to train features by Researchers and high rate of identification and accuracy is attained, and application of the machine learning that has been trained model to the further detection and categorization of attacks of the network that occur in network traffic. However, in the use of machine learning for research different challenges in the same area, testing and training by the used features may be different. For example, DDoS attacks detection

may need various packet information investigation generated from detecting spam. Hence, ability to choose the correct features and solid domain knowledge acquisition are vital functions in machine learning.

1.2 Statement of the Problem

Machine learning application in intrusion detection on the network is well studied. With over hundred research papers on machine learning Network Intrusion Detection System are published in different conferences every year. The motivating aspect is that many researchers are passionate in trying different approaches of machine learning algorithms in detecting network intrusion. Nevertheless, the cases of withholding of information of why some features are chosen by many researchers or the complete use of all features present in the dataset are numerous except for some few researchers who are studying on feature engineering phrase of machine learning.

After looking at machine learning in deep light, the use of numerous non-representative features could generate difficult problems for developing a precise and active model on machine learning. Feeding huge volume of data that could generate millions of combinations to a model of machine learning can complicate the pattern of attack detection during monitoring process[4]. On the other hand, one of major factor is training and classification time can be hindered by the inclusion of redundant features[5]. However, researchers often abandon the significant of feature selection[6]. An important role of feature selection in predicting or classification purposes or developing machine learning can't be overemphasis. Different kind of attacks on network have distinct attack pattern which can be discovered in the data categorized in different features. Feature selection procedures are verification process to ensure that every feature play the same important role in attack identification on the network in a certain machine learning model and indicate how possible it is to input all data into an

algorithm of machine learning in order to get a modest accuracy and identification. In one of the studies, it was indicated that since the decreased in performance of intrusion detection system is caused by the high dimensionality of data, disturbance of features that are not relevant needs eradication and the need for the best feature subset which would boost the accuracy of detection becomes a crucial job[7].

In relations to the above issue raised, can the efficacy of machine learning models in intrusion detection-based network be enhanced by domain knowledge-based features working with feature selection?

1.3 Aim and Objectives of the Study

The main aim of this study is to improve on Intrusion Detection System using hybridized feature selection methods

The specific objectives are :

i. development of a feature selection/ reduction model
ii. development of value-added Intrusion Detection System
iii. to test and evaluate the developed system.

1.4 Significance of the Study

In many researches about the machine learning in network security, the repeated emphases have been on the feature selection solution. Application of these improved feature selection approaches would result in a remarkable improvement to the operation of Intrusion Detection System in the network, the research indicated that “good feature selection results can improve learning accuracy, reduce learning time, and simplify learning results” [8]. On one hand, using

key features that are interrelated with each other with approach on machine learning could successfully decrease the repetition of the experiment[9]. On the other hand, acquiring the optimum feature subset can result to reduce effectively the scope of features which would be used for machine learning model training, which would at the long run reduce the effect of over-fitting and generate an improved broad view of models[10]. Also, time for processing of data and model training is save by less features which is fed into machine learning model[11]. Eradicating features that are not important using feature selection method boost the accuracy of classification of machine learning model[12].

1.5 Scope of the Study

The feature selection approach will create feature subsets that contain less than 15 features, which is the average size of feature subset in other related researches. The selected subsets of features will help the machine learning model achieve the same or better accuracy of network attack identification than that of using all features in the given dataset. The selected subset of features will help to shorten the training and testing time of the machine learning model that uses all features in the given dataset.

1.6 Limitation of the Study

The attacks and features that are tested in this research are limited to those in the NSL-KDD dataset and the model only performs the tasks of intrusion detection. The prevention function is not studied in this thesis. Only statistical methods of machine learning are considered as evaluation models to examine the results of experiment. No deep learning technique is used in this research because of lack of adequate tools.

1.7 Operational Definition of Terms

a. **True Positive (TP):** Classifying an intrusion as an intrusion. The true positive rate is synonymous with detection rate, sensitivity and recall, which are other terms often used in the literature.

b. **False Positive (FP):** Incorrectly classifying normal data as an intrusion, also known as a false alarm. False positives measure the false alarm rate.

c. **True Negative (TN):** Correctly classifying normal data as normal. The true negative rate is also referred to as specificity.

d. **False Negative (FN):** Incorrectly classifying an intrusion as normal. False negatives measure the detection rate.

e. **Dataset:** A data set is a collection of related, discrete items of related data that may be accessed individually or in combination or managed as a whole entity. A data set is organized into some type of data structure.

f. **Brute Force Attack:** An attempt to break a security system by trying every possible combination of passwords or encryption keys.

g. **Authentication:** Providing a verification system to determine who actually wrote a message. Common systems use a dual-key encryption system.

h. **Denial of Service (DoS):** An incident in which a user or organization is deprived of the services of a resource they would normally expect to have. Typically, the loss of service is the inability of a particular network service, such as e-mail to be available or the temporary loss of all network connectivity and services.

i. **Distributed Denial of Service (DDoS):** An incident in which a multitude of compromised systems attack a single target, thereby causing denial of service for users of the

targeted system. The flood of incoming messages to the target system essentially forces it to shut down, thereby denying service to the system to legitimate users.

j. **IDS (Intrusion Detection System):** An Intrusion Detection System (IDS) is a device or software application that monitors a network for malicious activity or policy violations.

1.8 Outline of the Thesis

Chapter One introduced the background information of machine learning of intrusion detection on the network and feature selection. This chapter also established the statement of problem that showed feature selection is not given the attention it requires in the current study about the machine learning in network intrusion detection, as well as the significance of the problems, scope, and limitations of this project.

Chapter Two comprised of the review of the literature including the conceptual review, the conceptual framework, the review of the related works and summary of the gaps identified from the literature.

Chapter Three outlined the methodology adopted in this thesis including the system design, the data and dataset source, data analysis method and the algorithm used.

Chapter Four described the implementation of the design made in chapter three and the various results obtained and finding made are discussed.

Chapter Five discussed the summary of the findings, contributions to knowledge are outlined and recommendations for future works are made.

Endnotes

1. A. Heidari, J. Jabraeil & A. Mohammad, *Internet of Things Intrusion Detection Systems: A Comprehensive Review and Future Directions*, **Cluster Computing**, 2022, doi: 10.1007/s10586-022-03776-z.

2. A. A. Mughal, *Well-Architected Wireless Network Security*, **Journal of Humanities and Applied Science Research,** vol. 5, no. 1, 2022, doi: https://journals.sagescience.org/index.php/JHASR/article/view/52.

3. J. David & C. Thomas, *Efficient DDoS flood attack detection using dynamic thresholding on flow-based network traffic*, **Computers & Security**, vol. 82, no. 284, 2019, pp. 0167-4048, doi: https://doi.org/10.1016/j.cose.2019.01.002.

4. M. Awad, S. Fraihat, K. Salameh & A .Al Redhaei, *Examining the Suitability of NetFlow Features in Detecting IoT Network Intrusions*, **Sensors**, vol. 22, no. 16, 2022, https://doi.org/10.3390/s22166164.

5. M. Mayuranathan, S. K. Saravanan, B. Muthusenthil & A. Samydurai, *An Efficient Optimal Security System for Intrusion Detection in Cloud Computing Environment Using Hybrid Deep Learning Technique*, **Advances in Engineering Software**, vol. 173, no. 103236, 2022, pp. 0965-9978, doi: https://doi.org/10.1016/j.advengsoft.2022.103236.

6. S. Lata & D. Singh, *Intrusion Detection System in Cloud Environment: Literature Survey & Future Research Directions,* **International Journal of Information Management Data Insights**, vol. 2, no. 2, 2022, pp. 2667-0968, doi: https://doi.org/10.1016/j.jjimei.2022.100134.

7. S. El Kafhali & I. El Mir, *Security Threats, Defense Mechanisms, Challenges, and Future Directions in Cloud Computing*, **Archives of Computational Methods in Engineering**, vol. 29, no. 223, 2022, doi: 10.1007/s11831-021-09573-y.

8. D. E. Denning, *An Intrusion-detection Model*, **IEEE Transactions on Software Engineering**, No. 2, 1987, pp. 222–232.

9. S. B. Bhoopesh, Dikshita, S. B. Nitesh, & C. Garvit, *A Comprehensive Study of Intrusion Detection and Prevention Systems* , **Wireless Communication Security**, vol. 7, 2022, pp. 115-142, doi: https://doi.org/10.1002/9781119777465.ch7.

10. Ö. Murat, K. Cihan, & Z. Ahmet, *Distributed Intrusion Detection Systems: A Survey*, **The Academic Perspective Procedia**, vol. 2, no. 2, 2019, pp. 400 – 407, https://doi.org/10.33793/acperpro.02.03.18.

11. J. Hochberg, K. Jackson, C. Stallings, J. McClary, D. DuBois, & J. Ford, *Nadir: An Automated System for Detecting Network Intrusion and Misuse*, **Computers & Security**, vol. 12, no. 3, 1993, pp. 235–248.

12. V. Paxson, *Bro: A System for Detecting Network Intruders in Real-Time*, **Computer Networks**, vol. 31, no. 23, 1999, pp. 2435–2463.

Chapter Two

Literature Review

2.1 Intrusion Detection Systems

In this chapter, the literature on current state of IDSs would be examined and investigated in details would be presented. Firstly, basic attributes that should be available in an IDSs in general and the information about the system would be discussed. Next would be the categorization of the IDSs based on the methods used in network traffic monitoring, record data flow, identify attacks and warning reports. Methodologies, approaches and all IDSs technologies within the area would be studied in details. The advantages and disadvantages of IDSs technology would be examined and a concise research on the study of each area would be presented. Then, dataset that are popular used in the testing and evaluation stage of the established Intrusion Detection System are scrutinized and concise information are presented about these datasets. Finally, common tools of intrusion detection utilized by organization, individual and institutions to identified threats are stated. Advantages and disadvantages, methods used by intrusion detection by each tools of intrusion detection are examined.

The system that works on observation of events that is taking place on the network/computer system and scrutinizing them for notification of potential events such as attacks and policies use violation or standard security practices is known as intrusion detection[1]. The main focus of IDS is the identification of events that are dangerous, keeping records of information about such events and alerting administrators[2]. In addition to these, detection of issues on policies on security are some of the focus of IDSs usage, alerting on current attacks and even making it hard for people to attacking the system. Generally, for a well-organized and successfully IDS where component of the system must be protected properly. IDS comprises of different parts such as sensors, management servers, users of the system, etc. Ensuring that IDSs components are secured is very

important due to the fact that they are easily targeted by unwanted third parties (attackers), who want to prevent the IDS preventing them from gaining access to vital documents, know the vulnerabilities of the system and detecting attacks[1]. There is therefore, the need for constant updates on all Operating System and application software of all the various parts of the system, and the protection from threat of all software-based IDS components should be put in place[3]. The option of using multiple IDS technologies which is to give a more robust and high efficient detection of attacks yet accurate in result can also be implemented. Various IDS are available for use which include host-based, network-based and wireless-based. Each of these technology gives basically different prevention of attacks and detection and recording of information gathered[4]. Moreover, each of the technology can detect some particular event more efficiently or accurate detecting threats or attacks more than others as their area of strength. The use of network-based and host-based IDSs could be used to offer a coherent clarification using that as an example. Simply put, when selecting IDS technologies, the need to consider the various features and the areas of strength is important. The most generally available tools, procedures and tactics of IDS inside the literature are listed in Figure 2.1

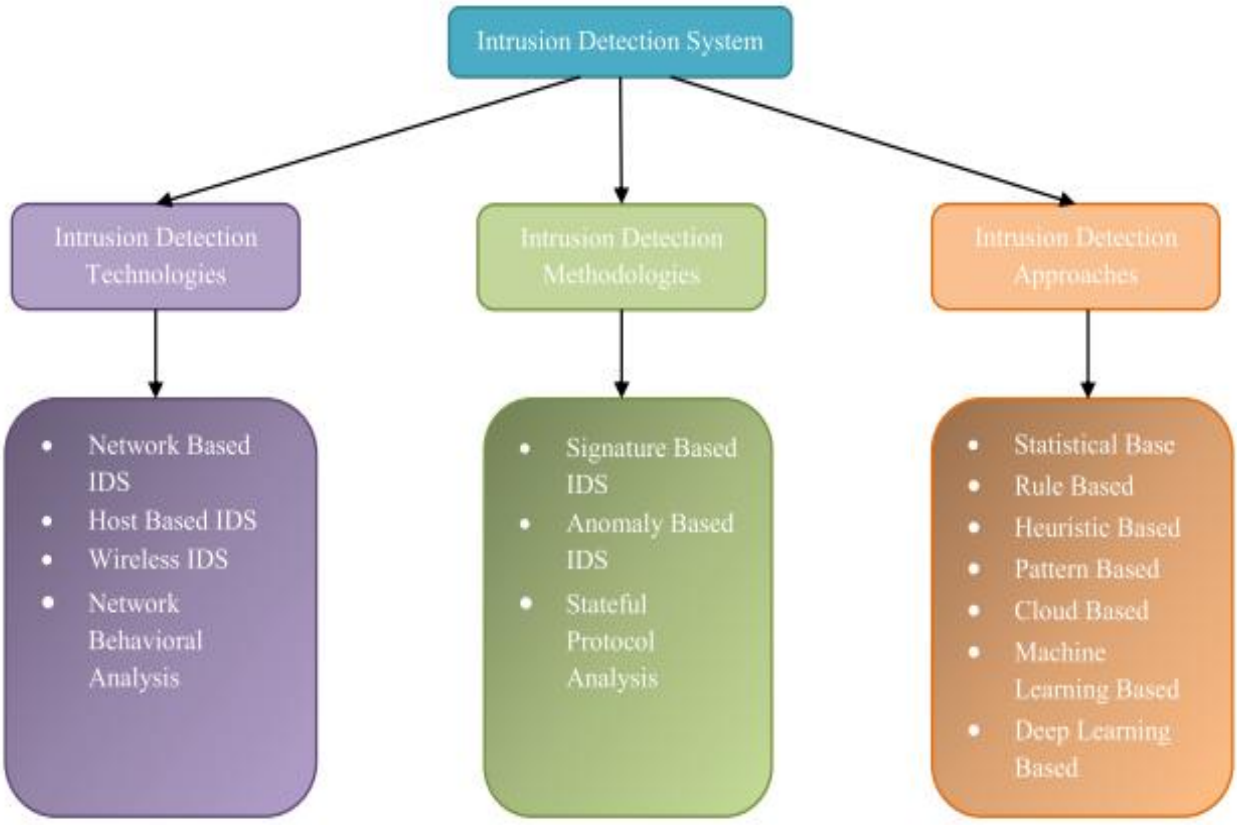


Figure 2.1 Classification of Intrusion Detection System

Source[5]

In conclusion, a necessary system has IDSs become towards securing almost every person, schools and offices because of the high rate of dependency on technology and information system, attacks spreading over the network and likely effect of their damages to files.

2.2 Principles of IDS

The procedure of noting events taking place in the computer system or network and scrutinizing such events so as to diagnose intrusion is known as Intrusion detection [6]. Escalation of privileges, unauthorized access, probe attacks, DoS or DDoS are inclusive in some of the many attacks. Although, some incidents have harmful traits on the system actually are detrimental to the system,

but not in all cases; for example, mistyping of computer address by the user or the wrongful connection of computer unknowingly. There is a need for the system to correctly distinguish attacks from the ordinary traffic on the network.

In summary, a program that shortens and computerizes the procedure of attacks identification. Some vital aspects are available for threat resolving which are effective when implementing IDS technologies:

i. Durability or reliability of the System
ii. The speed or quick detection
iii. Reduced false positives
iv. Maximum rate of detection
v. Minimum usage of hardware and software
vi. Ability of detecting accurate intrusion location
vii. Compatibility of operation with other technology

Concluding, in order to have high accuracy and timely threat detection, an IDS must possess the above-mentioned points.

2.3 Basic Functions of IDS

First and foremost, countless various IDS tools are available based on the type of attacks they are able to detect and the way by which they go about detecting the attacks. Furthermore, is the observation capacity and scrutinize events to be able to detect events that are not wanted, all the various types of IDS must be able to produce the following services[5].

i. Recording Information

In order to compare or generate profile that are normally set, information is saved locally. Furthermore, the information recorded is transferred separately to the information security solution, management systems and central recording servers.

ii. Important Events Documentation

There is a need to accurately and speedily recognize an event that took place outside the regularly recorded information and it is taken as normal.

iii. Report of Identified Important Events

Notifications also known as warning, are implemented via different methods such as messages appearing on interfaces of the emails, system etc. The information sent do contain fundamental notification about wary events that occurred. The users of the IDS would have to enter the system to get the comprehensive information.

iv. Generating Reports

The reports created by the system contains summarized events observed or detailed information provided about the exceptional occurrence. For example, in a situation where wary activities are explained in the part of the report, Intrusion Detection System could gather more comprehensive evidence. Furthermore, settings could be changed or adjusted to indicate the time that warning could be activated after the detecting of threat.

Fundamental characteristics that are common to IDS types are not adequate to provide a comprehensive and precise identification. A false positive is a situation the IDS indicate that a normal activity is a threat[7]. A false negative occurs when it fails to detect or discover a malicious event and declares it as normal. It is quite impossible to completely remove the incidents of negatives and false positives. Actually, in many situations, decreasing a parameter could cause the other conditions to accelerate. A great number of IDS programmers would rather decrease the false negative where it results to the increase the rate of the incidents of false positive.

2.4 Evaluation Metrics of IDS

In order to ascertain established IDS models and check the difference in their operation, metrics such as accuracy, precision, false positive, false negatives and recall are generally used. These values are computed by the use of confusion matrix Table 2.1.

		Predicted Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Table 2.1: Confusion Matrix

Source[65]

$$Recall = TP/(TP+FN) \quad \text{Equation (2.1)}$$

$$Precision = TP/(TP+FP) \quad \text{Equation (2.2)}$$

$$F-Measure = (2*Precision*Recall)/(Precision+Recall) \quad \text{Equation (2.3)}$$

$$Accuracy = TP+TN/(TP+TN+FP+FN) \quad \text{Equation (2.4)}$$

True Positive (TP) is an accurate prognosis of the positive (+) class (prediction and actual are both positive). False (-) Negative (FN) is a false prognosis of the positive group (predicted-negative, real-positive). False (-) Positive (FP) is the false prognosis of the negative group (predicted-positive, real-negative). True Negative (TN) is the accurate prognosis of the negative group (actual and prediction are both negative). Precision (also called positive predictive value) is the proportion of related examples among the taken examples; Recall (also known as sensitivity)

is the proportion of relevant trials engaged. Accuracy is the degree that indicate the quantity of the data was correctly classified. F-measure is the consonant measure of recall and precision[122].

2.5 Challenge of IDS

Intrusion Detection System can be stated as a system that is secured which serves as a detector of traffic in the network and computer systems and make use of these information to detect attacks coming from outside, attacks that are orchestrated from within and system abuses[8]. Today, Intrusion Detection Systems are considered as an important elementary security product among other measures that organization can deploy in business systems. IDSs could also function as an architecture with layers of security protocol when combined with other security products. For illustration purpose, the usage of IDS with anti-virus and firewalls. Thus, IDSs can provide more accurate detection of threat which other security product may not be able to detect. There are various ways and approaches by which IDS identify attacks.

Anomaly detections learning from system calls could be undertaken for a number of years. Nevertheless, in spite of the fact that many researches have been carried out in this scope to generate universal datasets, deficiencies still exist in datasets that could hypothetically be a prototype for all normal behaviour. Simultaneously, anomaly-based approaches can identify both known and unknown attacks to an extent while detecting normal behaviour as threat. It is therefore imperative for end-users and administrators to analyze the result or behaviour flagged by Intrusion Detection System as a threat.

Therefore, there is a reason to bring out the right signature for the program software which was identified via detection of the system through anomaly procedure and examine it if it is an attack or not after analysis.

Signature-based systems, moreover, can identify directly threats based on their pattern but can't identify attacks that are not known. Techniques of Machine Learning have lately been acknowledged widely in the area of intrusion identification.

Various categories of methods have been attested a success in proffering solution to extensive drawbacks such as cyber security, processing of images, pattern recognition, principally in the scope of intrusion identification. Furthermore, Machine Learning techniques are highly productive for approximating between two possible results, such as abnormal or normal, for a specified traffic on the network.

Software Defined Network (SDN) technology is built on a centralized control, demarcating the control or management level from the data level, consequently, giving the capability to plan the network. Network gadgets can be watched and directed from the joint location. Saving and enhancing storage and processing can be done by centralized control of SDN as well. However, SDN doesn't have a standardized security protocol. With the availability of third-party service providers, security concern still exists. In conclusion, coping with the current ever-changing nature of developing attacks types, the IDSs in use cannot be adequate to fight the threat adequately[10].

For the studies to be achieved on these research scope, the inclusion of novel dataset and the implementation of novel technologies would be added which result to a new method that would contribute to the literature would be created. Another issue is the creation of a hybrid IDSs to join the strong point of various IDS categories to protect each other's weakness and the usage of these systems in real world or environment. In this study, there was critical examination and analysis of detailed IDS types were carried out, advantages and disadvantages of the IDS types were also considered and selected to be used so as to create new technologies as a contribution to knowledge.

2.6 Network-Based IDS

A Network-Based IDS (NIDS) observes traffic on the network on behalf of the network devices' security and investigates the protocol (transport, network, application, etc.) which can aid the spotting of dubious events[12]. TCP/IP is extensively used to aid communication on the network. TCP/IP involves four layers which operate jointly. Anytime data is transferred by a user, the data passes through the layers starting from the highest to the lowest and as it passes, each layer adds information to the data. It is the responsibility of the lower layer to pass the data collected over the physical network; transmission of data is done from the layers to the intended end point.

The four TCP/IP levels work jointly to transmit data from one host to another. In network-based IDSs, majority of the scrutiny usually occurs at the application level. Some analysis in some of the network based IDSs are done at the hardware level in limited form.

Network-based IDSs, as a whole, have sensors, multiple consoles, one or more administrative servers and servers for database. Except for the sensors, all other components mentioned have similarities with other IDS architecture as well. Sensors in the network-based IDS observes and examine the events on network.

2.6.1 Security Features of NIDS

Network-based IDSs present comprehensive variety of security capacities. Regular security attributes, can be broadly categorizes into 3 groups, are described below: information collection, logging, and detection[13].

(a) Information Collection

Network-based IDSs have restricted capacity from the communication network getting information from it. The collected information mostly comprises of the details about network activities and related hosts. Attributes of a few gathered information are listed below as:

i. Hosts Identifying: An Intrusion Detection System can generate network hosts list.

ii. Operating System Identification: Identity of the Operating Systems and the version employed on the host can be done. Ability to identify the Operating System type is supportive in detecting the defenseless host.

iii. Applications Identification: The sensor in an IDS has the capacity to recognize application version by scrutinizing the port that is been used and observing communication between the application. Using this method, information gathered helps to recognize application that can be compromised and the types of usages which are not sanctioned.

iv. Determining Network Characterization: Some IDS sensors, traffic and network configuration's information are generally collected. With this collected information, helps to detect any network configuration changes that may occur.

(b) **Logging**

Detected events generate full-scale reports which are recorded by Network-based IDS. Activities such as validation, investigation and correlation of events are done using this data to validate alert. Data types commonly recorded by network based IDSs are as follows:

i. Time and Date

ii. Number of connections

iii. Type of event

iv. Protocols

v. IP addresses destination and Source

vi. Number of packets transmitted

vii. Application requests and responses;

(c) **Detection**

Network-based IDSs provide comprehensive capacity to detect. The combination of anomaly-based method and signature-based method are done by many network-based IDS in order to execute detailed analysis and expand the rate of detection. Whenever the anomalous activities are investigated by anomaly-based methods, it audits it into categories of requests and responses that are scrutinize and differentiated with signature of known attacks. That is, the execution of this method is ingrained.

2.6.2 Related Work on Network-Based IDS

Network-based Intrusion Detection and Prevention System was suggested by the author and his team [14]. This system is designed with the main purpose of identifying known attacks types and to implement prompt actions to counter the attacks. Diverse machine learning techniques and test using a network environment on the internet is the proposed approach for this system. The ability of the proposed IDPS shows result that it can identify normal event from attacks with high precision within seconds and automatically prevent the affected computer's network from attacks. Additionally, C4.5 Decision Tree Algorithm was applied with a suggested proposition to identify types of attacks that are not known. However, a better approach for identifying a known attacks and identifying unknown attacks can further improved this study.

A Network Based Intrusion Detection System was suggested by Amaral for IPv6-enabled wireless sensor networks. The planned system utilizes traffic signatures and irregular behaviour to identify attacks[15]. Proposed system is made up of two components Finger2IPv6 and PPPSniffer. In the planned system, IDS traces the nodes of the network selected as observers. In this way, possible attacks attempts and packets exchanged by neighbours are detected. The rule set created by NIDS are used to compare the observed messages. In the possibility of a match, an alarm is raised and Event Management System receives it. With this planned system, instead of

the detection of predefined attacks, possible misbehaviours can be detected. However, new detection rules should be added as a way of improving the system.

Network Based Intrusion Detection System based on machine learning was suggested and assessed in order to learn how to identify threats to the network[16]. In this study, dataset including labeled examples were generated from various supervised machine learning classifiers, from network traffic features generated by different malicious and benign application.

Android-based malware is the main goal of this study because of the popularity of the usage among phone users and the increase in malware designed to work with phone. To test the proposed traffic, generation of traffic was done. Various malware examples such as backdoor, spammer, ransomware, bots and Premium SMS sender were used to generate the traffic.

From the collected result, the proposed approach had accuracy of 99.4% in detecting unknown and known attacks. By creating an enlarged dataset and incorporating it into the current intrusion detection system mentioned can be used to improve this study.

Anomaly-based Intrusion Detection System (AIDS) can recognize the network traffic that is identified as malicious[17]. Once any activity deviates from the normal behaviour, it generates an alarm each time. Therefore, ability to know true alarms from false positives develops into a key challenge. This study planned a procedure which consist of two steps. Firstly, the most applicable attributes which aids in identifying anomalies in the network which are made up of set of network traffic features was suggested. Secondly, using a proposed AIDS alarm classifier, to group various activities automatically using a packet-header-based anomaly detection system. From the view of the authors, the projected system using the machine learning algorithms is well structured and successful in relations to grouping malicious activities. Diverse machine learning techniques can be used to rise the precision rate resulting to an improved study.

A new hybrid Network-based IDS was offered as an approach to identify irregularity by using AdaBoost and Artificial Bee Colony (ABC) algorithms[18]. The use of ABC algorithm was used to make feature selection. To classify and evaluate the selected features, AdaBoost algorithm was used. NSL-KDD and ISCXIDS2012 datasets was applied to the proposed system to gauge the accuracy of the method. The result shows a 98.9% rate was achieved. From the authors' view, the projected method had an outstanding performances compared to further IDSs on the same dataset. In upcoming studies, performance evaluation could be made on different datasets to improve the accuracy.

The UNSW-NB15 dataset was implemented by employing an anomaly-based network intrusion detection approach[19]. Their approach entails two main stages. Using Random Forests and Recursive Feature Elimination among other techniques, they were able to choose vital features for machine learning uses. A binary classification was then done to identify irregular traffic via diverse data mining techniques such as Logistic Regression, Support Vector Machine and Gradient Boost Machine. Support Vector Machine produced highest accuracy outcome of 82.11%. The output generated by SVM was feed into a set of polynomial classifiers to boost the precision of identifying various types of attacks.

In specific, they assessed the operation of Decision Trees, polynomial SVM, and Naive Bayes. The uses of the two-stage hybrid classification improve the outcome precision up to 86.04%. This research can be further improved by the development of novel categorizing algorithm or using deep learning techniques which can be developed further using diverse datasets. The training of NIDSs using unstable data disposes to give incorrect predictions against small classes of attacks, making misclassified or unidentified attacks slip through the crack. Previous studies have proffer solution to this class imbalance shortcomings by employing data level approaches by expanding minority-class instances or decrease majority class instances. The underlying problem still persist

or failed to be addressed even though these harmonizing methods indirectly improves the operations of NIDS.

Two-layer Improved Siam-IDS (I-SiamIDS) method was suggested to resolve the class inequity's problem based on the research carried out[20]. I-SiamIDS expresses both minority and majority classes as algorithms in the absence of applying any data level harmonizing procedure. The foremost layer of I-SiamIDS applied a binary collaborative of Siamese Neural Network, Deep Neural Network (DNN), Gradient Boosting and eXtreme for sieving of sample of inputs. In Subsequently stages, the second stage receives these attacks to be grouped into various classes of attacks via the multi-class eXtreme Gradient Boosting Classifier (m-XGBoost). In compasrism to alike studies, F1 score, accuracy, precision, recall and AUC values for both CIDDS-001 and NSLKDD dataset were all areas that I SiamIDS showed significant improvement. The computational charge breakdown of the projected method was also displayed, this is to give a clear picture of the result in order to avoid any controversies. At the same time, the examination of different dataset's result could improve this study.

A proposition about the distance between the data sample and its closest neighbor inside the cluster, as well as the distance between the data sample and the associated cluster center, are determined using the CANN technique was slated by the authors[21]. The newly created distance-based feature is then utilized to further categorize the data sample using a k-Nearest Neighbour (k-NN) classifier, which is used for intrusion detection. Higher accuracy and great efficiency in terms of computing are accomplished by the proposed approach.

A methodology was put forth by the authors for analyzing the threat for IoT that relies on Artificial Neural Networks (ANN) to control the threats[22]. Its capacity to detect DoS and DDoS attacks has been examined using multi-level perceptron. This article divides Internet of Things

network activity into lawful and unlawful activities. An Internet of Things network is used to test the Artificial Neural Network module. The results reveal accuracy of 99.4% and it is able to successfully detect DDoS/DoS attacks[23].

A fuzzy association rule-based IDS framework was implemented by the authors. The system is upgraded with a hierarchical and bidirectional fuzzy rule based technique. The suggested framework makes use of fuzzy rules which are subsequently employed for constructing the classifiers and at the same time it provides security alerts. Chandrasekhar and Raghuveer suggested a Least Square Support Vector Machine based IDS (LSSVM-IDS) deployed on the proposed feature selection technique. The efficacy of the IDS is examined on prominent datasets, including KDD Cup 99, NSL-KDD, and Kyoto 2006 . The experimental results reveal that as the proposed feature selection algorithm delivers crucial features for the Intrusion Detection System, the results are more accurate with fewer computing cost as compared to the other current approaches[24].

The authors reported a two-class problem, i.e., normal and anomalous. The author pointed out that although numerous ensemble approaches exist, to locate an appropriate ensemble strategy for a dataset is time-consuming. An ensemble construction approach using PSO generated weight was developed to create an integrated classifier that has greater accuracy. Local unimodal sampling (LUS) has been applied as a meta-optimizer to find more important parameters for Particle Swarm Optimization. In their investigation, they have picked five subsets chosen randomly from the KDD99 dataset. Ensemble classifiers have also been built utilizing multiple approaches and the weighted majority algorithm (WMA) technique. They employed the NSL-KDD data set, and handle the binary and multiclass problem utilizing 20% dataset for testing[25].

A hybrid technique was derived by the authors that can be utilized to assess the intrusion threshold degree which is based on the transaction data's optimal characteristics of the network

extracted from the training data. The results reveal that the hybrid strategy minimizes the computational and time complexity. Additionally, the model was successful in achieving a high degree of accuracy on the binary class and multiclass data sets of NSL-KDD, respectively, of 99.81% and 98.56%[26].

The LGP-BA algorithm was proposed by the authors for feature selection, and the features that were chosen are then classified using SVM. Higher accuracy and more efficiency are gained using the proposed approach[27].

A blend of abuse as well as anomaly detection approaches for intrusion detection. A multi-layered, PSO method for feature selection was proposed[28]. The suggested system is very reliable and efficient. Meta-heuristics based techniques like PSO are able to offer relatively decent outcomes in the given time frame. It is able to handle real-time attacks, also the detection is rapid and less time demanding.

The authors employed approaches including Support Vector Machine, Naive Bayes, and J48, for selection of the features. Since there are various classification strategies that produce more accurate results, the classification techniques employed in the work have little value to the research in IDS.

They have devised an Intrusion Detection System for smart houses. SmartGuard, which employs a cluster-based approach for intrusion detection, can assist the system in detecting infiltration. The nodes' energy is considered as part of the intrusion detection process. The suggested IDS has considerable contribution towards intrusion detection in Smart homes and at the same time the cluster based approach spreads the load of intrusion detection across the nodes Since the technology is evolving day-by-day, such form of IDS is the necessity of the day[29]. A different cluster-based method was offered for spotting intrusions in the Internet of Things. The report mentions two distinct methods for intrusion detection: the Key Match Algorithm and the Cluster Based Algorithm. The Key Mach Algorithm's true positive rate ranges from 50 to 80%, while the

cluster-based algorithm's ranges from 76 to 96%, demonstrating the effectiveness of this approach. Detecting intrusion in IoT based networks is still an issue and the proposed study seeks to address this[30].

A deep convolutional neural network-based in-vehicle IDS was proposed by the presenters. The test vehicle's Controller Area Network (CAN) bus has been subject to network intrusion detection by the system. The in-vehicle network communication standard is called CAN. The proposed system detects harmful behavior on the basis of learning. It has been demonstrated that the system has a high detection rate, a low rate of false negatives, and is less sophisticated. The datasets used in the studies have been created by the researchers. The results have also been compared to other machine-learning techniques[31].

They have suggested another CNN-based intrusion detection solution for the industrial Internet of Things (IIoT). The feature data is separated into four sections based on the correlation between them and later the data is turned into grayscale. The studies conducted using NSLKDD dataset exhibit great accuracy and are less difficult. In order to classify objects into binary and multi-class categories, the proposed method has been contrasted with deep learning and machine learning methods[32].

Particle Swarm Optimization (PSO) has also been utilized for optimizing parameters required to identify infiltration as indicated by Elmasry. A double PSO technique has been utilized to pick feature subset and hyper-parameters to address the concerns of redundant and irrelevant features. The optimized parameters have been provided to deep learning methods as Deep Belief Network, Deep Neural Networks, and Long Short term recurrent neural networks. In comparison to data that was not optimized, the research's findings show an increase in detection rate of 4-6% and a decrease in false alarm rate of 1%. The presented work has important significance as deep learning based approaches deliver good outcomes without much training of the network. Also the specifics are hidden from the user hence decreasing the complexity of the system[33].

2.6.3 Evaluation of Network-Based IDS

Network-based IDSs are known generally to generate lots of false negatives and positives. Many of the initial network-based IDSs developed applied signature-based detection to identify recognized simple threats. New skills have deployed blended methods of detection to actualize accuracy rate that are high and increase the attacks' types that can be detected by the system. Thus, the rate of false positive and negatives have decreased. The need to tune and customize the system to consider the attributes of the experimental surrounding takes a significant amount of work which is a problem.

Moreover, network-based IDSs have several important restrictions, they possess extensive detection capability. Handling hefty stream of traffic loads, launching counter attack against IDSs and ability to analyze encrypted traffic are the most important capability of the NIDSs. Attacks launched on scrambled traffic of network and performing complete scrutiny in case of high load are some of the limitations of the NIDSs. In addition, sensors from the IDS might make several events to go undetected, especially if stateful procedure study is utilized.

2.7 Host-Based IDS

Host-based IDSs (HIDS) scrutinize a host's properties and events to identify probable attacks. Some of the events monitored by a host-based IDS include file access, system logs and traffic information and modification[34]. Agents are detection software installed on interest hosts which most HIDS uses. A single host's activity is monitored by an agent. Data gathered by agents are forwarded to management servers that uses database servers. Managing and monitoring are done using the consoles. Special devices that run the agent software directly on the hosts are employed

by some host-based IDS instead of the installation on individual host. Each of these devices are placed on individual host which monitor that particular host's activity. Technically, these devices can be grouped as network-based IDS. Each device is designed precisely to shield one of the following:

i. Server: added to monitoring some application, agent can also monitor the operating system on the server.
ii. Client Host: Observation of Operating Systems, general application such as web browsers and email client are some of the functions performed by agents designated to monitor user's hosts.
iii. Application Service: Explicit applications such as database server program or web program are monitored by some agents only. Such agents are called application-based IDSs.

2.7.1 Security Features of HIDS

Host-based IDSs propose diverse security abilities. These are logging, detection and other features.

(a) **Logging**

Host-based IDSs frequently log wide-range of data on events detected. These are data that are utilized to confirm alarms, scrutinize events and compare events from other sources. Generally, host-based IDSs record data fields such as:

i. Date and time
ii. Information on application
iii. Event or alert types
iv. Information of port

v. IP address

vi. User IDs and filename/paths.

(b) **Detection**

Most host-based IDSs are capable in detecting several malicious activity types. The combined use of signature-based detection techniques is employed by host-based IDS in identifying known attacks, and the use of anomaly-based detection techniques such as rule sets or policy to detect formerly attacks that are unknown.

2.7.2 Related Work on Host-Based IDS

A host-based IDS, which comprises two detection technologies was designed and implemented. These are Back Propagation neural network and log file scrutiny technology[35]. Log file scrutiny was applied for misuse detection, and BP neural network was employed for anomaly finding. The goal of the amalgamation of these two detection know-hows is to illustrate that the anticipated HIDS can adequately boost the detection of invasion more accurately and efficiently. The result obtained indicated that the anticipated system enhanced the detection rate of attacks more efficiently and accurately.

A novel host-based anomaly intrusion detection method was proposed by using semantic algorithm[36]. The goal of this research is to discontinuous system call patterns so as to reduce false alarm rates while improving detection rates. In order to achieve this, the main idea was the application of semantic structure to kernel-level system calls to aid irregular behaviour's detection.

This novel approach was assessed using three different datasets. The new ADFA Linux Dataset (ADFALD) and KDD98 were utilized in the testing of the main performance, the portability and robustness testing were done using UNM dataset. From the authors' view, there was a noticeable better operation from the new semantic based algorithm than the present methods.

This study may probe the new procedures in order to improve the resilience of semantic features and reduce the training overhead.

According to research carried out, existing Host-Based IDS are slow at noticing attacks due to the use of the whole feature set[37]. In order to remedy the above-mentioned problem, the paper suggested a new host-based IDS called CPDT (Correlation based Partial Decision Tree Algorithm). The Partial Decision Tree for grouping traffic and Correlation feature selection for choosing features were fused together in CPDT. The algorithm is implemented and the evaluation is done by KDD '99 and the result showed an accuracy of 99.9458% was achieved. According to the author, CPDT's result indicate that its performance is higher than prevailing algorithms. This research can be improved upon using a novel method to identify the unidentified threats.

A new HIDS framework to decrease calculation and be resource intensive was proposed and implemented[38]. First, the system call traces is turned by the proposed framework into n-gram vectors and have the input size vector reduced by applying a dimensionality reduction. Several classifier models were scrutinized by reduced feature vectors using machine learning. In order to assess the output of the proposed model, an ADFA-LD dataset was implemented. From the obtained outcomes, it was able to detect intrusion effectively with low incident of high accuracy and false positive rate. The fine tuning of various parameters to accelerate this performance can enhance the study of this subject.

A novel Host based IDS to spot normal behaviour of a system using the sequence of calls system was considered[39]. This paper describes a well-organized anomaly-based Intrusion Detection System in terms of calculation using Recurrent Neural Networks. Instead of utilizing normal LSTM network but, rather the use of Gated Repetitive Unit made it possible to achieve results which are important with reduced training times. The combination of CNNs and GRUs help to improve anomaly IDS. ADFA dataset was implemented using the proposed technique. Based on the result obtained, it shows that combination of GRU/CNN is about 10 times faster compared to LSTM because of in training, the convergence is faster. In addition, there was a 60% False Alarm Rate and 100% True Positive Rate from the proposed system. By boosting the number of training sample or usage on diverse dataset can result to the improvement of this study.

According to research carried out, decades-old dataset sometimes remains useless for a long period of time, which could come about due to the fast changes in Operating System and the subsequent underlying complication[40]. In this research, by examining the up-to-date Linux kernel 5.7.0.rcl, they aimed at closing the gap that exist between the application environments and theoretical models. The feasibility of the system call-based HIDS residing in contemporary Operating Systems and the hindrances put on the developers of HIDS was examined by the environment. The kernel was examined for recent advances and a new plan was put forward on how to produce data and advanced the recognition model. Certain memory limitation and runtime surfaced which must be resolve in order to have the models operate at its envisioned limits.

A test was executed using the Leipzig Intrusion Detection Dataset (LID-DS), which happened to be a host-based IDS dataset formed in 2018[41]. In addition, an intrusion detection model comprises of vector-to-image processing, training and testing steps and host-based preprocessing is proposed to boost the operation of the system. The construction of a Siamese Convolutional Neural Network (Siamese-CNN), in the training and testing steps was done via a learning method

comprising of various stages with high output utilizing a slight amount of data. Types of attack are determined by Siamese-CNN built on the likeness score of each attack sample changed into image. Determination of exactness is done by calculation of few shot learning techniques. The comparism of performance between Siamese-CNN and Vanilla Convolution Neural Network (Vanilla-CNN) was done to determine performance evaluation. Judging by accuracy, judging from the F1-score indicators, recall, precision and accuracy, the proposed Siamese-CNN model showed a 6% approximately surge when checked against the Vanilla-CNN model. The projected model can be established by optimization of the hyper-parameter values of the increased exactness of intrusion detection of both known and unidentified cyber-attacks workability.

An overview of Intrusion Detection Systems was presented by the authors. The survey has covered several IDS approaches, including Network IDS, DIDS, and Host IDS. Additionally, the study has provided examples of detection techniques such protocol analysis, misuse-based, and anomaly detection. The survey has made reference of centralized and distributed systems using online and offline modes of detection[42].

Host-based IDS was presented by the authors, which is an amalgamation of usage detection in addition to supervised learning. Misuse detection is done with the help of OSSEC, which is an open-source Intrusion Detection System. It is capable of examining the log files. For detecting anomalies, back propagation neural networks (BPNN) have been employed. The log data has been gathered, pre-processed, and OSSEC has been used to analyze and report the findings for misuse detection. Since HIDS can detect intrusion solely on the host, the proposed technique has limited contribution. Moreover, it relies misuse based detection which is unable to identify new assaults[43].

The authors presented a centralized Host-based IDS design for private cloud computing environments. Using the system's resources as little as possible is the IDS's main objective. The

proposed model is assembled using OpenStack4, which is an open source platform. It is consisted of three nodes, i.e. compute, controller, and network nodes, and four modules for data collecting, pre-processing of data, IDS detection, and alert modules. The data collection module makes use of Log stash5 to gather logs from all Virtual Machines and stores them in Elastic search6 for the detection module's subsequent analysis. The detection module is based on C5.0 decision tree. In case of an anomalous activity, the detecting module warns the victim Virtual Machine. The model was verified on KDD99 dataset and compared to a traditional HIDS. The results suggest that in the proposed HIDS CPU use is 14% less, memory consumption is roughly 2% less, but detection rate is practically the same as the conventional HIDS which is 94%, and the detection time is somewhat more. However, the centralized nature of the IDS in the suggested technique increases the burden on the host machine. IDS techniques could be divided into three sub-categories, including computation-intensive approaches, artificial intelligence-based approaches, and biological concepts, according to Stavroulakis and Stamp's classification[44]. Though, this classification does not include all the attributes of detection approaches, but such a classification is apt in the current scenario since artificial intelligence based approaches including machine learning approaches are known to deliver good accuracy. A HIDS checks and gathers the features of hosts that hold sensitive data, servers, and other unusual behaviors. A Network Intrusion Detection System monitors the network packets with the use of sensors, and then flags suspicious situations by evaluating the activity across the network. Wireless Intrusion Detection Systems are like NIDS, but they collect the network traffic of wireless networks, like ad-hoc mesh and sensor networks. Adoption of different technologies like MIDS can help in more accurate detection. To avert the implementation of malicious code on the host machine, HIDS watches the logs of the system. But the application of HIDS is highly tough because of high false alert rate in the case of HIDS. To address the concerns like false alarm rates, semantic techniques have been used in this study. In the article, the semantic approach has been applied on the operating system calls to

detect anomalous behavior. ADFA-LD dataset has been utilized to use a semantic technique in order to detect intrusion on the host. ELM, The decision engine employed in the research is capable of fast learning rates at the same time it needs to be trained only once. But the overhead of processing time is larger. Host Intrusion Detection Systems (HIDS) have lately attracted interest amongst researchers. HIDS is able to identify harmful events on the host system. This article offers a discussion of several forms of IDS and describes a threat aware of Host-based IDS architecture. Different classic IDS architectures have been explored, and HIDS architecture has been presented in this study. Dispatcher is a component that distributes the input traffic to different analyzers. This influences the timing of detection and accuracy. Equalizer aids in data normalization. The correlation engine decreases the amount of warnings that need to be watched when an attack occurs.

They collected the properties of a system that includes the audit logs, events over the network, as well as the order of system calls. Network Intrusion Detection Systems like Snort monitors the network traffic by monitoring the data packets, and after that it analyses the traffic to discover if it contains the abnormal codes. With the technical evolution of the internet, network security issues have arisen as key considerations in web applications. Intrusion detection components strive to determine illegal and illegitimate actions by evaluating user behaviors across the network. The best intrusion detection system should have a high degree of accuracy in its ability to identify intrusions[45].

Cluster Centers and Nearest Neighbors (CANN), which the authors define as a feature depiction technique. The suggested technique changes the data to a single ‘‘distance-based feature,’’ which is further passed to a k-Nearest Neighbor classifier. However, this adds to an initial computational burden incurred in the lowering of the feature dimensions but yields greater outcomes in detection. Experiments conducted on KDD99 dataset suggest that CANN delivers higher accuracy, percentage of true positives, as well as false positive rate than other techniques

such as k-NN or SVM classifiers. The presented work has important contribution as cluster based approaches save the effort on the nodes as the load is dispersed across the nodes in the network. However, CANN is unable to identify user to root (U2R) and Root to Local (R2L) attacks[46].

System log analysis is the foundation of the host-based IDS architecture discussed by the authors. The five modules that make up the proposed architecture are: one for log collection and pre-processing; another for saving and updating; one for search and analysis; and one for alarming. The system logs are gathered by the four modules, who then turn them into records with fields that are taken directly from the system logs. These records comprise ‘‘Facility’’ and ‘‘Severity’’ fields, plus a header that contains the ‘‘Timestamp’’ and ‘‘Hostname’’ data. It also comprises ‘‘Tag’’ and ‘‘Content’’ fields. The record thus created is saved in a MySQL database. Regular expressions are utilized to extract the relevant records. additional, the records collected from database are converted to numeric values and passed to a back-propagation Neural Network (BPNN) model for additional examination. After the inspection is done, the alert module intimates the user[47].

They presented Host-based IDS which is an ensemble classifier that employs AdaBoost. They have incorporated a cognitive technique that assigns more weights to the weak classifiers. The researcher briefly examines Host-based IDSs and Network IDSs, and deliberates their architecture along with the applicability. They have also highlighted the downsides like increased communication and computation cost of certain techniques. An assessment of the threats posed by the cloud environments has been done. It also incorporates several intrusion detection and intrusion prevention technologies that handle the security challenges[48].

They introduced a HIDS dubbed Ghostbuster. The IDS creates user profiles for each user based on their file-system access patterns and it also looks for anomalies. The Linux program blktrace7 has been used to mine sequences of the events of file access. For each user, a profile is built. The

file access information is organized by sizes, frequency, and access patterns in the profile. Anomaly has been discovered using finite state automata and outlier analysis[49].

Performance evaluation is based on actual file accesses made by all 77 users over the course of 8 weeks, during which time 4 weeks were devoted to training and the remaining 4 weeks to testing, totaling 560 files accessed by users. Results suggest low false-positive rate and high detection rate.

In order to safeguard the virtual computers that are based in the cloud-based network, the authors proposed the (H-IDS) system. Logistic regression has been utilized for obtaining the relevant features of the classes. Various assaults have been categorized using approaches like neural networks, decision tree classifiers and linear discriminant analysis combined with bagging algorithm. The suggested strategy is explored on NSL-KDD data set. An accuracy of roughly 97.51% is attained to detect attacks[50].

The idea of HIDS for Android-enabled Mobile Devices was recently put up by the writers. Various Machine Learning methods are employed for profiling the malware's behaviors. The suggested model has been trained using both good and bad profile data. The proposed IDS had the benefit of being independent and not requiring any server connectivity. It mostly employs benign situations along with some malevolent examples[51].

The writers have described an industrial embedded device host intrusion detection system. The research claims to have considered the system, environmental and device specific aspects into consideration. The efficacy of the suggested architecture has been tested by constructing a prototype of Host IDS for industrial embedded devices. The system has been built in a Programmable Logic Controller (PLC) with a Real-Time Operating System. The evaluation has been done by establishing hypotheses and test scenarios. For the cloud context, a hypervisor-based intrusion detection system has been proposed. The suggested IDS are built on multivariate statistical analysis, which tracks changes and looks for unusual behavior. It monitors the

individual activity and connected instance behavior to detect intrusion. We use the fact that a hypervisor consists of a collection of instances to introduce an instance-oriented feature model, which departs from the traditional monolithic network IDS feature model and takes advantage of the individual and correlated behaviors of instances to enhance detection capability[18].

2.7.3 Evaluation of Host-Based IDS

The detection practices utilized in the system helps to determine the varying attacks host-based IDSs types identifies. Combination of these techniques are done by some host-based IDSs, while some make use of one or several techniques. With the enormous knowledge of host-based IDSs on host's features and configuration, the agent of IDS can most times determine the success of an attack that is not terminated. With other IDSs skills, false positive and negatives are more rampant. However, host-based IDSs cannot be very accurate in detection rate due to the fact that most identified events such as file system monitoring and log analysis context are unknown to most IDSs when it occurs. For example, the rebooting of a system by the user or the installation of a new application might occur, and these events might be malicious activities in nature. The detection of those events on their own are accurate, but often, with little or no additional information, there cannot be a conclusion if they are normal or are attacks in themselves. The accuracy rate of detection of host-based IDSs using a blend of various methods are higher than those using one technique. Since each technique is designed to provide monitoring of different aspect of the host, the use of more than one technique helps to gather more information about activities occurring in the system than if it was just one technique employed. This gives a clearer view of the additional information which can be used to aid the assessment of intent of event and to give a more comprehensive profile of event[52].

2.8 Network Behavioral Analysis

A Network Behavioral Analysis (NBA) system identify the unusual stream of traffic flow such as types of malware (for example, backdoors, worms), procedure violation and Distributed Denial of Service (DDoS) attacks on the network by examining the traffic on the network or network statistics[53].

NBA systems often have consoles and sensors; management servers are often used by some. NBA often have their sensors available only as gadgets. Some sensors work like the Network-based IDS's sensors which is able to sniff packets in order to observe network activities in single or more network sections. Other NBA sensors do not directly monitor networks, usage of routers and other network devices' providing information on network flow.

2.8.1 Security Features of Network Behavioral Analysis

NBA systems possess various types of security capabilities. These are gathering of information, logging and threat recognition. Some structures also offer information on security and management of event proficiencies.

(a) Collection of Information

NBA systems give comprehensive capacity on information-collection and need the knowledge of the characteristics of the mainframes for detection. Creation and storage of list of hosts communicating on the monitored network are done by NBA sensors. These are the information obtained by NBA systems for attacks detection:

i. IP address

ii. TCP and UDP ports

iii. IP protocols

iv. OS

v. Other hosts connected and services used.

NBA sensors consistently observe activities on the network for changes in these information. Collection of each server's stream is done constantly as additional information.

(b) Logging

NBA technologies record comprehensive data about events of detection of threat. Alerts validation, investigation of events and event correlation among other sources are data used for. Some of the data types that are often logged by NBA software comprise:

i. Date and time

ii. Protocols

iii. Additional package header fields

iv. IP addresses of source and destination

v. Activity or alert type

vi. TCP or UDP ports

vii. Number of bytes and packets.

(c) Detection

NBA knowledges naturally have the capacity to identify various activities types that are malevolent in nature. They are mostly anomaly-based detection, with some stateful protocol scrutiny methods embedded, which scrutinize the flow of the network. Signature-based detection are not mostly implemented in NBA technologies.

2.8.2 Related Work on Network Behavioral Analysis

According to study conducted, common Intrusion Detection System possesses restricted capacities and more often than not, are not able to identify malicious behaviour or issue an alert (false positive) when abnormality occurs in the network[54]. In this study, the use of Network

Behaviour Analysis system and the implementation of Data Mining (DM) procedures to network traffic data would assist in better development of Intrusion Detection System was thought of. Hybrid intrusion detection approach was proposed by the authors. Network Intrusion Detection was examined using NBA and DM approaches and with the claim that the effective network detection of intrusion was more with the combination of both approaches.

An IDPS technology called Network Behaviour Analysis System was scrutinized and assessed[55]. A Network Behavior Analysis System (NBAS) is primarily an IDPS (Intrusion Detection and Prevention System) expertise that checks traffic on network to detect attacks which can cause unfamiliar traffic flows such as some types of malwares and Distributed Denial of Service (DDOS) attacks. A comprehensive assessment of NBA techniques is presented in this article. First, the architectures and the technologies of NBA for deploying components as the major parts are explained. In addition, the abilities of the security of this technology, inclusive of the suspicious activity detection's methodologies are broken down in detail. The remaining section talks about the capabilities of managing the technologies, inclusive of recommendation for implementation and operation[55].

According to the work conducted by some group of researchers using Principal Component Analysis (PCA) and Network Behavior Analysis with the KDD cup 99 dataset in the identification of novel attacks or intrusions as well as known attacks[56]. Here, the PCA determination of the dataset and the NBA are used in the analysis of the network behaviour. The testing and training of proposed system is done by KDD cup 99 dataset. The evaluation of malicious dataset and detection is the main aim of this article.

Looking at the evaluation outcome, the planned technology has a more promising outlook in terms of detection precision and efficient calculation for real-time threats identification in contrast to earlier systems in the literature. This technique is very efficient in identifying novel threats and many familiar attacks. This work can be stretched to the usage of known datasets as well as online datasets.

APT (Advanced Persistent Threats) during initial intrusion, do modest attacks such as spear phishing, but after a long time from the initial intrusion, they upgrade their attacks into leaking of information, creation of backdoor and the scrutinization of network so as to spread malevolent code. In this research, a decision tree-based Intrusion Detection System that execute information based on behaviour scrutiny in order to spot changes in the APT attacks after the system has been attacked is hereby proposed[34]. The proposed system initially inspects the malicious code behaviour and then decides the protocol through the decision tree.

Reference to the results evaluation, the proposed system responds to APT threats promptly, identify the likely likelihood of intrusion and reduces the extent of the injury. The development of this research can be done using several research on dispersed APT security means both in the system and network by way creating a standard for the major behaviour formed in this process.

In their write-up, it proposed that at the cloud network layer, an intrusive detection approach which is both adaptive and lightweight called Behaviour Based Network Intrusion Detection (BNID)[57]. At the Cloud Network Node (CNN) is the analysis of the traffic behaviour done so as to detect intrusions. Deployment of BNID in the cloud has been proposed using a security framework. This action renders the need to use the IDS to every Tenant Virtual Machine (TVM) unnecessary. Both statistical learning techniques and feature selection are used by BNID for

traffic behavioral analysis. In order to assess the planned approach, the Information Technologies Operations Center (ITOC) threat dataset are used.

An accuracy rate of 98.88% and 1.57% false positives was attained by BNID. Using this proposed approach to detect unseen attacks can be used to improve this study.

In another study, it was suggested a new IDS called Hawkware, an ANN-based distributed IDS which analysis the runtime behaviour of device along the network traffic and runs on IoT device[58]. Detecting sophisticated attacks by analyzing device behaviour is the main is the design of Hawkware, as contrast to the traditionally expensive and deep data scrutiny used. Hawkware can be use on a Raspberry PI, can at a fairly tolerable level detect attacks and seen as light enough to be deployed. Development of this work can done to work in different systems and not only for IoT devices.

A behaviour-based Intrusion Detection System was proposed, that timely detection of malicious behaviour in the network using deep learning can be achieved. The proposed system is tested using UNSW-NB15 dataset and the projected system used the Bidirectional Long Short Term Memory Architecture. According to the authors, from the outcome of the experimental tests, effective detection of unknown or known malicious behaviour from earlier network environment can be done by the proposed system. The improvement of the study can be done by resolving the unstable dataset cataloguing problem.

2.8.3 Evaluation of Network-Based Analysis

NBA systems primarily operate by identifying when a behaviour is far from the normal ones, resulting in high accuracy attacks detection which result in the generation of a large number of activities on the network in a brief time and attacks with unfamiliar activity. Powerful detection

abilities on certain types of attacks are offered by NBA, but they comes also with significant limitations.

One of the limitations is that small-scale attacks are not effectively detected by NBA system. Especially, if the threat is executed slowly and the determined rules is not violated, the degree of detection is very low. This is as a result of the technologies of the NBA is implemented using anomaly-based detection methods; unless the numerous attacks deviation is very wide from the expected result, the effectiveness of the detection on attacks is low. Detection can be made if the DoS attack initiate slowly and increase over time but it might be too late. In cases where the sensors are adjusted to be very sensitive to irregular activity, this would generate alerts faster when it occurs, but the rate of false positive might increase[59].

Another important limitation is attacks detection delay. When streaming data from network devices and routers are used by NBA systems, they are behind in identifying attacks due to the data source. Data transmission in batches can take one minute to several hours period to the Intrusion Detection System. In the situation where the threat happens swiftly during this period, the system has already been tainted by the attacks or remain unnoticed until the harm is done to the system. Therefore, there is a need for powerful systems that would monitor directly instead of the data streaming approach so as to enable the NBA systems to capture threats and attacks quickly.

2.9 Intrusion Detection Methodologies

Intrusion detection procedures can be mostly categorized into three unique classes including:

i. Anomaly-based model;

ii. Stateful protocol analysis

iii. Signature-based model;

Each IDS method uses another approach to detect network attacks. For attacks that the types of threat are known, the fastest and most efficient tool to use is the signature-based detection, but when it comes to unknown types of attacks, it unsuccessful to identify the threat. Anomaly-based methodology is apt in detecting network-based attacks which have not been seen before, but issue out false alarm. From a different view, it sees and detect usual traffic as threat. In the case of stateful protocol practice, it can identify some types of new intrusion, it is complex and resource exhaustive but can't identify shrewd attacks.

2.9.1 Signature-Based Model

A signature can be said to be a pattern that is similar to an attack that is known. Signature-based identification is the method of checking for differences in patterns with the scrutinized events so as to identify probable threat [60].

Situation where there is a match in the process of comparing signatures, there is a warning or additional report generated by the system. Some signatures examples are: An attack attempt with the username "root" compromising the network safety, an email with the characteristic of known and malware that are common with a subject of "Free programs" or an Operating System with an indicator that the host regulator is incapacitated in the system record.

Signature-based discovery is the easiest method of finding due to the fact that events that are observed are examined against a list of signatures using a contrast process. A warning is issued should there be an existing distinct threat condition in the list.

Signature-based IDSs are active tools at identifying familiar attacks but are grossly inadequate at identifying existing attacks that are unknown or other forms of threats that are known. If an invader, for example, replaced a malevolent file "prog.exe" with another name such as "prog2.exe" a signature checking for "prog.exe" would not be exact.

2.9.1.1 Related Work on Signature-Based Model

According to the research carried out, Snort was used to implement signature-based attack detection[61]. While Intrusion Detection is working via Snort, DARPA Dataset was sent across the network and was primarily targeting the analysis of links that are abnormal identified during the transmission. A popular NIDS; Snort is used for inspecting network packets and making comparism of the packets to the database of signatures of known attacks. In addition, signature-based attacks' database of snort needs to be updated periodically. This IDS system has shown that both analyzing and detecting of intrusions in real-time network traffic can be handled by the system. According to the authors, the understanding of the concept of Snort-based IDS can be made possible to new user via this study. Analyzation and application of different intrusion detection tools can assist in the improvement of this study.

Signature-based IDSs has major challenges, one of such challenges is handling of enormous volumes of inbound traffic when each packet is examined and compared to all signatures in the database. In a situation, where the flood of traffic overwhelms the Intrusion Detection System, packets are dropped which can allow probable threat to be undetected.

The use of mobile agents was suggested by a team of researchers, who were working on a new Signature-Based Multi-Layer IDS model[62]. The proposed model has the ability to identify attacks with high efficiency of achievement rate by making and utilizing small and many databases automatically and dynamically. A mechanism is provided to update periodically these small databases via mobile agents. In order for the proposed database to be improved upon, it should be automated to have the capacity to allocate, improve and get rid of pattern between different databases of multiple IDS systems.

A procedure was proposed that by reducing the frequency of false alarm in signature-based Network Intrusion Detection System (NIDS)[63]. The pros, cons and classification of reducing wrong alert methods are given in signature-based IDS. To verify the claim, many of the prominent Security Information and Event Management that put this system to use coupled with their result were examined. According to the authors, in spite of all familiar methods, issues that needs attention are still present. This research can help security researchers use new post processing systems when dealing with signal from IDS. Other study areas that need future research can be done in order to boost the use of the projected systems.

In the study conducted, a decision tree algorithm working on the C4.5 decision tree approach was developed[64]. Split value and feature selection are consideration that are important for building a decision tree. In this study, these two issues would be addressed by the developed algorithm. The most important are when choosing the features, the information gained and unbiased classifier against the most frequent values when choosing the split based on the value it used. The proposed method was implemented on the NSL-KDD dataset and the experiment was followed strictly in accordance to the number of features. The duration used to develop the model and the accuracy received were used as metrics. According to the authors, the proposed Decision Tree Splitting (DTS) algorithm, when it comes to signature-based attack detection is an effective method to use. This study can be improved by minimizing the number of features used and the improvement of the split value.

In their study, it was purposed to decrease similar load of the signature-based model and increase the rapidity of the algorithm by parallelizing the algorithm implementing the pattern identical on the multi-core CPU[65]. In this study, a vector algorithm, Myers algorithm, is parallelized on a multicore CPU beneath the MapReduce framework. Acceleration up to four times approximately

was obtained with the multi-core application in comparism to the serial version. Two implementation of MapReduce was also used to parallelize the Myers algorithm. A prior Message Passing Interface (MPI) based execution was used in comparism with the implementation of the proposed method of the algorithm. Based on the result obtained, MAPCG and PhoenixCC MapReduce applications displayed 1.7 and 1.3 times progress over MPI, respectively.

The detection of intrusion was executed by developing a new system which uses a set of rules as a form of documented engine. PBID (Pattern Based Intrusion Detection) model was used, which validated earlier implemented SBID (Statistical Based Intrusion Detection) model. The dataset that was produced within the scope of this study was tested using the proposed model. The result showed 75% accuracy was obtained. From the authors, the combined results of the experiments, SBID and PBID methods produce a detailed system for detecting invasion. Though, signature-based attack detection cannot alone generate an effective result. Therefore, the development of this work can be worked on more by the integration of anomaly-based intrusion detection.

An intrusion detection model called AS-IDS was offered, which combines these two methods to identify both the strange and known threats in IoT networks[66]. Three phrases are contained in the proposed model: hybrid IDS, preprocessing and traffic filtering. The first stage entails the filtering of the network traffic at the IoT entryway using exact packet attributes, then the application of the Target Encoder, Z-score and Discrete Hessian Eigenmap (DHE) respectively at the preprocessing stage. At final stage, the combination of anomaly based and signature base model is done. At the part of the signature-based, Generalized Suffix Tree (GST) algorithm is implemented and classification of attacks based on matching signature are grouped into intruder, normal and unknown. On the side of the anomaly-based system, Deep Q-learning was used to detect attacks that are unknown and implement Signal to Noise Ratio (SNR) and bandwidth to

group attacks. NSL-KDD dataset has been used in real-time traffic for the application and testing by the proposed AS-IDS model. Result shows 96.9% attack detection rate achieved. This study can be improved upon by the extensive experimental result obtained been used on dissimilar datasets.

2.9.1.2 Evaluation of Signature-Based Model

Signature-based discovery is the easiest identification method and easily understood. Comparism is done by the system on activities such as packets or entry logs against registered signatures lists. Thus, giving users the ability to direct the signature based and also administrator of system, understand types of attacks that can raise alarm. Known attacks are effectively detected by signature-based IDSs but as unsuccessful with unknown threats, known threat's variant and lurking threats. To have success rate that is high, dispersed signatures must be stated for all attacks that can originate from an invader and the frequent updating of the database.

2.9.2 Anomaly-Based Model

Anomaly-based discovery is the procedure of differentiating studied events with definition earmarked usual to identify irregular events[67]. An anomaly-based detection system implemented IDS have guidelines that model routine network connection, behaviour of users, application, and hosts. The creation of these rules is based on the features of normal event over a long duration. For instance, the average time for web activity for usage during the business day hours can be a network rule. Using statistical methods, IDS can sniff majorly higher-than anticipated use of web usage and raise alerts while differentiating attributes of present action with guidelines. Rules can be generated for diverse behaviour traits, such as number of packets transferred over stipulated time, number of failed logins attempt and number of sent emails by a user. The most vital benefit

is ability of anomaly-based detection method to effectively detect previously attacks types that are unknown. For example, infection of a suppose computer by a new type of malware, consumption of computer's processing resources by the malware, enormous number of emails sent, many network connections initiated and other behaviour engaged in that differs differently from the computer's profile created.

Anomaly-based detection generated-rules are of two categories: static and dynamic. The static rules lists are not altered once it is created except when IDS is directed to generate new rules. Dynamic list is consistently altered as new activities are scrutinized. As network and systems undergo changes periodically, matching procedure of normal behaviour also change. Static list ultimately becomes obsolete; thus, it needs consistent updates. Dynamic profiles do not possess this issue but are likely to be commandeered by invaders. For instance, small amount of malicious activity can be performed by an attacker, then slowly surge up the frequency and the number of activities. If the amount of modification is slow enough, IDS may see these elicit behaviour as normal behaviour and add it in its outline. Unintentional enclosure of dubious activity as inclusive part of the rule is a collective delinquent with products of anomaly-based IDS. Additional challenge faced with anomaly-based IDSs is the difficulty of setting the rules right in some cases. For example, if an incident which generate large files to be sent once a month only, the consistency of observing such events is minimal, resulting in it been considered as abnormal activity and generate an alarm. Many false-positive are generated by anomaly-based products often caused by benign activity that depart considerably from the rules, particularly in dynamic or different situation. Another main issue with anomaly-based detection technique is its difficulty in deciding the reason of the alert or validate that the status is not a false alarm caused by the complexity and quantity of the procedures.

2.9.2.1 Related Work on Anomaly-Based Model

Anomaly-based Intrusion Detection System was presented by the team to lessen the number of fake alerts and upsurge efficiency[68]. Fuzzy rule-based modeling and fuzzy controller were used to bring up to date the model in the conception and testing phrase, individually. Inclusively, the user was presented with the result of the structure estimation. The handler then can authenticate the decision and the adjustment of the discovery model via response from the system operator is done by fuzzy controller. Evaluation of the system is done using the NCL dataset. KDD '99 dataset is parent of the dataset. According to the writers, using adaptive IDS, 20% significant improvement from the performance of the system was obtained from these test results. In addition, the accuracy of the system was improved by the proposed anomaly-based intrusion detection by 15%. Test on different datasets can also be done using this proposed Intrusion Detection System.

A combined machine learning algorithm based on K-Means clustering and the Naive Bayes Classifier (NBC) named KMCCNBC was suggested to maximize uncovering and precision while diminishing fake alarm[69]. Labeling process has been used in the K-Means clustering. Collection of all data in their matching clusters as routine or hostile based on their performance with K-Means, while the ungrouped data into the precise classes is re-categorized by the Naive Bayes Classifier. ISCX-2012 dataset consists of KMCCNBC and NBC performance evaluation. Based on the result obtained, increase in accuracy and detection rate up to 99% and 98.8%, was obtained by KMCCNBC respectively, while reduction of the fake alarm is 2.2%. This research can be expanded to contain feature selection methods.

Genetic algorithm (GA) based anomaly detection method was utilized by the team, which is among the most active evolutionary procedure in machine learning, for attack uncovering on the network[70]. Since the increase in the performance and accuracy of the system which has also reduce the rate of false positive, optimization of the rate of false positive is the main focus of this

study especially. Limitation experienced with additional accuracy procedure for their rate of untrue positive would be deliberated in this study. For the trials, KDD99cup data was utilized. From the gotten result, the detection swiftness can be boosted with the fitting feature selection while false alarm can be reduced. This study can be enhanced by implementation of dynamic feature selection methods for choosing additional essential attributes.

An anomaly-based technique was put forward in order to identify network variances, based on fuzzy clustering[71]. The planned method comprises of three phrases: Preprocessing, Feature Selection and Clustering. In Preprocessing stage, data that are duplicated get removed from the dataset. Then, important scrutiny of part is used to choose the unique attributes. In Clustering step, Robust Spatial Kernel Fuzzy C-Means (RSKFCM) algorithm was used for clustering samples in the network. RSKFCM is a variant of the standard Fuzzy C-Means that consider area information and utilizes the kernel distance metric. For the proposed method to be appraised, the EDA dataset which is a version of the KDD dataset, was utilized and matched with the usual techniques in the literature. Cluster validity indices, untrue positive rate and accuracy were utilized as measure of performance. 17.04% rate of false alarm and 86.3% accuracy were achieved as result. According to the author, other methods results were not as good as the proposed method. However, the improvement of this study can be achieved by the use of diverse approaches such as Evolutionary algorithm.

A new hybrid model operating on best features transaction data on the network to assess the invasion coverage threshold was created[72]. In the appraisal of the planned model, which is NSL-KDD dataset and 20% test dataset, which is multi-class problem and binary, were utilized. Considering the results acquired, there is a significant effect from the hybrid approach on minimization time cost and computation while deciding the feature association effect scale.

99.81% and 98.56% accuracy rate were achieved for the dual-class and multi-class NSL-KDD dataset, individually. Besides, issues such as low untrue negative and high false rates are there. In proffering solution to the problems, a hybrid method comprising of two major components have been planned. First, Vote algorithm and Information Gain, are used for choosing vital features that would surge up the accuracy of the proposed model using the combination of probability distributions. Then, Meta Pagging, AdaBoostM1, J48, Naïve Bayes, REPTree, Random Tree and Decision Stump classification algorithms were implemented in the mix algorithm. As an outcome, high false negative rate, low false positive status and improved accuracy were seen. Further development of this study can be achieved by the application of the planned technique on diverse datasets using dissimilar optimization methods.

In the research implemented, a process for choosing important attributes and an Intrusion Detection System built on two-level ensembles of classifiers are proposed[53]. In the bid to decrease the training datasets attribute's size, 3 diverse means were used: ant colony algorithm, particle swarm optimization and genetic algorithm. The use of Reduced Error Pruning Tree (REPT) for selecting features based on classification performance was done. Then, bagging methods and rotation forest, which is a two-level class of classifiers, are implemented. According to the authors, UNSW-NB15 and NSL-KDD dataset was attained, significantly performing more than other freshly projected classification techniques. The development of this study can further be done with novel methods that can attain sophisticated precision using lesser attributes.

An anomaly-based intelligent Intrusion Detection System (IDS) was proposed, called Passban that can guard IoT devices trustily attached to it[74]. This proposed system's feature can be installed trustily to cost-effective IoT gateways. Based on the positivity of this feature, it can use the edge computing paradigm, which uses the opportunity to identify cyber threat as close to data

source as possible. During the Passban evaluation phrase, the use of two different scenarios was used. In the initial situation, Passban was implemented as an IDS operating trustily on the entrance getting facts from the internet and the IoT devices. In the second situation, a special device considered the 'security in the box' that get traffic from the local gateway and the internet is implemented as the infrastructure element. Based on the assessment result, low false positive and high accuracy rates including attacks such as Port Scanning, SYN Flood, SSH Brute Force and HTTP were detected by Passban.

2.9.2.2 Evaluation of Anomaly-Based Model

Anomaly-based detection is built on the value of matching traffic with activity considered usual in order to detect diverse events. Anomaly-based Intrusion Detection Systems are more preferred than signature-based systems, because they do not need previous knowledge of attacks pattern to identify a threat, but the difficulty to manage the alarms are higher than the signature-based Intrusion Detection Systems. This is because reports are generated along with the reported alarm by Signature-based IDS, while anomaly-based IDS detect malicious threat using the traffic flow. The anomaly-based IDS whenever an event that departs from the elementary outline of usual behaviour is detected, an alarm is raised, but the reason for the anomaly is lost to the Intrusion Detection System. The ability to manage alarm and been able to tell the difference between true alarm and false positive is a major challenge. Therefore, while the detection of unknown attacks can be generated by the anomaly-based detection system, it is imperative to know the class of the detected attacks.

2.10 Intrusion Detection Approaches

There are numerous methods to spot invasions in the computer networks including statistical, rule, heuristic, cloud, pattern, deep learning based and machine learning. The name varies based upon the procedures and stands that are utilized during the procedure of detection.

2.10.1 Statistical-Based IDS

Statistical based (statistic-based) intrusion detection system builds a legitimate profile by observing normal transactions[75]. When the normal (legitimate) profile differs from the observed event, it indicates attacks. For each transaction, intrusion that deviate from normal traffic is assigned a score. When the threshold value is smaller than the measured score, alarm is raised.

The number of activities that occurs in the stated time period determines how the value of the threshold is set. In building normal profile, statistical metrics including variance, standard deviation, mode, mean and median were used[76].

The Statistic based techniques vary. It can be allocated mainly into three groups: multivariate, time series and univariate mode. These statistical based methods can be distributed into subgroups such as operational model, time series model, statistical moments, parametric and nonparametric models, markov process model, threshold metric and multivariate model.

Some advantages of statistic-based IDS include some of the listed items below:

i. It can identify current attacks because it doesn't use attack signature.
ii. Detection of DoS and DDoS attacks
iii. Maintenances is easy because update is not needed

On the contrary, the following are some of the disadvantages of statistic-based methods:

i. Normal profile can be altered overtime
ii. It is time consuming building a normal profile

iii. Effectiveness and accuracy on the statistical distributions used

2.10.2 Rule-Based IDS

Rule based IDS implement guidelines when identifying likely network traffic intrusion[77]. The rule-based IDS uses less manual work than technology. The rules could be considered as pattern of patterns. When rules are mined, rules are derived from the attacks pattern by the use of Artificial Intelligent (AI). Many attacks can be recognized using a single rule.

To distinguish number of attacks and same type, few rules are needed in a rule-based approach while thousands of signatures are needed in signature-based approach. Rule-based IDS are easily maintained. Additional rules can be added by the service provider to the system after the definition of the first rule. Due to the fact that simple change in attack can't affect the intrusion pattern completely, new attacks are easily detected by rule-based detection system. However, several rules are needed to completely detect potential threats in the network.

2.10.3 Heuristic-Based IDS

Heuristic based IDS checks for malicious behaviour based on intrusion. The heuristic-based IDS creates an observation model that repels any other behaviour but accept specifies acceptable behaviour. Heuristic approach requires experience and knowledge in order to correctly identify attacks. In the heuristic approach, any suspicious behaviour is scrutinized in the collected execution traces the alert is activated should any suspicious behaviour pattern is identified. Timely, whenever new types of attack are detected in the network, malicious behaviour list is updated. Well known and current attacks are detected in heuristic based approach. However, some attacks types can circumvent the heuristic detection engine by using hiding techniques.

2.10.4 Pattern-Based IDS

Pattern based approach detect strings, forms and characters to withdraw patterns that are meaningful in the data collected and detect attacks based on the pattern[78]. The pattern based approach stipulates instructions for malicious sequence to attacks. Signatures are recognized patterns in Intrusion Detection Systems. Attacks detection is faster and efficient with pattern-based approach, but fails in detection of recent attacks since their signatures (pattern) are foreign to it. Execution of pattern-based approach is very effortless. Pattern matching algorithm could be grouped into two types plus multiple and single signature matching. Scanning by current IDS multiple times is avoided using multiple matching algorithm which makes it effective for use[79]. However, more processing and memory is needed.

2.10.5 Cloud-Based IDS

Cloud computing is one of the furthermost auspicious technologies which sustains multiple online services efficiently[80]. Cloud computing offers limitless facilities with lower cost, more calculation power, pay-as-you-go and 24/7 data access. Three types of services are provided by cloud environment: PaaS (platform as a service), IaaS (infrastructure as a service) and SaaS (software as a service)[81]. Complex activities over the network are one of the great benefits offered by SaaS to users. Users can implement easily various algorithm and techniques on huge software platform offered by PaaS. Companies' infrastructure as well as central computing devices, storage resources and network, as well as users are provided by IaaS. Several advantages abound developing IDS on top of cloud environment.

Cloud environments provides detection of network events that are malicious in nature, from various angles and overcame the weakness of the classical intrusion detection[82]. In addition, the usage of private and public clouds offers the prospect to identify diverse types of network intrusion aligned with high performance.

Cloud intrusion detection, user data collector and cloud services are the three components, Cloud-based IDS is made up of. Collection of standardize, network packets, filter and transmitting to the cloud are done by User Data Collector, which is an independent server. Analysis and validation of data received from the user data collector, translation of the data into a generic format for cloud intrusion detection component are done by cloud service[119]. The core portion of invasion detection at the cloud is cloud intrusion detection component. Taken of facts from the cloud service module, packet analysis and intrusion specification are done by cloud intrusion detection module. It comprises of signature database, analysis engine and service console. The cloud-based IDS is still in its initial development. The application of this technique should be implemented farther in future Intrusion Detection Systems so as to boost model conduct[118].

2.10.6 Machine Learning-Based IDS

The aim of this machine learning is to make scrutiny process automated without human involvement. In order words, describing data in algorithmic format. Different learning method can be used by Machine Learning including semi-managed (few tagged, several unlabeled data), managed (labeled/labeled data) and unsupervised (unlabeled data)[120]. This is a new method that is used for detecting invasion. There are wide range of procedures and algorithms which IDSs uses inclusive of fuzzy logic, k-nearest neighbor, neural networks, decision trees, support vector machines, Bayesian algorithms and genetic algorithms. Recently, several papers are in print which uses the Machine Learning approach for the use on IDSs[83]. Basic merits of ML method includes high outcome, flexibility, adaptability and detection of new attacks types. From another view, there are some drawbacks of ML-based IDS which are listed as the following

i. Attack complication is on the rise
ii. Domain knowledge is not taken into account by algorithm

iii. The processes are (parameter choices, feature engineering, etc.) more vital than the algorithms
iv. Outliers handling is difficult all the time
v. ML algorithms are susceptible to bias (history, knowledge)
vi. Unknown attacks pose a challenge in cases of detection and prevention
vii. High dimensionality (hundreds of proportions are conceivable)
viii. Difficult in preprocessing of data (changing data format into appropriate format from system monitoring to analysis)
ix. ML algorithms generates data assumption
x. Need contextual features not just IP addresses
xi. Size of data is large (millions of network connections)[120].

Machine Learning Algorithms used in IDS Machine learning belongs to the family of artificial intelligence, which employs software application to create outcomes that are more accurate without being expressly taught to do so. Based on previous data they develop new output values. Knowing what type of data you have is the first step because this strategy is data driven. This part aims to know about various machine learning methods used to implement IDSs. Support for Naive Bayes, decision trees, clustering, SVM (support vector machine), KNN (K-nearest neighbor), ANN (artificial neural networks), LR (Logic regression), and hybrid approaches. Artificial Neural Network (ANN). The design idea of ANN is based on the brain activities generated from the human behavior. These are the artificial neurons, which are a group of interconnected nodes. An ANN has three layers: the input layer, the hidden layer, and the output layer, where every node is fully interconnected. All of these nodes must be trained prior to deployment in order to complete the work, however due to their laborious structure, training

these nodes takes a lot of time. The back propagation approach, which cannot be employed for deep network training, is typically used to train ANN models.

i. Support Vector Machine (SVM): It is used to distinguish the maximum margin hyperplane in the n-dimensional feature space. Since the separation of hyperplane can be done even with tiny amount of support vectors these methods produce remarkable results (Pressley). But near the hyperplane these approaches are delicate to noise and furthermore they are capable of solving linear issues.

ii. K-Nearest Neighbor (KNN): This is one of the basic method of classification which is non -parametric and very efficient in classification. According to Tirumala, Sathu, and Sarrafzadeh, a sample is considered to have a probability of belonging to a class if the majority of its neighbors are also members of that class. The performance of KNN is reliant on parameter K that is found by the user. According to the sample test conducted, K training points are selected by considering the shortest distance to the test sample and consequently the performance of KNN is significantly dependent on the parameter K. The value of K determines the model's complexity and its capacity for fitting data.

iii. Logistic Regression (LR): For solving the classification, problem these models are used which works by computing the probability of different classes by considering the parametric logistic distribution. These models are efficient due to their capacity of furnishing probabilities and to divide new data based on continuous and discrete datasets. Logistic distribution is calculated with the formula given below.

$$P(Y= k—x) = \frac{e^{x*w_k}}{1+\sum_k^{K-1} e^{x*w_k}} \quad .....1$$

where k = 1,2...K-1.

iv. Naive Bayes: Working principle of this method is conditional probability and attributes independence hypothesis proposed by the authors. These network acts as directed acrylic graphs in which every node acts as a discrete random variables of interest. The conditional probability table (CPT) is maintained, where each node carry the random variable state, which is used to specify the conditional probability of the domain variables with other connected variables. The formula for conditional probability is given below:

$$P(X = x \mid Y = C_k) = \prod_{i=1}^{n} P(X^i = x^i \mid Y = C_k) \quad \text{.....2}$$

v. Decision tree: These algorithms are used in classification problems for learning and modelling a dataset. Classification of new data set is done based on what it has got from the previous dataset. By using decision tree algorithms, inappropriate and unnecessary features are excluded automatically. The learning steps involved in this model is, selecting the feature, generating tree and tree pruning. In order to train a decision tree model most appropriate features are selected individually and child nodes are generated from it.

vi. Clustering: It is the task of segregating the data points into a number of groups, such that highly similar data are grouped into one cluster and less similar data are grouped into another cluster. Benefits of this algorithm are they do not need prior knowledge. It is mandatory to refer external information while detecting attacks using clustering algorithm. K-means is an example of clustering algorithm. K-means is an exemplar of clustering algorithm which makes the use of Euclidean distance to compute centre of cluster and data. K refers to the number of clusters and mean stand for attribute mean.

The idea behind this algorithm is to achieve less distance inside the same cluster and to have maximum distance between clusters.

2.11 ML-based NIDS Observation

ML/DL techniques have been used to develop NIDSs, such as Artificial Neural Networks (ANN), Support Vector Machines (SVM), Naive-Bayesian (NB), Random Forests (RF), self-organizing map (SOM) etc. The authors implemented a NIDS based on a restricted Boltzmann machine (RBM) for feature reduction and a support vector machine (SVM) for classification. The accuracy of the system is approximately 87%. The authors developed a network anomaly detection system using discriminative RBM in conjunction with generative models with good classification accuracy abilities to gather knowledge from training data[84].

2.11.1 ML/DL Approach used for NIDS

Eight tree-based classification algorithms are evaluated in predicting network events. The decision tree algorithm is used for feature selection and a random forest algorithm is applied as a classifier for NSL-KDD dataset. They deployed a principal component analysis (PCA) algorithm for feature selection and a support vector machine as a classifier to select the optimum feature subset[85].

The authors implemented flexible NIDS using self-taught learning on NSL-KDD data for network intrusion and developed a sparse encoder for further reduction. They also used soft-max regression as a classifier and evaluated their model independently on training and test datasets with an accuracy on training data 92.48%. Most of the approaches used training data for both training and testing purpose, They used separate training and testing data for training and testing

which provides accuracy of detection techniques[86]. They experimented that if they tested their proposed classifier in different training data, performance degraded[84]. The experiments showed a random tree model holds the high accuracy and low false alarm rate in detection system as a classifier[27].

2.12 Software-defined Networking (SDN) Based NIDS

The separation of the control plane and data plane, which simplifies packet forwarding, is one of the elements of the Software-Defined Networking (SDN) architecture. Real-time feedback control capabilities and open interfaces with modular plug-in features are elements of the SDN's centralized controller[87]. The centralized controller offers an abstract view of the network, defining tasks with APIs and allowing for more network programmability. It can integrate security devices into the network topology, which can lead to increase in accuracy, detecting security incidents and simplify management[88]. In this section, we first describe the architecture of SDN and applications, followed by SDN-Based NIDS observation utilizing ML/DL[86].

2.12.1 SDN Architecture and Applications

Open Networking Foundation (ONF) is one of the ideal architecture for SDN; it is separated into three primary functional levels. These are the application layer, the control layer, and the infrastructure layer. Figure 3, depict the overview of SDN architecture, the upper layer is the application layer; the control plane is in the center and data plane is the lower layer which is also known as infrastructure layer.

i) **Infrastructure Layer**: Infrastructure layer is also known as data plane. It largely consists of forwarding devices (FEs) including physical switches which interconnected over wired or

wireless media. Examples of hardware switches are Juniper, HP etc. and virtual switches such an OpenvSwitch.

ii) **Control Layer**: Control layer is also known as the control plane; it consists of a set of software-based SDN controllers providing a combined control capabilities through open APIs to supervise the network forwarding behavior through a public interface. Three communication interfaces allow the controllers to interact: southerly, northbound and east/westbound interfaces. Southbound APIs achieves communication between the controller and the physical networking hardware. SDN North Bound Interfaces (NBI) interact between SDN application and control layer that provide generic network overviews. The east-westbound interfaces using largely to interact between controller to expand controls inside a domain.

iii) **Application Layer**: The higher layer, application layer comprises of the end user business application such as network monitoring and security apps. Using enhanced characteristics of SDN, number of SDN applications have been developed to increase flexibility of a network, lower the total time to market and total cost of ownership of future IT network infrastructures. SDN has found applications in a wide number of networking channels. Furthermore, due to the recent increase in the frequency of cyber-attacks, SDN architecture has been employed for rapid development and deployment of new services. In this section, some of the important uses of SDN are explored.

Wireless Communication: The programmability aspect of SDN paradigm brings new applications to mobile communication networks. SDN has the capacity to fine tune mobile communication performance. The SDN architecture can be used in wireless network contexts including internet of things (IoT), wireless mesh networks, wireless cellular communication, and Wi-Fi access networks. Scalability can be added by utilizing the SDN and IoT paradigms. As a result, SDN opens up possibilities for network connectivity and bandwidth sharing by streamlining

management and traffic engineering in wireless mesh networks and implementing crowd-sharing models

iv) **Data Centers**: In a data center context, optimal traffic engineering, network control, and policy implementations are essential when operating at high sizes. We can automate and dynamically add security into data centers while reducing network latency via SDN-based traffic orchestration[115].

v) **SDN-Based Cloud**: Combining cloud techniques and SDN paradigm allows a close integration of applications on the cloud. With the network programmable interfaces and automation, SDN is a good technique to combat cloud incursion. Thus, SDN promotes the service scalability in cloud systems & home environment: SDN architecture offers users and service providers' more visibility into home and small office networks[89]. SDN may build anomaly detection systems in a SOHO network leveraging programmability for higher accuracy and scalability[90].

2.12.2 SDN-based NIDS Observation Using ML/DL

SDN-based Intrusion detection system employing ML/DL technique exhibits various advantages in terms of security enforcement, virtual management, and Quality of Service (QoS). SDN provides us an opportunity to increase our network security and enables flexibility to program network devices and reduces hardware dependency.. An SDN network with software switch implementations and programmable feature can be designed utilizing simulation and emulation platforms. Open Flow is one of the most common protocol standard that facilitates the application of the SDN concept in both hardware and software settings. There are various simulation tools, such as NS-2, Mininet, NS-3, OMNeT++ [91].

The key aspect of an SDN networks the SDN controller, often known as a network operating system. SDN controller is responsible for focusing communications with all programmable

elements of the network, offering a combined picture of the network.. Deep learning is capable of automatically discovering a correlation in the data, hence it is a promising way for the future generation of intrusion detection algorithms[92]. DL based approaches beat conventional machine learning techniques when applied to various categorization challenges in SDN networks. Most of the supervised ML algorithms are strong at classification problems, but not at modelling logic. DL based approaches surpassed conventional machine learning techniques in logic modelling. As threats are unknown, unsupervised learning methods such as stacked autoencoder, RNN and hybrid based algorithms will be the ideal for NIDS implementation in SDN platform. In recent years, researchers are implementing ML based NIDS in SOHO networks utilizing SDN environment and it was observed that the IDS accuracy has substantially increased due to ML based algorithms and scalability of SDN[93].

Deep Learning algorithms are a recent update to artificial neural networks that use abundant, affordable computation. Deep learning lets an algorithm to learn representation of data with various levels of generalization. These algorithms have been applied to optical object recognition, object detection, detecting network intrusion and many more domains. A deep learning system can be trained as a supervised and unsupervised approach. Deep Learning algorithm in a supervised way: Convolutional neural network (CNN) is generally trained in a supervised fashion. CNN is presently the benchmark model for the computer vision purpose. The CNN architecture utilized to build 2D pictures and a most important use of CNN is facial recognition. Deep Learning algorithm in an unsupervised way: An autoencodel is used to learn a representation (encoding) for a set of data for the goal of dimensionality reduction. A Deep Belief Network (DBN) can learn to recreate its inputs when trained with a set of examples in an unsupervised method. The layers then operate as feature detectors on inputs. A DBN is subsequently trained to perform categorization after this learning phase in a supervised manner.

Dimensionality reduction, regression, collaborative filtering, feature learning, topic modeling, and other processes are all made possible by DBNs, such as restricted Boltzmann machines (RBMs) or auto-encoders. & Deep Learning method in a supervised or unsupervised way: Recurrent neural network (RNN) methods are considered as a supervised or unsupervised learning method. RNNs can process inputs in random order by using internal memory. RNN is frequently used for speech recognition . RNN is effective at character prediction in text and also has the capacity to learn dependencies and actual evidence that is stored for a long period[94].

2.13 NIDS Using Artificial Intelligence/Machine Learning

The method presented by the authors uses a Convolutional Neural Network (CNN) with a multi-layer perceptron for its model. Multi-layer perceptron can be seen as a completely connected network where a neuron belongs to one layer and is connected to all neurons in the following layers. For neural networks, a CNN is composed of an input layer, hidden layers, and an output layer. Contrary to a standard neural network, in one of its hidden layers, CNN performs a mathematical process called a convolution instead of employing a multiplication matrix. The input for CNN is a tensor composed of several properties, including the quantity of inputs, their height, width, and channels. The input is convolved by the convolutional layer, which then sends the result to the following layer. However, the tensor size might rise substantially after numerous convolutions.

To tackle this issue, they utilize padding to minimize the tensor dimension. They trained their model by adjusting the hyperparameters until a drop in performance is satisfied. Their final model includes ten classes (nine for assaults and one for regular traffic) and is built of multiple dual convolutional layers followed by a pooling and a dropout layer to avoid overweight. However, their model shows a class imbalance between the top and bottom class which demands

the usage of bootstrapping to overcome the issue. They tested their model on the pre-partitioned UNSWNB15 dataset and on a user-defined dataset which corresponds to 30% of the overall dataset. They obtained, respectively an accuracy of 94.4% and 95.6% for both datasets[95].

The presenters proposed another IDS based on CNN. Their solution is composed of two parts. The first one is offline training using CNN, where in their model, they start with an input layer of 9 x 9 and decrease it through consecutive convolutional layers and a maximum pooling layer to achieve an output layer of 1 x 1. The second element of their system is the online detection phase, when they use Suricata, an open-source IDS, to catch the traffic.

After pre-processing the packets, the trained model is applied to the network traffic to provide the detection result. Researchers applied the CICIDS2017 dataset to the model's testing. Both the feature dataset and the raw traffic dataset were used for testing. Researchers correspondingly attained accuracies of 96.55% and 99.56%, demonstrating that their model performs better with actual traffic data than it does with an extracted feature set[96].

The authors proposed an ensemble strategy to identify intrusion. They perform three experiments to show how their strategy proposed superior results. They first performed normalization on the KDD Cup99 dataset, then, they apply a correlation method to accomplish feature selection. The feature selection employed information gain as a decision element, and finally, they apply an ensemble technique incorporating three algorithms: Naïve Bayes, PART, and Adaptive Boost. The outcome is then determined by averaging the outputs of the several algorithms or by the majority of votes. In addition, they apply the bagging method to reduce the variance error. Researchers attained an accuracy of 99.9732% on the KDD Cup99 dataset using their method[97].

They suggested a new model for intrusion detection systems, employing a hybrid multi-level model integrating SVM (Support Vector Machine) and ELM (Extreme Learning Machine). In their model there are five levels, the first level separates the traffic into DoS or Other. The second level distinguishes the prior unknown traffic into Probe or Other. The third level separates the

previously unidentified traffic into User to Root (U2R) attacks and the fourth level separates the previously unidentified traffic into Remote to Local (R2L) attacks. The fifth level is where the preceding unidentified traffic is finally separated from normal traffic. Since R2L and U2R are identical to standard connections, they are positioned at the bottom level. A classifier is utilized at each level. Their model is built of 4 SVM classifiers at levels 1, 3, 4, and 5 and includes 1 ELM classifier at level 2. Since ELM has demonstrated superior performance to SVM, researchers decide to employ an ELM classifier to identify Probe. Researchers utilized a modified K-means for feature extraction after preprocessing the training set from the KDD dataset to obtain the 5 distinct categories that their method can identify. Using their method, they attained an accuracy of 95.75%, which is somewhat better than if they merely utilized multi-level SVM (95.57%). Additionally, compared to multi-level SVM at 2.17%, their hybrid model's false alarm rate is lower at 1.87%[98].

They offered a technique employing oppositional tunicate fuzzy C-mean for detecting cloud invasions. In their model, they first pre-processed the data and did a normalization to have two datasets, one for training and one for testing. They performed a feature selection using logistic regression to maintain the more relevant features, and they employed the OPTSA and FCM clustering model. The dataset is separated into C clusters using the fuzzy C-means algorithm. Once the data is clustered, they executed a cluster expansion and integration to remove redundant clusters. Researchers tested their method on other datasets such as CICIDS2017 and attained an accuracy of 80%[99].

They designed an intrusion detection system for wireless networks based on the random forest algorithm. Researchers first established a signal detection model to catch the crucial properties of signals, then created the model to detect hostile nonlinear scrambling incursion signals. An improved random forest technique was utilized to extract the spectral properties of the malicious signal, and then, optimal detection of malicious traffic in a wireless network was done using a

reinforcement learning method and static feature fusion. They obtained a mean accuracy of 96.93%[100].

They developed a method that uses an outlier detection strategy to find unidentified attacks. The outlier identification strategy is based on recognizing data points that are isolated from clustered ones. This strategy employs the neighbourhood outlier factor to discover points that are not close to each other. Researchers used the KDDcup99 datasets to test their solution. In addition, the key advantage of their method is its execution speed which is substantially better compared to other solutions such as back propagation neural network which demands a lot of computational resources[101]. An enhanced Nave Bayes approach for intrusion detection systems was proposed. The Naïve Bayes algorithm is based on the Bayes equation:

$$P(H|U) = \frac{P(H|U) * P(H)}{P(U)}$$

where:

U is the data with an unknown class

H is the hypothesis class of U

P() is the probability

The Nave Bayes algorithm has a problem when one of the probability is 0. This results in its limited precision when used. In their answer, the authors presented two improvements to the Naïve Bayes method. The first one is deleting any variable that has a probability of 0. When the probability is 0, the second modification is to substitute an addition operation for the multiplication operation. In their solution, they first accomplished a feature selection utilizing the correlation-based selection of features (CFS). As a result, there are now only 10 features instead of 41. On the NSL-KDD dataset, where they tested their two modifications, the second one produced positive outcomes with a success rate of 89.33%.

They developed a new solution to improve IDS using SVM. Their technique uses Naïve Bayes algorithm to do feature selection. Then, researchers trained the model with the altered data from the feature selection. Researchers evaluated their approach on the UNSW-NB15 and the CICIDS2017 datasets. Compared to utilizing solely the SVM classifier, the usage of Naïve Bayes for feature extraction before employing the SVM classifier offers better results. Indeed, they got an accuracy of 93.75% on the UNSW-NB15 dataset and an accuracy of 98.92% on the CICIDS2017 dataset. However, their method just displays if there is an incursion, it cannot be utilized to detect what kind of assault is in operation[102].

They devised a solution to detect infiltration in wireless networks. Researchers approach was founded in cloud computing to have a maximum efficiency in terms of power for computation. They used sink nodes based in the fog to minimize the pressure on the cloud computing section. In order to have a solution as light as possible, they employed a mix of Polymorphic Mutation (PM) and Compact SCA (CSCA), as CSCA helps to lower the processing burden by reducing the density of the data by leveraging probability. They incorporated Polymorphic Mutation to mitigate the loss of precision when utilizing CSCA. They used PMCSCA to adjust the parameters of KNN algorithms to have the optimal configuration. On the UNSW-NB15 and NSL-KDD datasets, they examined their solution. They, respectively attained an accuracy rate of 99.327% and 98.27%[103].

CNN-based solutions were put forth by them. Researchers started by extracting features using both PCA and AE, or Principal Component Analysis and Auto-Encoder. AutoEncoder is a dimension reduction method using numerous hidden layers of neural networks to remove insignificant input. The data was then changed from having a single dimension to having a two-dimensional matrix, which was then sent to the CNN model for training. The model is trained and refined using back propagation procedures. Researchers tested their model on the KDDcup99 and attained an overall accuracy of 94%. Research model performed somewhat better when compared

to DNN and RNN models. Researchers model, though, has a poor rate of detection for U2R and R2L, which are underrepresented in the sample[104].

They suggested a multi-layer methodology to detect attacks. Their solution integrated two machine learning techniques: CNN and GcForest. The GcForest is a random forest technique which builds a cascade structure of decision trees. Their model is built of two basic sections. In the first stage, they run a CNN algorithm to recognize different sorts of attacks and normal traffic from the input data. Their CNN method is an upgraded model of GoogLeNet dubbed GoogLeNetNP. The second portion consists of using a deep forest model to construct more subclasses of the assaults. This second layer increases the precision of their answer by dividing the abnormal classes into N-1 subclasses. The second layer employs the gcForest cascade idea, however XGBoost is used in place of random forest. XGBoost is similar to a random forest, but the trees are built one at a time until the goal function is optimized. Using the UNSW-NB15 and CICIDS2017 datasets together, they evaluated their solution. They acquired a cumulative accuracy of 99.24% which is superior compared to the algorithms utilized singularly[105].

They suggested an IDS model based on Few-Shot Learning (FSL). A deep learning technique called FSL can learn from little to no data. In their approach, feature extraction was carried out using two embedding models: CNN and DNN. Those models allow to minimize the dimension of the input data without losing crucial information. Researchers used the UNSW-NB15 and NSL-KDD datasets to test their model. Their answer obtained, respectively an accuracy of 92.34% and 92%[106].

They suggested an ensemble machine learning IDS. Researchers employed the Principal Component Analysis approach for feature extraction. After several testing on the NSL-KDD datasets, their ensemble algorithm is merging Decision Tree, Random Forest, KNN, DNN and MultiTree. The outcomes of the ensemble algorithm are created by a majority vote utilizing

weights for each method to have better accuracy. They acquired an accuracy of 85.2% which is better than the accuracy if they were simply employing one method. However, when assessing attacks that are not frequent, their model is ineffective[84].

They offered a solution employing a Deep Belief Network (DBN) and an ensemble technique made of multiple SVMs. A DBN is a sequence of unsupervised networks such as Restricted Boltzmann Machines (RBM). An RBM consists of an input layer and a hidden layer where the nodes are connected to the layer above and below them but not to the layer below. The unsupervised pre-training method used by DBN is based on a greedy layer-wise structure. Then, they employ a supervised fine-tuning strategy to learn the important traits. In their solution, they use DBN for feature extraction. Then, the retrieved features are transmitted to the multi-layer ensemble SVM An algorithm that generates votes produces the result. On the datasets UNSW-NB15, KDDcup99, NSL KDD, and CICIDS2017, they evaluated their solution. They, accordingly obtained a precision of 94.76%, 97.27%, 90.47% and 90.40%. However, it was proven that when more layers are used their solution is more time intensive[107].

They increased the efficiency of DBN for IDS by employing an optimizing technique. To optimize their model, they employed a combination of Particle Swarm Optimization (PSO), Artificial Fish Swam Algorithm (AFSA), and Genetic Algorithm (GA). AFSA is used to optimize the PSO first. Then, GA is utilized to identify the global optimal solution of the first particle search. The optimal solution is then employed in the DBN model to improve its accuracy. They tested their approach on the NSL-KDD dataset and attained an accuracy of 82.36%[85].

They suggested an IDS based on Deep Neural Network (DNN). Their DNN architecture is built of an input layer, five hidden layers and an output layer. They have a scalable solution that allows the DNN models to have anywhere from one to five hidden layers. They made advantage of the computer system Apache Spark. Their solution can work in both scenarios, HIDS and NIDS. For NIDS, they tested their approach on KDDcup99, NSL-KDD, Kyoto, UNSW-NB15

and CICIDS2017 datasets. They, accordingly acquired an overall accuracy of 93%, 79.42%, 87.78%, 76.48% and 94.5% when adding the accuracy for each number of DNN layers[87].

A solution combining Random Forest and Non-symmetric Deep Auto-Encoder (NDAE) was put out by the authors. Usually, an auto-encoder uses the symmetric method from encoder-decoder, however, in their solution, they just used the encoding phase. It minimizes the processing time without influencing too much on the accuracy of the IDS. To handle complex datasets, they decided to stack their NDAE. However, they realized that employing solely NDAE was not enough to obtain an appropriate categorization. Therefore, they incorporated Random Forest as their classifier after completing feature extraction using two NDAE with three hidden layers each. Researchers put their approach to the test against a DBN solution using the KDDcup99 and NSL-KDD datasets. They obtained, respectively a total accuracy of 97.85% and 85.42%. However, their approach struggles to detect minor classes such as R2L and U2R[88].

The authors showed the influence of feature extraction using a Stacked Sparse AutoEncoder (SSAE) to improve IDS. A sparse auto-encoder is an autoencoder which utilizes a sparsity penalty, normally, the penalty is activated when hidden nodes are employed. Thus, adopting a sparse auto-encoder minimizes the amount of hidden nodes utilized. The use of several sparse auto-encoders results in stacked sparse auto-encoding. It enables for lowering the dimension of the incoming data without losing substantial information. They employed the error back propagation technique to improve their SSAE, and they used the NSLKDD dataset to evaluate their model. To demonstrate the extent to which using SSAE for feature extraction increases accuracy, they employed various classifiers both with and without their SSAE model. The best accuracy was attained when the SSAE and SVM classifiers were combined. They attained an overall accuracy of 99.35%. One of the key benefits of choosing their solution is the significant time savings for testing and training, which takes about one-tenth as long as alternative solutions. However, the detection rate for R2L and U2R is lower compared to the other classes[108].

They suggested a two-stage deep learning model (TSDL) to improve IDS. In the first stage, they classify the traffic as normal or abnormal using a likelihood value. In the second step they used this value as an additional feature to train the classifier, they employed a DNN technique for both stages, where they used a Deep stacked auto-encoder (DSAE) for feature extraction and Soft-max as a classifier. For issues requiring multi-class classification, soft-max is frequently utilized in neural networks. They tested their solution on the KDDcup99 and UNSW-NB15 datasets. They, respectively obtained a total accuracy of 99.996% and 89.134%[109].

They offered a technique combining an unsupervised approach with two auto-encoders and a supervised step to construct the datasets. Using both regular and attack traffic, they independently trained the two autoencoders. The samples are then rebuilt by the auto-encoders and added to the dataset used to train the model. The dataset runs through a one-dimension CNN. To better distinguish between the two classes—normal and attack—this is done in order to see how one channel affects the other. Finally, they utilized a Soft-max classifier to distinguish if the data was an attack or normal. On the KDDcup99, UNSW-NB15, and CICIDS2017 datasets, they evaluated their model. They respectively obtained an overall accuracy of 92.49%, 93.40% and 97.90%. One of the downsides of their system is that it does not provide details about the different types of attacks[110].

Particle swarm optimization (PSO)-based Fast Learning Network (FLN) modeling was proposed by the authors. Due to the wasteful nature of the weights utilized in the neural network, they applied PSO to increase FLN's accuracy. Researchers tested their approach on the KDDcup99 dataset against other FLN solutions. They acquired a superior accuracy to detect the different classes than the other solutions. They attained a total accuracy of 89.23%. However, their overall accuracy is hampered by their low accuracy when detecting one of the tiny classes of attack (R2L)[111].

They offered a hybrid method combining clustering with SVM. In their solution, they initially used K-means clustering to analyse the data and segregated it into different subgroups. Then, they utilized SVM on each of those subsets. They evaluated their approach on the NSL-KDD datasets and they acquired an overall accuracy of 99.45%. In addition, compared to other methodologies their solution enhanced the detection rate. Their solution also takes less time processing compared to SVM algorithms employing different parameters. However, the authors supplied no information concerning the correctness of any assault classification[112].

The authors presented a Double-Layered Hybrid Approach (DLHA). In their solution, they first generate two groups in the NSL-KDD dataset. The first one contains all classes while the second one comprises only the U2R,R2L and regular classes. They created these two groups to have better accuracy of the U2R and R2L classes which are often the weakness of most of the IDS solutions that we have seen. Then, they performed feature extraction in both groups. They first used Intersectional Correlated Feature Selection (ICFS). In ICFS, the Pearson Correlation Coefficient (PCC) is used to select important features between two random variables. PCC can determine how much two variables vary from each other. Once ICFS is done, they performed Principal Component Analysis (PCA) to reduce the dimension of the data. Finally, to have a ratio of 1:1 between attacks and normal data in the second group they randomly choose the same amount of data as R2L and U2R combined. Then, they used those two groups to train their model which is composed of a first layer using Naïve Bayes classifier and a second layer using SVM. The first layer is used only to detect DoS and Probe. If the outcome is not one of those two classes, then the data goes through the second layer to detect if it is a R2L, U2R or Normal data. They tested their solution on the NSL-KDD dataset. They obtained an overall accuracy of 93.11% and detection of 96.67% for R2L and 100% for U2R classes. Their solution outperformed other solutions when identifying the small classes, however, contrary to other efficient solutions, their accuracy for the large classes was not as good[113].

They proposed a Hybrid Nested Genetic-Fuzzy Algorithm (HNGFA) to detect attacks. They first performed feature selection using Naïve Bayes. Major features and Minor features are split into two groups. Their model is composed of two geneticfuzzy algorithms. The first one is the Outer Genetic-Fuzzy Algorithm (OGFA) and the second one is the Inner Genetic-Fuzzy Algorithm (IGFA). Each of these algorithms used two nested genetic algorithms. The outer one is used for the fuzzy sets and the inner one is used for the fuzzy rules. The OGFA is used for classifying data with major features, whereas the IGFA is used for classifying data with minor features. The two genetic-fuzzy algorithms interact with each other to develop new solutions to have better accuracy. The goal is to make the interaction between the best results of the OGFA with weak results from the IGFA to have the best model possible. They tested their solution on the KDDcup99 and UNSW-NB15 datasets and they obtained an overall accuracy of 98.19% and 80.54%, respectively. In addition, their solution got a good accuracy for detecting small classes such as R2L and U2R. However, due to the complexity of their model, the training time is high[114].

Deep Learning algorithms are a recent update to artificial neural networks that use abundant, affordable computation. Deep learning lets an algorithm to learn representation of data with various levels of generalization. These algorithms have been applied to optical object recognition, object detection, detecting network intrusion and many more domains. A deep learning system can be trained as a supervised and unsupervised approach.

Deep Learning algorithm in a supervised way: Convolutional neural network (CNN) is generally trained in a supervised fashion. CNN is presently the benchmark model for the computer vision purpose. The CNN architecture utilized to build 2D pictures and a most important use of CNN is facial recognition

Deep Learning algorithm in an unsupervised way: An autoencoder is used to learn a representation (encoding) for a set of data for the goal of dimensionality reduction. A Deep Belief

Network (DBN) can learn to recreate its inputs when trained with a set of examples in an unsupervised method. The layers then operate as feature detectors on inputs. A DBN is subsequently trained to perform categorization after this learning phase in a supervised manner. Dimensionality reduction, regression, collaborative filtering, feature learning, topic modeling, and other processes are all made possible by DBNs, such as restricted Boltzmann machines (RBMs) or auto-encoders.

Deep Learning method in a supervised or unsupervised way: Recurrent neural network (RNN) methods are considered as a supervised or unsupervised learning method. RNNs can process inputs in random order by using internal memory. RNN is frequently used for speech recognition RNN is effective at character prediction in text and also has the capacity to learn dependencies and actual evidence that is stored for a long period.

2.14 General Evaluation of Intrusion Detection Systems

This all-embracing review paper is not the same as previous survey papers in several ways. Only one or two subjects are the focus of previous works such as methodologies employed by intrusion detection and dataset which have been used. On the contrary, current IDSs various aspects were examined. In addition, each subject had several suggestions been proposed. Contribution was also made in this paper for researches and also private corporations who may desire to implement IDSs more successfully.

In the study, all IDSs procedures, technologies and methods were in detailed examined. Each technique employed has greater advantage to complement each other in many aspects for example, speed, accuracy of attack detection, alarming method and method of collecting data. In spite of the fact that numerous developed and underdeveloped IDS are available, they seem to still be inclined to untrue positives or false alarms. There is a need for proper configuration of IDSs so as to distinguish potential activity that are malicious to normal network traffic. However,

despite the inefficiencies cause by this, significant damages to the network are not committed by false positive. The graver IDS blunder is false negative. This is due to the failure of the IDS in detecting a danger and passing it as normal traffic. In an untrue negative situation, attack that took place is not flagged, alarm generation non-existence and only after the network is compromised in some way would the attack be defined.

2.15 Datasets

To train and test their models, the researchers used datasets. Following is a discussion of the most known and used datasets for Intrusion Detection System training and testing.

2.16 KDDcup99

The KDDcup99 dataset has been one of the most widely used datasets to assess IDS. It is based on the DARPA'98 dataset. The KDDcup99 contains approximately 4,900,000 samples. Each sample has 41 features and is labelled as Normal or Attack. The attack samples are classified into four categories:

i. Denial of Service (DoS),
ii. User to Root (U2R),
iii. Remote to Local (R2L), and
iv. Probe.

There are three different datasets for KDDcup99, the first one is the whole dataset, the second corresponds to 10% of the whole dataset the third one is a test dataset which contains 311,029 samples. One of the main disadvantages of this dataset is that it is imbalanced, i.e., many samples

are like each other for major classes such as DoS and Probe whereas for R2L and U2R there are few. Depending on which part of the dataset is used some classes might be completely absent[116].

2.17 Kyoto 2006

This dataset was created by deploying honeypots, darknet sensors, email servers, web crawlers, and other network security measures outside and inside Kyoto University to collect various types of traffic. Based on the 41 features from the KDDcup99 dataset, they extracted 14 statistical features. In addition, they also extracted 10 additional features to form the dataset, thus, each sample has 24 features. The most recent version of the Kyoto dataset includes traffic from 2006 to 2015.

2.18 NSL-KDD

This dataset was created to fix the main issue of the KDDcup99 dataset. It was proposed in 2009 by the authors[117]. It keeps the four attack categories of the KDDcup99. The NSL-KDD proposes two files, a training set, and a testing set. The training set is made of 21 different attacks and has 126,620 instances. The testing set is made of 37 different attacks and has 22,850 instances[27].

2.19 UNSW-NB15

This dataset was created by the Australian Centre for Cyber Security. It was created to generate traffic which is a hybrid of normal activities and attack behaviours. This dataset has nine types of attacks:

i. Fuzzers,

ii. Analysis,

iii. Backdoors,

iv. DoS,

v. Exploits,

vi. Generic,

vii. Reconnaissance,

viii. Shellcode and

ix. Worms.

The UNSW proposes two files, a training set, and a testing set. These files contain records from different types of traffic, attacks and normal, from the original dataset. The original dataset has a number of records of 2,540,044 while, the training set has 175,341 records, and the testing set has 82,332 records.

2.20 CICIDS2017

This dataset was created by the Canadian Institute for Cybersecurity (CIC) in 2017. This dataset was built using real-world traffic containing both normal and recent attack samples. The results were analyzed based on the time stamp, source, and destination IP, protocols, and attacks using CICFlowMeter. In addition, they implemented common attacks such as Brute Force FTP, Brute Force SSH, Denial of Service (DoS), HeartBleed, Web Attack, Infiltration, Botnet and Distributed Denial of Service (DDoS)[116].

2.21 Summary of Gaps in Literature Reviewed

In conclusion, hypersensitive to abnormal activities and production of false positive is better than to be insensitive and produce false negatives. With the evolution of attacks and complexity it

possesses, IDSs are battling with big problem of false negative. Known attacks can be detected quickly and more accurately using signature-based IDSs. However, in the case of strange threats, anomaly-based detection systems, which are capable of identifying such threats are not sufficient. As a result, the need for IDSs to identify new behaviour, dynamically detect novel attacks and evading methods as soon as possible. In this setting, some recommendations are itemized below:

i. Hiding techniques are some of the ways new generation of attacks are using. In order to identify these attacks both quickly and more accurately, hybrid of signature-based and anomaly-based approach hybrid system can be established.
ii. Network observation in real-time and attack detection is a difficulty method. Majority of the researches conducted so far make use of dataset for detecting studies and not applicable for real-time monitoring. Development of real-time attack detection in a system would greatly contribute to this field.
iii. FPs and FNs are the major attacks most intrusion detections are prone to. In addition to achieving high precision frequency in this novel study, research should focus more on how to reduce FPs and FNs.
iv. The deficiency of well-known, high-volume and most importantly updated dataset which can be utilized to assess the outcome of Intrusion Detection System.

Novel attacks types contained in a new dataset that would fill the gap which could be developed would be a vital pace in this area.

Endnotes

1. M. Ozkan-Okay, R. Samet, Ö. Aslan & D. Gupta, *A Comprehensive Systematic Literature Review on Intrusion Detection Systems*, **IEEE**, vol. 9, pp. 157727-157760, 2021, 3-4, doi: 10.1109.

2. P. Bedi, N. Gupta, & V. Jindal, ‘‘*Siam-IDS: Handling Class Imbalance Problem in Intrusion Detection Systems using Siamese Neural Network*, **Procedia Computer Science**, Vol 171, 2020, pp 780-789.

3. S. Prasad, S. Alamuru, S. Arun & S. Alamuru, *Intrusion Detection and Prevention Systems in Smart Environments – A Survey*, **2022 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON),** 2022, pp. 1-5, doi: 10.1109/SMARTGENCON56628.2022.10084309.

4. J. C. S. Sicato, S. K. Singh, S. Rathore & J. H Park, *A Comprehensive Analysis of Intrusion Detection System for IoT Environment.* **Journal of Information Processing Systems, 2020**, vol. 16, no. 4, 2020, pp. 975-990.

5. D. A. Effendy, K. Kusrini & S. Sudarmawan, *Classification of Intrusion Detection System (IDS) Based on Computer Network*. **2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE),** 2017, pp. 90-94, doi: 10.1109/ICITISEE.2017.8285566.

6. M. Pradhan, C. K. Nayak & S. K. Pradhan, *Intrusion Detection System (IDS) and Their Types in Securing the Internet of Things: Concepts, Methodologies, Tools, and Applications* (**IGI Global),** 2020, 481-497, https://doi.org/10.4018/978-1-5225-9866-4.ch026.

7. R Ganeshan, C.S. kolli, M. kumar & T. Daniya, *A Systematic Review on Anomaly Based Intrusion Detection System*, **IOP Conference Series: Materials Science and Engineering**, vol. 981, no. 2, 2020, 10.1088/1757-899X/981/2/022010.

8. H. Alqahtani, I. H. Sarker, A. Kalim, S. M. H. Minhaz, S. Ikhlaq, & S. Hossain, *Cyber Intrusion Detection using Machine Learning Classification Techniques*, **In Computing Science, Communication and Security: First International Conference, COMS2 2020***,* 2020*,* pp. 121-131.

9. S. E, Quincozes, C. Albuquerque, D. Passos, & D. Mossé, *A Survey on Intrusion Detection and Prevention Systems in Digital Substations*, **Computer Networks**, 184, 2021, https://doi.org/10.1016/j.comnet.2020.107679.

10. C. Tuan-Hong & S. Iftekhar, *Evaluation of Machine Learning Algorithms in Network-Based Intrusion Detection Using Progressive Dataset*, **Symmetry**, Vol. 15, 2023, pp. 1251, doi: 10.3390/sym15061251.

11. P. Hadem, D. K. Saikia & S. Moulik, *An SDN-based Intrusion Detection System Using SVM with Selective Logging for IP Traceback,* **Computer Networks**, vol. 191, 2021, Art. no. 108015. https://doi.org/10.1016/j.comnet.2021.108015.

12. S. Kumar, S. Gupta & S. Arora, *Research Trends in Network-Based Intrusion Detection Systems: A Review*, **IEEE Access**, vol. 9, 2021, pp. 157761-157779, doi: 10.1109/ACCESS.2021.3129775.

13. A. Khraisat & A. Alazab, *A Critical Review of Intrusion Detection Systems in the Internet of Things: Techniques, Deployment Strategy, Validation Strategy, Attacks, Public Datasets and Challenges*, **Cybersecurity**, vol. 4, no. 1, 2021, doi: 10.1186/s42400-021-00077-7.

14. A. Omar, *A Feature Selection Model for Network Intrusion Detection System Based on PSO, GWO, FFA and GA Algorithms,* ***Symmetry***, vol. 12, no. 6, 2020, https://doi.org/10.3390/sym12061046.

15. A. Smith, T. D. Ramotsoela & G. P. Hancke, *Behavioural Intrusion Detection for Wireless Sensor Networks*, **2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)**, 2021, pp. 01-06, doi: 10.1109/ISIE45552.2021.9576349.

16. P. Horchulhack, E. K. Viegas & A. O. Santin, *Toward Feasible Machine Learning Model Updates in Network-based Intrusion Detection*, **Computer Networks**, vol. 202, 2022, pp. 1389-1286, , https://doi.org/10.1016/j.comnet.2021.108618.

17. I. F. Kilincer, F. Ertam & A. Sengur, *Machine Learning Methods for Cyber Security Intrusion Detection: Datasets and Comparative Study*, **Computer Networks**, Vol. 188, ISSN 1389-1286, 2021, https://doi.org/10.1016/j.comnet.2021.107840.

18. M. Mazini, B. Shirazi & I. Mahdavi, *Anomaly Network-based Intrusion Detection System Using a Reliable Hybrid Artificial Bee Colony and AdaBoost Algorithms*, **Journal of King Saud University-Computer and Information Sciences**, vol. 31, no. 4, 2019, pp. 541-553.

19. S. Meftah, T. Rachidi & N. Assem, *Network Based Intrusion Detection Using the UNSW-NB15 Dataset,* **International Journal of Computing and Digital Systems**, vol. 8, no. 5, 2019, pp. 478-487.

20. P. Bedi, N. Gupta & V. Jindal, *I-SiamIDS: An Improved Siam-IDS for Handling Class Imbalance in Network-Based Intrusion Detection Systems* **Applied Intelligence**, vol. 51, no. 2, 2021, pp. 1133-1151.

21. J. Long, F. Fang & H. Luo, *A Survey of Machine Learning-based IoT Intrusion Detection Techniques*, **IEEE 6th International Conference on Smart Cloud (SmartCloud)**, 2021, pp. 7-12, doi: 10.1109/SmartCloud52277.2021.00009.

22. S. Mani, B. Sundan, A. Thangasamy & L. Govindaraj, *A New Intrusion Detection and Prevention System Using a Hybrid Deep Neural Network in Cloud Environment*. **Computer Networks, Big Data and IoT. Lecture Notes on Data Engineering and Communications Technologies,** vol 117, 2022, https://doi.org/10.1007/978-981-19-0898-9_73.

23. S. Jin, R. Diao, & Q. Shen, *Backward Fuzzy Interpolation and Extrapolation With Multiple Multi-Antecedent Rules*. **In Proceedings of IEEE International Conference on Fuzzy Systems**, 2012, pp. 1170–1177.

24. A. A. Khalil & M. A. Rahman, *Adaptive Neuro-Fuzzy Inference System-based Lightweight Intrusion Detection System for UAVs*," **2023 IEEE 48th Conference on Local Computer Networks (LCN),** 2023, pp. 1-9, doi: 10.1109/LCN58197.2023.10223340.

25. A. Beachy, H. Bae, J. A. Camberos, R. V. *Grandhi, Epistemic Modeling Uncertainty of Rapid Neural Network Ensembles for Adaptive Learning*, **Finite Elements in Analysis and Design**, vol. 228, 2023, https://doi.org/10.1016/j.finel.2023.104064.

26. O. Alaa & A. Mohammed, *An Efficient Approach towards Network Routing using Genetic Algorithm*, **International Journal of Computers Communications and Control**, vol. 17, 2022, https://doi.org/10.15837/ijccc.2022.5.4815.

27. Y. Yuhua , J. Jang-Jaccard, X. Wen, S. Amardeep, Z. Jinting, S. Fariza & K. Jin, *IGRF-RFE: A Hybrid Feature Selection Method for MLP-based Network Intrusion Detection on UNSW-NB15 Dataset*. **Journal of Big Data,** vol. 10, 2023, https://doi.org/10.1186/s40537-023-00694-8

28. Y. Chen, Q. Lin, W. Wei, J. Ji, W. Ka-Chun & A. C. Carlos, *Intrusion Detection Using Multi-Objective Evolutionary Convolutional Neural Network for Internet of Things in Fog Computing*, **Knowledge-Based Systems**, vol. 244, 2022, pp. 108505, https://doi.org/10.1016/j.knosys.2022.108505.

29. N. Kesswani, & B. Agarwal, *SmartGuard: An IoT-based Intrusion Detection System for Smart Homes.* **International Journal of Intelligent Information and Database Systems**, vol. 13(1), 2020, pp. 61–71.

30. S. Muhammad, G.Zhaoquan, C. Omar , A. Wajdi, & H. Habib, *The Rise of "Internet of Things": Review and Open Research Issues Related to Detection and Prevention of IoT-Based Security Attacks*, **Hindawi Wireless Communications and Mobile Computing**, vol. 2022, 2022, pp. 12, https://doi.org/10.1155/2022/8669348.

31. H. M. Song, J. Woo, & H. K. Kim, *In-vehicle Network Intrusion Detection Using Deep Convolutional Neural Network*. **Vehicular Communications**, vol. 21, 2020, 100198.

32. Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, (2020). *Robust Detection for Network Intrusion of Industrial IoT Based on Multi-CNN Fusion.* **Measurement**, vol. 154, 2020, 107450.

33. W.Elmasry, A. Akbulut, & A. H. Zaim, *Evolving Deep Learning Architectures for Network Intrusion Detection Using a Double PSO Metaheuristic*. **Computer Networks**, vol. 168, 2020, 107042.

34. M. Doaa & I. Osama, *Enhancement of an IoT Hybrid Intrusion Detection System Based on Fog-to-cloud Computing*, **Journal of Cloud Computing**, vol. 12, no. 1, 2023, 41, https://doi.org/10.1186/s13677-023-00420-y.

35. I. Martins, J. S. Resende, P. R. Sousa, S. Silva, L. Antunes, & J. Gama, *Host-based IDS: A Review and Open Issues of an Anomaly Detection System in IoT*, **Future Generation Computer Systems,** vol. 133, no. 95, 2022, 0167-739X, https://doi.org/10.1016/j.future.2022.03.001

36. Y. Shin & K. Kim, *Comparison of Anomaly Detection Accuracy of Host-based Intrusion Detection Systems based on Different Machine Learning Algorithms*, **International Journal of Advanced Computer Science and Applications**, vol. 11, no. 2, 2022, doi: 10.14569/IJACSA.2020.0110233.

37. A. Giuseppina, A. Annalisa, C. F. Paolo, M. Donato, & V. Gennaro, *ROULETTE: A neural attention multi-output model for explainable Network Intrusion Detection*, **Expert Systems with Applications**, vol. 201, no. 117144, 2022, https://doi.org/10.1016/j.eswa.2022.117144.

38. K. Chaouki, & K. Saoussen, *A NSGA2-LR Wrapper Approach for Feature Selection in Network Intrusion Detection*, **Computer Networks**, vol. 172, 2020, doi: https://doi.org/10.1016/j.comnet.2020.107183.

39. C. Khammassi & S. Krichen, *A NSGA2-LR Wrapper Approach for Feature Selection in Network Intrusion Detection*, **Computer Networks,** vol. 172, pp. 1389-1286, 2020, https://doi.org/10.1016/j.comnet.2020.107183.

40. J. Byrnes, T. Hoang, N. N. Mehta & Y. Cheng, *A Modern Implementation of System Call Sequence Based Host-based Intrusion Detection Systems*, in **Proc. Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)** (TPS-ISA), 2020, pp. 218-225.

41. D. Park, S. Kim, H. Kwon, D. Shin & D. Shin, *Host-based Intrusion Detection Model Using Siamese Network*, **IEEE Access**, vol. 9, 2021, pp. 76614-76623.

42. M.Alkasassbeh & B. S. Al-Haj, *Intrusion Detection Systems: A State-of-the-Art Taxonomy and Survey*. **Arabian Journal for Science and Engineering,** vol. 48, 2023, pp.10021–10064 https://doi.org/10.1007/s13369-022-07412-1.

43. L. Ouarda, B. Malika, & B. Brahim, (2023). *Towards a Better Similarity Algorithm for Host-based Intrusion Detection System*. **Journal of Intelligent Systems**, vol. *32*, 2023, pp. 20220259. https://doi.org/10.1515/jisys-2022-0259.

44. P. Stavroulakis, & M. Stamp, *Handbook of Information and Communication Security.* **New York: Springer**. 2010.

45. L. Zheng, J. Zhang, F. Lin & X. Wang, *Feature-Fusion-Based Abnormal-Behavior-Detection Method in Virtualization Environment.* **Electronics**, vol. 12, 2023, 3386. https://doi.org/10.3390/electronics12163386.

46. L.Chen, S. Gao, & B. Liu, *An Improved Density Peaks Clustering Algorithm Based on Grid Screening and Mutual Neighborhood Degree for Network Anomaly Detection*. **Scientist Report**, vol. 12, 2022, 1409. https://doi.org/10.1038/s41598-021-02038-z.

47. W. Qiu, Y. Ma, X. Chen, H. Yu & L. Chen, *Hybrid Intrusion Detection System Based on Dempster-Shafer Evidence Theory*, **Computers & Security**, vol. 117, 2022, https://doi.org/10.1016/j.cose.2022.102709.

48. F. Chuan, H. Pengchao, Z. Xu, Y. Bowen & L. G. Yejun, *Computation Offloading in Mobile Edge Computing Networks: A Survey*, **Journal of Network and Computer Applications**, vol. 202, 2022, doi.org/10.1016/j.jnca.2022.103366.

49. H. W. Li, Y. S. Wu, & Y. Huang, *On the Feasibility of Anomaly Detection with Fine-Grained Program Tracing Events*. **Journal of Network and Systems Management**, vol. 30, 2022, pp.28,. https://doi.org/10.1007/s10922-021-09635-3.

50. A. Vinolia, N. Kanya & V. N. Rajavarman, *Machine Learning and Deep Learning-based Intrusion Detection in Cloud Environment: A Review*, **2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)**, 2023, pp. 952-960, doi: 10.1109/ICSSIT55814.2023.10060868.

51. L. Suman & S. Dheerendra, *Intrusion Detection System in Cloud Environment: Literature Survey & Future Research Directions*, **International Journal of Information Management Data Insights**, vol. 2, 2022, https://doi.org/10.1016/j.jjimei.2022.100134.

52. W. N. H. Ibrahim, S. Anuar, A. Selamat, O. Krejcar, R. G. Crespo, & H. Fujita, *Multilayer Framework for Botnet Detection Using Machine Learning Algorithms*, **IEEE**, vol. 9, 2021, pp. 48753-48768, doi: 10.1109/ACCESS.2021.3060778.

53. M. M. Alani & A. I. Awad, *An Intelligent Two-Layer Intrusion Detection System for the Internet of Things*, **IEEE Transactions on Industrial Informatics,** vol. 19, 2023, pp. 683-692, doi: 10.1109/TII.2022.3192035.

54. W. A. H. M. Ghanem, *Cyber Intrusion Detection System Based on a Multiobjective Binary Bat Algorithm for Feature Selection and Enhanced Bat Algorithm for Parameter Optimization in Neural Networks*, in **IEEE Access,** vol. 10, 2022, pp. 76318-76339, doi: 10.1109/ACCESS.2022.3192472.

55. R.A. Disha, & S. Waheed, *Performance Analysis of Machine Learning Models for Intrusion Detection System Using Gini Impurity-based Weighted Random Forest (GIWRF) Feature Selection Technique*, **Cybersecurity**, vol. 5, no, 1, 2022, https://doi.org/10.1186/s42400-021-00103-8.

56. A. Thakkar, & R. Lohiya, *Attack Classification Using Feature Selection Techniques: a Comparative Study*, **Journal of Ambient Intelligence and Humanized Computing**, vol. 12, 2021, pp. 1249–1266, https://doi.org/10.1007/s12652-020-02167-9.

57. A. N. Calugar, W. Meng & H. Zhang, *Towards Artificial Neural Network Based Intrusion Detection with Enhanced Hyperparameter Tuning,* **GLOBECOM 2022 - 2022 IEEE Global Communications Conference,** 2022, pp. 2627-2632, doi: 10.1109/GLOBECOM48099.2022.10000809.

58. T. Monis, & S. Mohd, *A Review on Intrusion Detection in Cloud Computing*. **International Journal of Engineering and Management Research**, vol. *13*(2), 2023, 207–215. https://doi.org/10.31033/ijemr.13.2.35l.

59. S. Ahn, H. Yi, Y. Lee, W. R. Ha, G. Kim & Y. Paek, *Hawkware: Network Intrusion Detection Based on Behavior Analysis With ANNs on an IoT Device*, in *Proc.* **57th ACM/IEEE Design Automation Conference (DAC)**, 2020, pp. 1-6.

60. S. Yang, *Research on Network Malicious Behavior Analysis Based on Deep Learning*, in *Proc.* **IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)**, 2021, pp. 2609-2612.

61. S. Waskle, L. Parashar & U. Singh, *Intrusion Detection System Using PCA with Random Forest Approach*, **2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)**, 2020, pp. 803-808, doi: 10.1109/ICESC48915.2020.9155656.

62. S. Sharma, P. Nand, & P. Sharma, *Intrusion Detection and Prevention Systems Using SNORT*, **Advances in Data Science and Management**, vol 86, 2022, doi: org/10.1007/978-981-16-5685-9_46.

63. M. Boopathi & R. Seetha, *Accurate Detection of Multi-layer Packet Dropping Attacks Using Distributed Mobile Agents in MANET*, **Journal of Physics: Conference Series**, vol. 1979, no. 1, 2021, pp. 1742-6596, doi: 10.1088/1742-6596/1979/1/012040.

64. S. Maesaroh, L. Kusumaningrum, N. Sintawana, , P. Lazirkha, D., & Regina Dinda.O., *Wireless Network Security Design And Analysis Using Wireless Intrusion Detection System*, **International Journal of Cyber and IT Service Management**, vol. *2,* no. 1, 2022, pp.30–39. https://doi.org/10.34306/ijcitsm.v2i1.74.

65. S.M. Mousavi, V. Majidnezhad, & A. Naghipour, *A New Intelligent Intrusion Detector Based on Ensemble of Decision Trees,* **Journal of Ambient Intelligence and Humanized Computing**, vol. 13, 2022, pp. 3347–3359, https://doi.org/10.1007/s12652-019-01596-5..

66. U. Dixit, S. Bhatia & P. Bhatia, *Algorithms for Alert Classification in Intrusion Detection and Prevention Systems: A Detailed Review*, **2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE),** 2022, pp. 1199-1205, doi: 10.1109/ICACITE53722.2022.9823605.

67. U. Dixit, S. Bhatia & P. Bhatia, *Utilizing ML and DL Algorithms for Alert Classification in Intrusion Detection and Prevention Systems: A Detailed Review*, **2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)**, 2022, pp. 1199-1205, doi: 10.1109/ICACITE53722.2022.9823605.

68. V.Sundararajan, & E. Dietz, *Centralized Hierarchical Cybersecurity Monitoring Towards Securing the Defense Industrial Base Supply Chain*. **Series Of Research Network**, 2023, p. 8, http://dx.doi.org/10.2139/ssrn.4603578.

69. M. Z. Gunduz & R. Das, *Cyber-security on Smart Grid: Threats and Potential Solutions*, **Computer Network**, vol. 169, 2020, https://doi.org/10.1016/j.comnet.2019.107094.

70. Z. S. Malek, B. Trivedi & A. Shah, *User Behavior Pattern –Signature Based Intrusion Detection*, *Proc.* **Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)**, 2020, pp. 549-552, doi: 10.1109/WorldS450073.2020.9210368.

71. Y. Otoum & A. Nayak, *AS-IDS: Anomaly and Signature Based IDS for the Internet of Things*, **Journal of Network and Systems Management**, vol. 29, no. 23, 2021, pp. 1-26, doi: https://doi.org/10.1007/s10922-021-09589-6.

72. Z. Yang, X. Liu, T. Li, D.Wu, J. Wang, Y. Zhao & H. Han, *A Systematic Literature Review of Methods and Datasets for Anomaly-based Network Intrusion Detection*, **Computers & Security,** vol. 116, no. 102675, 2022, pp. 0167-4048, doi: https://doi.org/10.1016/j.cose.2022.102675.

73. S. K. Gupta, M. Tripathi & J. Grover, *Hybrid Optimization and Deep Learning Based Intrusion Detection System*, **Computers and Electrical Engineering**, vol. 100, no. 107876, 2022, pp. 0045 – 7906, doi: https://doi.org/10.1016/j.compeleceng.2022.107876..

74. M. Leon, T. Markovic & S. Punnekkat, *Comparative Evaluation of Machine Learning Algorithms for Network Intrusion Detection and Attack Classification*, **2022 International Joint Conference on Neural Networks (IJCNN)**, 2022, doi: 10.1109/IJCNN55064.2022.9892293.

75. S.K. Prashanth, S. Shitharth, B. K. Praveen, V. Subedha & K. Sangeetha, *Optimal Feature Selection Based on Evolutionary Algorithm for Intrusion Detection*. **SN Computer Science**. Vol. 3, no. 439, 2022, https://doi.org/10.1007/s42979-022-01325-4.

76. P. Kanimozhi & V. T. Aruldoss, *Oppositional Tunicate Fuzzy C-means Algorithm and Logistic Regression for Intrusion Detection on Cloud*, **Concurrency and Computation Practice and Experience**, vol. 34, no. 4, 2022, e6624. doi:10.1002/cpe.6624.

77. D. N Mhawi, A. Aldallal & S. Hassan, *Advanced Feature-Selection-Based Hybrid Ensemble Learning Algorithms for Network Intrusion Detection Systems*. **Symmetry**, vol. 14, 2022, 1461. https://doi.org/10.3390/sym14071461.

78. S. Dwivedi, M. Vardhan, S. Tripathi & A. K. Shukla, *Implementation of Adaptive Scheme in Evolutionary Technique for Anomaly-based Intrusion Detection*, **Evolutionary Intelligence**, vol. 13, no. 1, 2020, pp. 103-117. https://doi.org/10.1007/s12065-019-00293-8

79. S. Geeta & K. Neelu, *A Survey of Intrusion Detection From the Perspective of Intrusion Datasets and Machine Learning Techniques,* **International Journal of Computers and Applications**, vol. 44, 2022, 659-669, DOI: 10.1080/1206212X.2021.1885150.

80. S. Choudhar & N. Kesswani, *A Hybrid Classification Approach for Intrusion Detection in IoT Network*, **Journal of Scientific & Industrial Research**, vol. 80, no. 9, 2021, pp. 809-816, https://doi.org/10.1155/2022/4553502.

81. D.N. Mhawi, A Aldallal, S. Hassan, *Advanced Feature-Selection-Based Hybrid Ensemble Learning Algorithms for Network Intrusion Detection Systems*. **Symmetry**. Vol. 14(7), 2022. 1461. https://doi.org/10.3390/sym14071461.

82. V. Kumar, D. Sinha, A. K. Das, C. P. Subhash & T. G. Radha, *An Integrated Rule Based Intrusion Detection System: Analysis on UNSW-NB15 Data Set and the Real Time Online*

Dataset, **Cluster Computing**, vol. 23, 2020, pp 1397–1418, https://doi.org/10.1007/s10586-019-03008-x.

83. A. Heidari, & J. M.A. Jabraeil, *Internet of Things Intrusion Detection Systems: A Comprehensive Review and Future Directions*. **Cluster Computing,** vol. 26, 2023, 3753–3780. https://doi.org/10.1007/s10586-022-03776-z.

84. Y. Zhongjun, L. Zhi , Z. Xuejun, & W. Guogang, *An Optimized Adaptive Ensemble Model with Feature Selection for Network Intrusion Detection*, **Concurrency and Computation: Practice and Experience**, vol. 35, 2023, https://doi.org/10.1002/cpe.7529.

85. V. Parganiha, S. P. Shukla, & L. K. Sharma, *Cloud Intrusion Detection Model Based on Deep Belief Network and Grasshopper Optimization*. **International Journal of Ambient Computing and Intelligence (IJACI)**, vol. 13(1), 2022, pp. 1-24. http://doi.org/10.4018/IJACI.293123

86. H. Alavizadeh, H. Alavizadeh & J. Jang-Jaccard, *Deep Q-Learning Based Reinforcement Learning Approach for Network Intrusion Detection*. **Computers,** vol. 11, 2022, pp. 41. https://doi.org/10.3390/computers11030041.

87. A. Sharon, P. Mohanraj, T. E. Abraham, B. Sundan & A. Thangasamy, *An Intelligent Intrusion Detection System Using Hybrid Deep Learning Approaches in Cloud Environment*. **Computer, Communication, and Signal Processing. (ICCCSP)**, vol. 651, 2022. https://doi.org/10.1007/978-3-031-11633-9_20.

88. T. Zhongyun, H. Haiyang & X. Chonghuan, *A Federated Learning Method for Network Intrusion Detection*, **Concurrency and Computation: Practice and Experience**, vol. 24, 2021, https://doi.org/10.1002/cpe.6812.

89. K. A. A. Abdulrahman & I. Muhammad, *Distributed Denial of Service Attack Alleviated and Detected by Using Mininet and Software Defined Network*, **Webology**, vol. 19, 2022, pp. 1 – 16, DOI: 10.14704/WEB/V19I1/WEB19272.

90. N.M. Raja, & S. Vegad, *An Empirical Study for the Traffic Flow Rate Prediction-based Anomaly Detection in Software-defined Networking: A Challenging Overview.* **Social Network Analysis and Mining**, vol. 13, 2023, https://doi.org/10.1007/s13278-023-01057-0.

91. S. Muzafar & N. Jhanjhi, *DDoS Attacks on Software Defined Network: Challenges and Issues,* **International Conference on Business Analytics for Technology and Security (ICBATS)**, 2022, pp. 1-6, doi: 10.1109/ICBATS54253.2022.9780662.

92. Q. Emad-ul- Haq, I. Muhammad, H. Noman, S. Muhammad & R. Imran, *An Intelligent and Efficient Network Intrusion Detection System Using Deep Learning*, **Computers and Electrical Engineering**, Vol. 99, 2022, https://doi.org/10.1016/j.compeleceng.2022.107764.

93. A. H. Janabi, T. Kanakis & M. Johnson, *Convolutional Neural Network Based Algorithm for Early Warning Proactive System Security in Software Defined Networks*, ***in*** **IEEE Access**, vol. 10, 2022, pp. 14301-14310, doi: 10.1109/ACCESS.2022.3148134.

94. A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi & R. Ahmad, *Machine Learning and Deep Learning Approaches for CyberSecurity: A Review*, **IEEE Access**, vol. 10, 2022, pp. 19572-19585, doi: 10.1109/ACCESS.2022.3151248.

95. A. Ibrahim, D. Abdelghani, A. C. Samia, Mohammed, A. A. Al-qaness, & A. E. Mohamed, *Feature Selection Model Based on Gorilla Troops Optimizer for Intrusion Detection Systems*. **Journal of Sensors**, vol. 2022, 2022, pp. 1-12, https://doi.org/10.1155/2022/6131463.

96. L. Chen, X. Kuang, A. Xu, S. Suo, & Y. Yang, *A Novel Network Intrusion Detection System Based on CNN.* **In Proceedings of the 2020 Eighth International Conference on Advanced Cloud and Big Data (CBD), Taiyuan, China,** 2020, pp. 243–247. https://doi.org/10.1109/CBD51900.2020.00051.

97. S. Gautam, A. Henry, M. Zuhair, M. Rashid, A. R. Javed & P. K. R. Maddikunta, *A Composite Approach of Intrusion Detection Systems: Hybrid RNN and Correlation-Based Feature Optimization*, **Electronics**, vol. 11, 2022, 3529. https://doi.org/10.3390/electronics11213529.

98. M. Arunkumar & K.A. Kumar, *GOSVM: Gannet Optimization Based Support Vector Machine for Malicious Attack Detection in Cloud Environment*. **International Journal of Information Technology**, vol. 15, 2023, pp. 1653–1660. https://doi.org/10.1007/s41870-023-01192-z.

99. P. Kanimozhi & T. A. A Victoire, *Oppositional Tunicate Fuzzy C-means Algorithm and Logistic Regression for Intrusion Detection on Cloud*. **Concurrency and Computation Practice and Experience.**, vol. 34, 2022, e6624. https://doi.org/10.1002/cpe.6624.

100. Y. Chen, & F. Yuan, *Dynamic Detection of Malicious Intrusion in Wireless Network Based on Improved Random Forest Algorithm.* **In Proceedings of the 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC),** 2022 pp. 27-32. https://doi.org/10.1109/IPEC54454.2022.9777557.

101. A. R. Yeruva, P. Chaturvedi, A. L. N. Rao, S. C. DimriL, C. Shekar & B. Yirga, *Anomaly Detection System using ML Classification Algorithm for Network Security*, **2022 5th International Conference on Contemporary Computing and Informatics (IC3I)**, Uttar Pradesh, India, 2022, pp. 1416-1422, doi: 10.1109/IC3I56241.2022.10072303.

102. J. Gu, & S. Lu, *An Effective Intrusion Detection Approach Using SVM with Naïve Bayes Feature Embedding.* **Computers & Security.** Vol. 103, 2021, 102158. https://doi.org/10.1016/j.cose.2020.102158.

103. J. S. Pan, F. Fan, S.C. Chu, H. Zhao & G. A. Liu, *Lightweight Intelligent Intrusion Detection Model for Wireless Sensor Networks*. **Security and Communication Networks**, vol. 2021, 2021, pp. 1–15. https://doi.org/10.1155/2021/5540895.

104. J. Wintrode & D. DeTienne, *Adaptive Encrypted Traffic Characterization via Deep Representation Learning*, **2022 Intermountain Engineering, Technology and Computing** (**IETC**), 2022, pp. 1-6, doi: 10.1109/IETC54973.2022.9796734.

105. X. Zhang, J. Chen, Y. Zhou, L. Han & J. Lin, *A Multiple-Layer Representation Learning Model for Network-Based Attack Detection*. **IEEE Access,** vol. 7, 2019, pp. 91992–92008. https://doi.org/10.1109/ACCESS.2019.2927465.

106. Y. Yu, & N. Bian, *An Intrusion Detection Method Using Few-Shot Learning*. **IEEE Access**, vol. 8, 2020, pp. 49730–49740. https://doi.org/10.1109/ACCESS.2020.2980136.

107. N. Marir, H. Wang, G. Feng, B. Li & M. Jia, Distributed Abnormal Behavior Detection Approach Based on Deep Belief Network and Ensemble SVM Using Spark**. IEEE Access**, vol. 6, 2018, pp. 59657–59671. https://doi.org/10.1109/ACCESS.2018.2875045.

108. B. Yan, & G. Han, *Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System*. **IEEE Access.** Vol. 6, 2018, pp. 41238–41248. https://doi.org/10.1109/ACCESS.2018.2858277.

109. F. A. Khan, A. Gumaei, A. Derhab, & A. Hussain, *A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection.* **IEEE Access**, vol. 7, 2019, pp. 30373–30385. https://doi.org/10.1109/ACCESS.2019.2899721.

110. G. Andresini, A. Appice, N. D., Mauro, C. Loglisci & D. Malerba, *Multi-Channel Deep Feature Learning for Intrusion Detection*. **IEEE Access**, vol. 8, 2020, pp. 53346–53359. https://doi.org/10.1109/ACCESS.2020.2980937.

[111] M.H. Ali, B. A. D. A. Mohammed, A. Ismail & M. F. Zolkipli, *A New Intrusion Detection System Based on Fast Learning Network and Particle Swarm Optimization.* **IEEE Access,** vol. 6, 2018, pp. 20255–20261. https://doi.org/10.1109/ACCESS.2018.2820092.

[112] D. Liang, Q. Liu, B. Zhao, Z. Zhu & D. Liu, *A Clustering-SVM Ensemble Method for Intrusion Detection System.* **In Proceedings of the 2019 8th International Symposium on Next Generation Electronics (ISNE),** 2019, pp. 1–3. https://doi.org/10.1109/ISNE.2019.8896514.

[113] T. Wisanwanichthan, & M. Thammawichai, *A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM*. **IEEE Access** vol. PP, 2021, pp. 1 – 1, https://doi.org/10.1109/ACCESS.2021.3118573.

[114] R. Elhefnawy, H. Abounaser, & A. Badr, *A Hybrid Nested Genetic-Fuzzy Algorithm Framework for Intrusion Detection and Attacks.* **IEEE Access**, vol. 8, 2020, pp. 98218–98233. https://doi.org/10.1109/ACCESS.2020.2996226.

[115] T. Bakshi, *State of the Art and Recent Research Advances in Software Defined Networking*. **In Wireless Communications and Mobile Computing**, vol. 2017, 2017, https://doi.org/10.1155/2017/7191647.

[116] D. Protic, *Review of KDD Cup '99, NSL-KDD and Kyoto 2006 and datasets.* **Vojnoteh. Glas**. vol. 66, 2018, pp. 580–596. https://doi.org/10.5937/vojtehg66-16670.

[117] M. Tavallaee, E. Bagheri, W. Lu & A. A. Ghorbani, *A Detailed Analysis of the KDD CUP 99 Data Set**, 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications***, 2009, pp. 1-6, doi: 10.1109/CISDA.2009.5356528.

[118] O. Aslan, M. Ozkan-Okay & D. Gupta, *Intelligent Behavior-based Malware Detection System on Cloud Computing Environment*, **IEEE**, vol. 9, 2021, pp. 83252-83271.

[119] D. Gupta, S. Bhatt, M. Gupta, O. Kayode & A. S. Tosun, Access Control Model for Google Cloud IoT, ***Proc.* IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)**, 2020, pp. 198-208.

[120] S. Garg, K. Kaur, N. Kumar, G. Kaddoum, A. Y. Zomaya, & R. Ranjan, *A Hybrid Deep Learning-Based Model for Anomaly Detection in Cloud Datacenter Networks*, **IEEE**

Transactions on Network and Service Management, vol. 16, no. 3, 2019, pp. 924-935, doi: 10.1109/TNSM.2019.2927886.

[121.] Rawat S., Srinivasan A., Ravi V. & Ghosh U., *Intrusion Detection Systems Using Classical Machine Learning Techniques vs Integrated Unsupervised Feature Learning and Deep Neural Network.* ***Internet Technology Letters***. Vol. 5, no.1, 2022, pp. e232, https://doi.org/10.1002/itl2.232.

[122.] M. Ioannis, R. Ioannis, G. Ioannis, K. George, D. Anastasios & D. Nikolaos, *Multiclass Confusion Matrix Reduction Method and Its Application on Net Promoter Score Classification Problem,* **School of Rural and Surveying Engineering, National Technical University of Athens, 9th, Heroon Polytechniou Str., 15773 Athens, Greece:** An Article (PETRA Conference Series, vol. 5, 2018, pp. 1-23.

Chapter Three

Methodology

3.1 Research Approach

Research Methodology is stated in this section. We can deduce that from the literature review, that there is an enormous need for the advancement of a potent machine learning/ deep learning models for identifying the threat in the dataset. Among the dataset, is the NSL-KDD dataset used for intrusion detection related research projects. NSL-KDD dataset was subjected to an experiment in order to analyze the dataset and teach them using three machine learning

algorithms such as hybrid decision trees, Naives Bayes (NB) and Random Forest (RF) to identify threats. This section also focuses on the steps taken to perform this research.

Series of actions done in order to get the outcome is described. Figure 3.1, displays several steps taken to develop this hybrid model. The numerated clarification of each step in the flow diagram is listed below:

i. First, the loaded and pre-processed NSL-KDD dataset is done using numerical one Hot-encoding.
ii. Then, redundant and irrelevant data is eliminated using Feature selection of the data
iii. Next, classifiers for all the attributes and for the reduced attributes for later comparison was trained by us.
iv. Three classifiers, such as Hybrid decision trees, Naïve Bayes classifier and Random forest are implemented using the steps built by a completed Hybrid model.
v. Precision, F-measure, Recall and Accuracy were yardsticks used for the assessment of the productivity of the model.

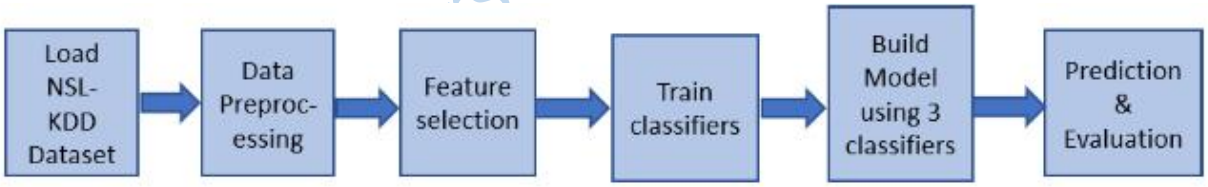


Figure 3.1: Sequence of actions for the Hybrid model

[Researcher: Fadeyi O, 2022]

3.2 System Design

The recent NSL-KDD dataset utilized in this research conquer the restraint of NSL-KDD cup 99 datasets. It is a lighter variety of the DARPA 99 dataset and it possess 42 unmistakable features; the select classes of this dataset can be contemplate from two ways.

i. As Binary class: NSL-KDD datasets with multiple features are exemplary for this type of datasets, in order to checkmate the complexity. Its binary as the target is simply '1 or 0'. '0' for usual activities on the network, '1' to indicate attacks. This groups all various attacks to the target value of '1'. The targets for this project follow this scheme.

ii. As Multi class: For numerous class targets, instead of responding using binary values by a target, you would have numerous target responses. There are 22 unmistakable groups of threats in this dataset, as such multiclass division would result in the model attempting to group out of 22 viable target responses. During the experiment it was noticed instantly from the outcomes of the maiden train-test process on the algorithm used that the use of the dataset for multiclass grouping is not idea.

Table 3.1: Feature Names and Data Types

Feature name	Data type
Duration	Continuous
src_bytes	Continuous
dst_bytes	Continuous
Land	Continuous
wrong_fragment	Continuous
Urgent	Continuous
Hot	Continuous
num_failed_logins	Continuous
logged_in	Continuous
num_compromised	Continuous
root_shell	Continuous
su_attempted	Continuous
num_root	Continuous
num_file_creations	Continuous
num_shells	Continuous

num_access_files	Continuous
num_outbound_cmds	Continuous
is_host_login	Continuous
is_guest_login	Continuous
Count	Continuous
srv_count	Continuous
serror_rate	Continuous
srv_serror_rate	Continuous
rerror_rate	Continuous
srv_rerror_rate	Continuous
same_srv_rate	Continuous
diff_srv_rate	Continuous
srv_diff_host_rate	Continuous
dst_host_count	Continuous
dst_host_srv_count	Continuous
dst_host_same_srv_rate	Continuous
dst_host_diff_srv_rate	Continuous
dst_host_same_src_port_rate	Continuous
dst_host_srv_diff_host_rate	Continuous
dst_host_serror_rate	Continuous
dst_host_srv_serror_rate	Continuous
dst_host_rerror_rate	Continuous
dst_host_srv_rerror_rate	Continuous

Table 3.1 shows some of the features in the features in the NSL-KDD datasets with the corresponding datatypes.

Source: Researcher's Computation, 2022

A very important step in preparation of data is Pre-processing of data which is fed into the algorithm. 42 features are contained in the dataset with the last column suggesting if it is normal and attack's name. 80% train set and the remaining 20% test set are contained in it. Both the training and testing set samples are displayed in table 3.2 and table 3.3 to demonstrate why data cleaning is essential.

	0	tcp	ftp_data	SF	491	0.1	0.2	0.3	0.4	0.5	...	25	0.17.1	0.03	0.17.2	0.24	0.25	0.26	0.05	0.27	normal
0	0	udp	other	SF	146	0	0	0	0	0	...	1	0.00	0.60	0.88	0.00	0.00	0.00	0.0	0.00	normal
1	0	tcp	private	S0	0	0	0	0	0	0	...	26	0.10	0.05	0.00	0.00	1.00	1.00	0.0	0.00	neptune
2	0	tcp	http	SF	232	8153	0	0	0	0	...	255	1.00	0.00	0.03	0.04	0.03	0.01	0.0	0.01	normal
3	0	tcp	http	SF	199	420	0	0	0	0	...	255	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.00	normal
4	0	tcp	private	REJ	0	0	0	0	0	0	...	19	0.07	0.07	0.00	0.00	0.00	0.00	1.0	1.00	neptune

5 rows × 42 columns

Table 3.2 Sample view of Train Dataset.

Source: Researcher's Computation, 2022

	0	tcp	private	REJ	0.1	0.2	0.3	0.4	0.5	0.6	...	10.1	0.04.1	0.06.1	0.22	0.23	0.24	0.25	1.2	1.3	neptune
0	0	tcp	private	REJ	0	0	0	0	0	0	...	1	0.00	0.06	0.00	0.00	0.00	0.0	1.00	1.00	neptune
1	2	tcp	ftp_data	SF	12983	0	0	0	0	0	...	86	0.61	0.04	0.61	0.02	0.00	0.0	0.00	0.00	normal
2	0	icmp	eco_i	SF	20	0	0	0	0	0	...	57	1.00	0.00	1.00	0.28	0.00	0.0	0.00	0.00	saint
3	1	tcp	telnet	RSTO	0	15	0	0	0	0	...	86	0.31	0.17	0.03	0.02	0.00	0.0	0.83	0.71	mscan
4	0	tcp	http	SF	267	14515	0	0	0	0	...	255	1.00	0.00	0.01	0.03	0.01	0.0	0.00	0.00	normal

5 rows × 42 columns

Table 3.3 Sample view of the Test Dataset.

Source: Researcher's Computation, 2022

Below are some of the issues possessed by the dataset addressed in this research;

i. Six missing groups are missing from the test set and the missing features from the train set have been revised.

ii. Names of the data column were missing and a fix was made by referring to the NSL-KDD pre-processing paper. A list of Column names were created and designated to each column, with 42nd column being the class of attack.

3.3 Research Methods

All data attributes encounter transformation of data procedure so as to input it inside the algorithm. A crucial step in this research is Data re-modeling. Next is the explanation of how it is carried out.

3.3.1 Data Pre-processing

There are 125,972 data records in the training set contained in the NSL-KDD dataset and the 22, 543 data records in the test set. Figure 3.2 and Figure 3.3 shows that data has categorical and numerical values. There is no null or missing values, which aids in the consistent and ease of identifying attacks accuracy.

The dataset undergoes different methodology to enable it fit for classification. Taking the direction of machine learning approaches are done during the data mining procedure;

i. One-Hot-encoding (one-of-k): The attributes are turned numerical using one-Hot-encoding. It can alter all categorical attributes into binary features. The input to this must be a matrix of integers will be Saprx matrix with each column holding value of one feature.

ii. Label-encoding: The attributes are changed from categorical to numerical using a label encoding methodology in machine learning via python. The attributes are first changed with Label-encoder from group to a number.

iii. Split Dataset: The dataset is then divided into four datasets for every assaulting category. Then, we change the name of every attack label as 0 for normal, 1 for DOS, 2 for Probe, 3 for R2L and 4 for U2R.

iv. In order to avoid too much of weigh on the results, the attributes are scaled to prevent the features from having large values.

3.3.2 Feature Selection

Recursive Feature Elimination (RFE) is executed to generate the vital attributes out of 41 attributes in the dataset. In this research, elimination of superfluous and immaterial data is done

by choosing a subset of applicable attributes that fully denotes the given problem. Attributes selection with ANOVA F-test. The strength of the relationship between the attributes and labels are determined by analyzing each feature individually. By using second percentile approach (sklearn.feature selection) to choose attribute built on percentile of the maximum scores. When this subset is found: Recursive Feature.

Eliminating (RFE) is done. Finding the best features is done by the eliminated (RFE)..

3.4 Classifiers Used for the Model

Three classifiers have been selected by us for building this model. Combining algorithm in the Hybrid model is to integrate both the machine learning, deep learning methodologies unsupervised learning methods and the supervised learning methods on the data is the reason for selecting this process. Measurement and validation of the performance of these classifiers on the NSL-KDD dataset was conducted using the experiment. Described below is the classifier chosen to build this hybrid machine learning models:

3.4.1 Random Forest Classifier

A very important machine learning algorithm is the Random Forest. It is an ensemble learning algorithm, implying that many decision trees are created (that is the reason for the name 'Forest' while in the training phrase of the algorithm and generates then the most widespread outcome as its classification. Regression and prediction problems make use of it. Speed and classification materialize very fast are the major advantages of Random Forest. It generates a forest of autonomous subsets of the dataset. Randomly selecting n variables at every individual node is initiated by best split.

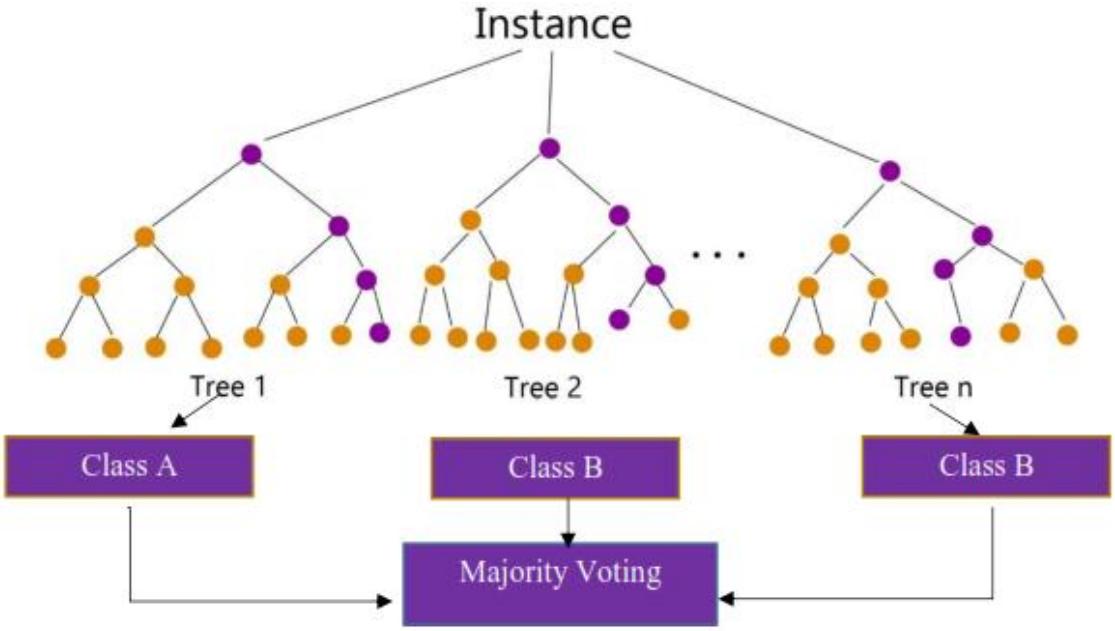


Figure 3.4: Random Forest in Operation.

Source: Researcher's Computation, 2022

3.4.2 Hybrid Decisions Tree

Supervised or unsupervised learning are utilized by Hybrid Decision Tree. Handling both categorical and numerical data well is one of the reasons for choosing decision tree. Recursive partitioning principle is used to construct and operate these trees. Pruning methods are used to solve issues with over-fitting.

3.4.3 Naive Bayes Classifier

Independence assumption in which probability of attributes are autonomous assumption in which prospect of features are autonomous of the other is the function of the method of supervised learning method. Precision results are almost always generated by the NB classifiers, the presence of errors could be attributed to the noisy data or other data modification. Accurate result is generated by NB as indicated by researchers.

3.4.5 K-Nearest Neighbors

A simple and precise machine learning algorithm that groups instances due to the nearest training instance in the feature space is known as K-Nearest (KNN). Both classification and regression can be solved using K-Nearest (KNN). An algorithm that generates no expression about the building of data and dispersion which implies that it is not-parametric algorithm is the KNN. Obeying theoretical rules by real life data is hardly possible which mirrors the strength of KNN. Prediction or cataloguing is built on fixed number (K) of training instances which is bordering to the input instance is how KNN works. This implies that a selected value of K, an input instance would be grouped or forecast to equal to the same class as closest number of K instances bordering to it.

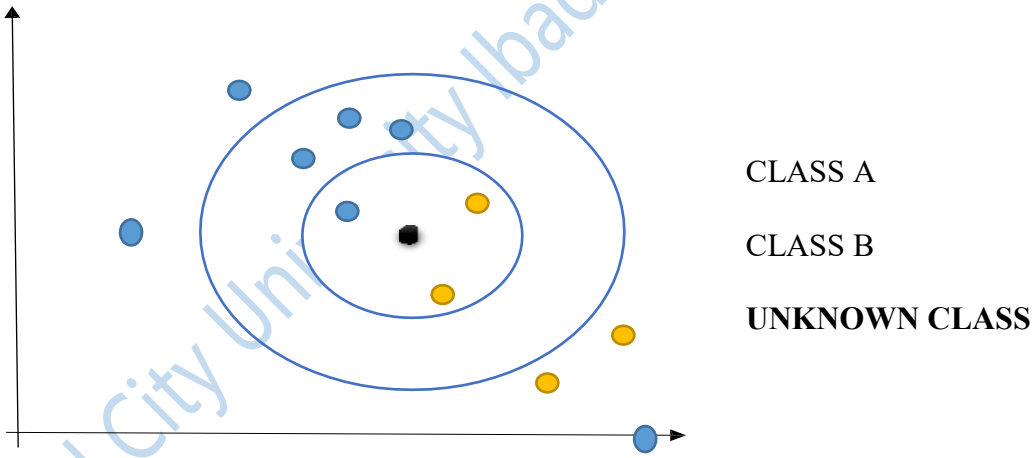


Figure 3.5 Example of KNN

Source: Researcher's Computation, 2022

The distance from the unknown to the nearest neighbors is measured using different methods. The most popular one is Euclidean distance, other popular methods include Manhattan distance, Hamming distance and Minkowski distance. Mathematically they can be represented by:

$$Euclidean\ distance\ = d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_1 - p_1)^2 + \ldots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \qquad Equation\ (1)$$

$$Manhattan\ distance = \ d_1(p,q) = \ \|p - q\| = \ \sum_{i=1}^{n} |p_i - q_i| \qquad Equation(2)$$

$$Hamming\ distance = \ d_{ij} \sum_{k=1}^{p} |x_{ik} - x_{jk}| \qquad Equation(3)$$

$$Minkowski\ distance = \ \sum_{i=1}^{n} (|x_i - y_i|^p)^{\frac{1}{p}} \qquad Equation(4)$$

Certain used cases are ideal for each distance type. The features are suited for different types while the Manhattan distance is the best match for KNN problems. While features are of the same type, the Euclidean distance is a match for opposite case. Due to forecasting accuracy, the important part of KNN algorithm is the value of 'k'. A task not to be taken for granted is the selection of the value of 'k'. lower accuracy result are most likely to occur with the selection of small values for k, if the training algorithm is done with noisy dataset, and low accuracy may occur due to over-fitting should the value of 'k' is too high. In order to prevent ties in the majority vote, an odd number 'k' should be selected should the number of classes are even.

3.5 Cross Validation

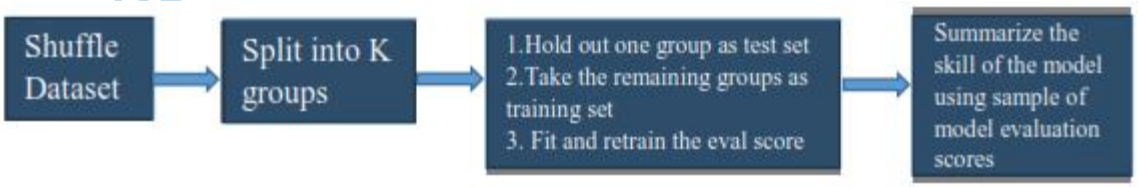


Figure 3.6: Cross Validation Process.

Source: Researcher's Computation, 2022

The inability to forecast the pattern they would behave on out of sample of new data is most method suffers accuracy evaluation. This drawback also affects the K nearest neighbour algorithm. Solution is the use of cross validation. From the chart of figure 3.2, first is the shuffling of the dataset, then the splitting into k groups (into a huge category of training, a smaller class of testing) repetition of this is k times until each section of the divide data is used for training data and testing data.

Three methods of cross validation are used namely:

1. K-fold method: Equal sizes of k samples is achieved after the original dataset is shuffled.

One of the k samples is assigned to be used as a testing sample for the model. The other remaining samples are reserved for training the model. There are repetition of the process k times with each of the other k – 1 sample. There is an error value for each run of the cross validation process. All the k models errors are calculated by finding the average. Doing this reduces the variance and keeps accuracy among the models. Running time using this method is a disadvantage.

2. Holdout method: In this stage, samples are spitted into two. One section for training while the other for testing. Outcome of this exercise is that the training set is larger than the testing set. Fitting the model is done by the larger set while the testing set is second set is used for model testing. Simplest cross validation method is this because primarily it is single cross validation. Running time is an advantage over the K-fold method by the holdout method.

3. Leave one out method: K-fold method has similarity with this method, with k having a very large value, generally as large as the sample universe. Every value of the dataset is trained using the model except one, which is left out for testing. This mitigate against discrepancy but is resource and time intensive.

3.6 Tools and Environment

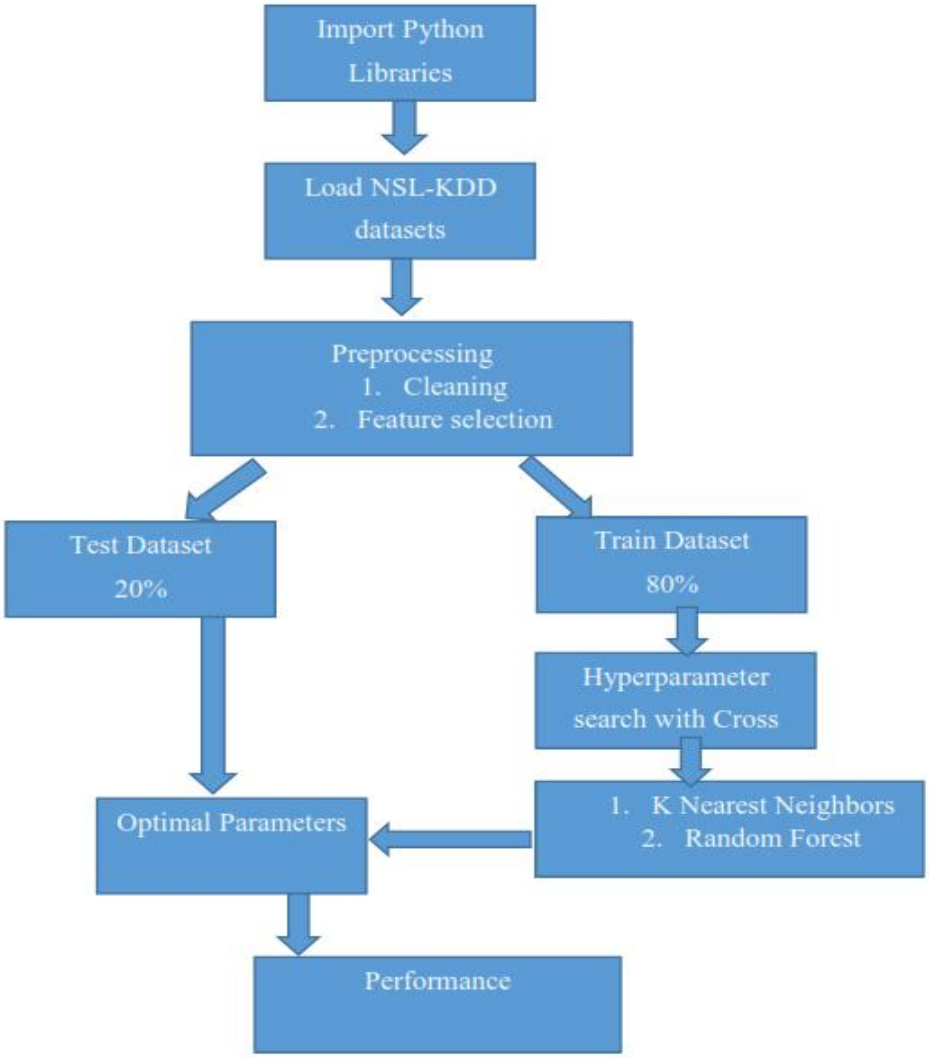


Figure 3.7: System Flow diagram

Source: Researcher's Computation, 2022

3.6.1 Anaconda

Anaconda comprises of python and R spreading for machine learning application and data science. It is open source and free software and comprises 1400 packages. The major advantage of using Anaconda is that it serves as a centralized libraries for would be need for scientific computing. The terminals are used to installed any required library.

3.6.2 Jupyter Notebook

Computations within the browsers of the user's system is allowed by the environment of the Jupyter notebook. Contained in the Jupyter notebook which is simply a JSON document, is an ordered list of I/O cells which could encompass code, mathematics, text, graph/plots and other media.

Users of Jupyter notebook can test code in browser, without the need for rerun of the whole code every time the researcher wants to test on just a minute block of code. This gives room for a lot of flexibility.

3.6.3 Scikit-Learn

Scikit-learn is a free library for the scientific computation using python language. It comprises numerous algorithms for clustering, classification and regression.

3.6.4 NumPy

Numpy is a python library that adds support for large multi-dimensional arrays and matrices. It also comprises of groups of high-level mathematic functions to work on arrays.

3.6.5 Matplotlib

This is a python and numpy plotting library. This accommodates for plotting different kinds of graphical representation of data. This allows for plotting various kinds of graphical representation of data.

3.6.6 Pandas

Pandas is another python library that is utilized for data handling and breakdown. Pandas utilizes data frames. Pandas contains lot of inbuilt tools that could be used for a host of things. One vital function is data cleaning. Data cleaning consist of 80% of the time for the major activity carried out when using machine learning, according to IBM analytics. Many datasets have blank fields which can greatly affect a model negatively. An example of a function in pandas for dealing with this is the '.isnull()' function.

3.6.7 System Configuration

This research was implemented on a system with the following configuration:

i. Core i3 CPU
ii. 6 GB RAM
iii. Windows 10 Operating System
iv. Python 3.7

Chapter Four

Results and Discussion of Findings

4.1 Overview of the Experiment

In this chapter, the results of the experiments carried out based on the methodology described in the previous chapter are presented and discussed. All the experiments were implemented with

PYTHON programming language with its associated libraries used in machine learning as numpy, pandas, scikit-learn because its codes are open-source and that it is portable across many platforms.

The experiment was divided into three categories based on the research objectives.

i. Experiment on the Dataset Loading and Preprocessing

ii. Experiment on the improved Intrusion Detection System

iii. Experiment to test and evaluate the developed system

Finally, the results of all the experiments are presented and discussed.

4.2 Experiment on the Dataset Loading and Preprocessing

The NSL-KDD dataset (both the training the test subsets) were loaded into the Jupyter Notebook environment from where it is stored and the column headings for both subsets were appended as shown in Figure 4.1

```
train=pd.read_csv('NSL_Dataset\Train.txt',sep=',')
test=pd.read_csv('NSL_Dataset\Test.txt',sep=',')
columns=["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","urgent","hot",
         "num_failed_logins","logged_in","num_compromised","root_shell","su_attempted","num_root","num_file_creations",
         "num_shells","num_access_files","num_outbound_cmds","is_host_login","is_guest_login","count","srv_count","serror_rate",
         "srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate",
         "dst_host_count","dst_host_srv_count","dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
         "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerror_rate",
         "dst_host_srv_rerror_rate","attack","last_flag"]
train.columns=columns
test.columns=columns
```

Figure 4.1: Snapshot of NSL-KDD Dataset Being Loaded Into the Jupyter Notebook

Source: Researcher's Computation, 2022

In Figure 4.2 a, the first five rows of the training sub dataset is previewed starting from the first column header. Since there are 43 columns in all, some columns were not shown. This is indicated by the three dots (…) as shown in the Figure 4.2.a. Also in Figure 4.2.b the same preview was shown but from the last column reading from the right to the left. The first five rows of the test sub dataset can also be displayed by modifying the train.head() code in Figure 4.2 to test.head().

```
train.head()
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	

5 rows x 43 columns

Figure 4.2 a: Snapshot of the Loaded Dataset from the First Column

Source: Researcher's Computation, 2022

```
train.head()
```

dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate	attack
0.88	0.00	0.00	0.00	0.0	0.00	normal
0.00	0.00	1.00	1.00	0.0	0.00	neptune
0.03	0.04	0.03	0.01	0.0	0.01	normal
0.00	0.00	0.00	0.00	0.0	0.00	normal
0.00	0.00	0.00	0.00	1.0	1.00	neptune

Figure 4.2 b: Snapshot of the Loaded Dataset from the Last Column

Source: Researcher's Computation, 2022

The different types of attack that should be detected by the Network Intrusion Detection Systems are shown in Figure 4.2 b under the 'attack' column. These attacks types are grouped into the following four categories as part of the dataset pre-processing tasks:

i. DOS: Denial of service is an attack category, which depletes the victim's resources thereby making it unable to handle legitimate requests. There are ten attacks that fall under this category, they are: Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, and Worm.

ii. Probe: Surveillance and other probing attack's objective is to gain information about the remote victim. There are six attacks that fall under this category, they are: Satan, Ipsweep, Nmap, Portsweep, Mscan, and Saint.

iii. R2L: This is an unauthorized access to local super user (root) privileges is an attack type, by which an attacker uses a normal account to login into a victim system and tries to gain root/administrator privileges by exploiting some vulnerability in the victim. There are sixteen attacks that fall under this category, they are: Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named.

iv. U2L: This is an unauthorized access from a remote machine, the attacker intrudes into a remote machine and gains local access of the victim machine. There are seven attacks that fall under this category, they are: Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps.

These categories are used to form another column in the datasets (train and test) named attack_class as shown in Figure 4.3 and Figure 4.4. In attack_class normal means 0, DOS means 1, PROBE means 2, R2L means 3 and U2R means 4.

```
train.loc[train.attack=='normal','attack_class']=0

train.loc[(train.attack=='back') | (train.attack=='land') | (train.attack=='pod') | (train.attack=='neptune') |
        (train.attack=='smurf') | (train.attack=='teardrop') | (train.attack=='apache2') | (train.attack=='udpstorm') |
        (train.attack=='processtable') | (train.attack=='worm') | (train.attack=='mailbomb'),'attack_class']=1

train.loc[(train.attack=='satan') | (train.attack=='ipsweep') | (train.attack=='nmap') | (train.attack=='portsweep') |
         (train.attack=='mscan') | (train.attack=='saint'),'attack_class']=2

train.loc[(train.attack=='guess_passwd') | (train.attack=='ftp_write') | (train.attack=='imap') | (train.attack=='phf') |
         (train.attack=='multihop') | (train.attack=='warezmaster') | (train.attack=='warezclient') | (train.attack=='spy') |
         (train.attack=='xlock') | (train.attack=='xsnoop') | (train.attack=='snmpguess') | (train.attack=='snmpgetattack') |
         (train.attack=='httptunnel') | (train.attack=='sendmail') | (train.attack=='named'),'attack_class']=3

train.loc[(train.attack=='buffer_overflow') | (train.attack=='loadmodule') | (train.attack=='rootkit') | (train.attack=='perl') |
         (train.attack=='sqlattack') | (train.attack=='xterm') | (train.attack=='ps'),'attack_class']=4
```

Figure 4.3: Train Dataset Attack Type Classification

Source: Researcher's Computation, 2022

```
test.loc[test.attack=='normal','attack_class']=0

test.loc[(test.attack=='back') | (test.attack=='land') | (test.attack=='pod') | (test.attack=='neptune') |
        (test.attack=='smurf') | (test.attack=='teardrop') | (test.attack=='apache2') | (test.attack=='udpstorm') |
        (test.attack=='processtable') | (test.attack=='worm') | (test.attack=='mailbomb'),'attack_class']=1

test.loc[(test.attack=='satan') | (test.attack=='ipsweep') | (test.attack=='nmap') | (test.attack=='portsweep') |
         (test.attack=='mscan') | (test.attack=='saint'),'attack_class']=2

test.loc[(test.attack=='guess_passwd') | (test.attack=='ftp_write') | (test.attack=='imap') | (test.attack=='phf') |
         (test.attack=='multihop') | (test.attack=='warezmaster') | (test.attack=='warezclient') | (test.attack=='spy') |
         (test.attack=='xlock') | (test.attack=='xsnoop') | (test.attack=='snmpguess') | (test.attack=='snmpgetattack') |
         (test.attack=='httptunnel') | (test.attack=='sendmail') | (test.attack=='named'),'attack_class']=3

test.loc[(test.attack=='buffer_overflow') | (test.attack=='loadmodule') | (test.attack=='rootkit') | (test.attack=='perl') |
         (test.attack=='sqlattack') | (test.attack=='xterm') | (test.attack=='ps'),'attack_class']=4
```

Figure 4.4: Test Dataset Attack Type Classification

Source: Researcher's Computation, 2022

The effect of the code in Figure 4.4 is shown in Figure 4.5 under the attack_class column heading. There are now 44 column headings as against the 43 column headings in the initial train sub-dataset. The case is the same with the test sub-dataset. The attack-class column is also added to it and there are now 44 columns in all.

```
train.head()
```

ort_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate	attack	last_flag	attack_class
0.88	0.00	0.00	0.00	0.0	0.00	normal	15	0.0
0.00	0.00	1.00	1.00	0.0	0.00	neptune	19	1.0
0.03	0.04	0.03	0.01	0.0	0.01	normal	21	0.0
0.00	0.00	0.00	0.00	0.0	0.00	normal	21	0.0
0.00	0.00	0.00	0.00	1.0	1.00	neptune	21	1.0

Figure 4.5: Snapshot Showing the Modified Dataset from the Last Column

Source: Researcher's Computation, 2022

Furthermore, since machine learning algorithms perform better with numerical data, the features with the categorical values are converted into numerical datatype using the one-hot encoding technique. There are four features with categorical values. The features are 'protocol_type', 'service', 'flag' and 'attack'. Figure 4.6 shows the code that created the dummy variables (columns) for every possible occurrence of each categorical features.

```
# An utility function to create dummy variable
def create_dummies( df, colname ):
    col_dummies = pd.get_dummies(df[colname], prefix=colname, drop_first=True)
    df = pd.concat([df, col_dummies], axis=1)
    df.drop( colname, axis = 1, inplace = True )
    return(df)
```

Figure 4.6: A Utility Function for Dummy Variable Creation

Source: Researcher's Computation, 2022

In Figure 4.7, the utility function is applied to both the train and test sub-datasets and each categorical feature in both sub datasets was replaced with the dummies variables created for that feature.

```
#for c_feature in categorical_features
for c_feature in ['protocol_type', 'service', 'flag', 'attack']:
    train_cat = create_dummies(train_cat,c_feature)
    test_cat = create_dummies(test_cat,c_feature)
train_cat.head()
```

Figure 4.7: Application of Utility Function for Dummy Variable Creation

Source: Researcher's Computation, 2022

Figure 4.8 shows the result of the one-hot encoding technique on the categorical features. There are 103 columns headings replacing the original four columns. For instance, using protocol_type categorical feature because it has only two type: tcp and udp. The 'protocol_type' column heading in the sub datasets (train and test) are replaced with protocol_type_tcp and protocol_type_udp. Also, for other three categorical features: 'service', 'flag' and 'attack', each of these column heading is replaced with every possible combination that can be derived under that column heading.

	protocol_type_tcp	protocol_type_udp	service_X11	service_Z39_60	service_aol	service_auth	service_bgp	service_courier	service_csnet_ns	service_ctf	...
0	0	1	0	0	0	0	0	0	0	0	...
1	1	0	0	0	0	0	0	0	0	0	...
2	1	0	0	0	0	0	0	0	0	0	...
3	1	0	0	0	0	0	0	0	0	0	...
4	1	0	0	0	0	0	0	0	0	0	...

5 rows x 103 columns

Figure 4.8: Result of One-Hot Encoding on the Categorical Features

Source: Researcher's Computation, 2022

The final sub datasets after all the pre-processing tasks has been done is shown in Figure 4.9. There are now 143 columns as against 43 columns in the un-preprocessed datasets. Since there are no missing values in the NSL-KDD dataset, there was no need for carrying out any missing value pre-processing task.

```
train_new = pd.concat([train_num, train_cat], axis=1)
test_new = pd.concat([test_num, test_cat], axis=1)
train_new.head()
```

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...	attack_phf	attack_pod	attack_ports
0	0.0	146	0	0	0	0	0	0	0	0	...	0	0	
1	0.0	0	0	0	0	0	0	0	0	0	...	0	0	
2	0.0	232	8153	0	0	0	0	0	1	0	...	0	0	
3	0.0	199	420	0	0	0	0	0	1	0	...	0	0	
4	0.0	0	0	0	0	0	0	0	0	0	...	0	0	

5 rows x 143 columns

Figure 4.9: The Final Pre-Processed Dataset

Source: Researcher's Computation, 2022

4.3 Result of Feature Selection

The speed of any machine learning algorithm is impacted by the nature and characteristics of data in the dataset such as the number of rows (instances), the number of columns (features), the correlation between the features etc. High number of instances is desirable because, the higher this number, the more reliable is the prediction made by the algorithm. However, when the number of feature is too many (usually referred to as the curse of dimensionality), it degrades the performance and slow down the execution time of the algorithm. The section discusses the result of feature reduction techniques used to address this dimensionality problem. As there are 143 features, it as a result of the preprocessing steps in section 4.1.

The correlation matix was computed on the final sub datasets and it was discovered from Figure 4.10 that the features with NaN value have no (zero) correlation with every other feature in each of the sub datasets. Also, there are features with low variance (near zero variance), high missing values (>25%) and those with high correlation between two numerical variables. All these features have minimal discriminatory power and are removed from both sub datasets. Figure 4.11 shows how these non-discriminatory features are dropped from the training sub dataset.

```
# correlation matrix (ranges from 1 to -1)
corrm=train_new.corr()
corrm
```

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...	attack_phf
duration	1.000000	0.011740	0.036892	NaN	NaN	NaN	0.021274	NaN	-0.064538	0.062081	...	-0.000757
src_bytes	0.011740	1.000000	0.127827	NaN	NaN	NaN	0.379973	NaN	0.163275	0.603363	...	-0.000898
dst_bytes	0.036892	0.127827	1.000000	NaN	NaN	NaN	0.127002	NaN	0.421999	0.237363	...	0.010050
land	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
wrong_fragment	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
attack_smurf	-0.020279	-0.002928	-0.051676	NaN	NaN	NaN	-0.020616	NaN	-0.118539	-0.014876	...	-0.000825
attack_spy	0.000605	-0.000555	-0.000156	NaN	NaN	NaN	-0.000561	NaN	0.000850	-0.000405	...	-0.000022
attack_teardrop	-0.011691	-0.013779	-0.029791	NaN	NaN	NaN	-0.011886	NaN	-0.068341	-0.008576	...	-0.000476
attack_warezclient	0.037013	0.047995	-0.011541	NaN	NaN	NaN	0.270177	NaN	0.104233	-0.008567	...	-0.000475
attack_warezmaster	-0.001571	-0.002012	0.064962	NaN	NaN	NaN	0.003599	NaN	-0.007621	-0.001280	...	-0.000071

143 rows x 143 columns

Figure 4.10: Correlation Matrix on Train Sub Dataset

Source: Researcher's Computation, 2022

The features are: 'land', 'wrong_fragment', 'urgent', 'num_failed_logins', 'root_shell', 'su_attempted', 'num_root', 'num_file_creations', 'num_shells', 'num_access_files', 'num_outbound_cmds', 'is_host_login', 'is_guest_login', 'dst_host_rerror_rate', 'dst_host_serror_rate', 'dst_host_srv_rerror_rate', 'dst_host_srv_serror_rate', 'num_root', 'num_outbound_cmds', 'srv_rerror_rate' and 'srv_serror_rate'.

```
train_new.drop(columns=['land','wrong_fragment','urgent','num_failed_logins',"root_shell","su_attempted","num_root",
                        "num_file_creations","num_shells","num_access_files","num_outbound_cmds","is_host_login","is_guest_login",
                        'dst_host_rerror_rate','dst_host_serror_rate','dst_host_srv_rerror_rate','dst_host_srv_serror_rate',
                        'num_root','num_outbound_cmds','srv_rerror_rate','srv_serror_rate'], axis=1, inplace=True)
```

Figure 4.11: Dropping the Irrelevant Features from the Training Sub Dataset

Source: Researcher's Computation, 2022

After the removal of the non-discriminatory features, there are still many features left. Since having too many features will slow down the learning process, SelectKBest technique is used to select the best features from among the remaining 123 features. Figure 4.12 shows the line of codes that is used in selecting 15 features that have the highest scores among the 123 features. The default k value is 10 but 15 is used to avoid have too low and the same time to prevent having too many fields/ parameters to enter when used by the end users.

```
X = train_new[train_new.columns.difference(['attack_class'])]
X_new = SelectKBest(f_classif, k=15).fit(X, train_new['attack_class'] )
```

Figure 4.12: Selecting Best 15 features from the train sub dataset

Source: Researcher's Computation, 2022

The best 15 features are printed out and displayed as shown in Figure 4.13. The features are as follows: attack_neptune, attack_normal, attack_satan, count, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_same_srv_rate, dst_host_srv_count, flag_S0, flag_SF, last_flag, logged_in, same_srv_rate, serror_rate, service_http

```
# capturing the important variables
KBest_features=X.columns[X_new.get_support()]
KBest_features
```

```
Index(['attack_neptune', 'attack_normal', 'attack_satan', 'count',
       'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
       'dst_host_same_srv_rate', 'dst_host_srv_count', 'flag_S0', 'flag_SF',
       'last_flag', 'logged_in', 'same_srv_rate', 'serror_rate',
       'service_http'],
      dtype='object')
```

Figure 4.13: Printing Out the Best 15 Features

Source: Researcher's Computation, 2022

The feature selection step has been concluded with the selection of the 15 best features based on their predictive/discriminatory power and only these features will be used in training and building the intrusion detection model.

4.4 Result of Training and Testing the Intrusion Detection Classifiers

In the section, the results of the training and testing different the intrusion detection classifiers is presented and the machine learning algorithm that produced the best predictive accuracy is eventually used to implement the intrusion detection system.

In Figure 4.14, the two (train and test) sub datasets are further separated into the following parts viz: X_train, X_test, y_train and y_test using 80:20 ratio. X_train is the train data with all independent variable and y_train is the train data with the dependent (output) variable. Also, X_test and y_test are test data with all the independent variables and the test data with the dependent (output) variable respectively. The top_features contain the 15 best selected features.

```
top_features=['attack_neptune','attack_normal','attack_satan','count','dst_host_diff_srv_rate','dst_host_same_src_port_rate','dst_
X_train = train[top_features]
y_train = train['attack_class']
X_test = test[top_features]
y_test = test['attack_class']
```

Figure 4.14: Splitting the Dataset

Source: Researcher's Computation, 2022

4.4.1 Logistic Regression

The figure 4.15 show the result of logistic regression algorithm. It uses the cross-entropy loss because the 'multi_class' option is set to 'multinomial' and the algorithm implemented regularized logistic regression using 'lbfgs' solvers. This is because 'lbfgs' solvers support L2 regularization (optimization). The resulting model was used predict the intrusion class referred to as the y_pred. From figure 4.15, the accuracy score of the LogisticRegression model built was approximately 0.8377.

```
lr_clf = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomial').fit(X_train, y_train)
```

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

```
array([1., 0., 2., ..., 1., 0., 2.])
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.8376879740939538
```

Figure 4.15: Result of the Logistic Regression Model

Source: Researcher's Computation, 2022

4.4.2 K-Nearest Model

The KNeighborsClassifier was also used to build the intrusion detection model using the n_neighbors parameter which was set to 3 nearest neighbors. The resulting model was used predict the intrusion class, y_pred. From figure 4.16, the accuracy score of the K-nearest model was approximately 0.7538.

```
from sklearn.neighbors import KNeighborsClassifier
```

```
k_neigh = KNeighborsClassifier(n_neighbors=3)
k_neigh.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

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

```
array([1., 0., 2., ..., 1., 0., 0.])
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.7538482012154549
```

Figure 4.16: Result of the K-Nearest Model

Source: Researcher's Computation, 2022

4.4.3 Decision Trees

The DecisionTreeClassifier was also used to build the intrusion detection model. The two important parameters that are set are the maximum depth, max_depth, of the tree and the maximum number of features, max_feature, to be used. The two parameters are set to 8, respectively because this value gave optimal result from various experiments. The resulting model was used predict the intrusion class, tree_test_pred. From Figure 4.17, the accuracy score of the DecisionTreeClassifier model was approximately 0.8127.

```
clf_tree = DecisionTreeClassifier( max_depth = 8, max_features=8 )
clf_tree.fit( X_train, y_train )
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=8, max_features=8, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
accuracy_score( tree_test_pred.actual, tree_test_pred.predicted )
```

```
0.8127134809031629
```

Figure 4.17: Result of The Decision Tree Model

Source: Researcher's Computation, 2022

4.4.4 Linear Support Vector Classification (Linear SVC)

The LinearSVC classifier was also used to build the intrusion detection model. The two important parameters that are set here are the random state which controls the pseudo random number generation for shuffling the data and the tolerance for stopping criteria, tol. The radom_state value was set to 0, while tol was set to 1e-5 because this value gave optimal result from various experiments. The resulting model was used predict the intrusion class, y_pred. From Figure 4.18, the accuracy score of the LinearSVC model was approximately 0.8101.

```
from sklearn.svm import LinearSVC
```

```
svm_clf = LinearSVC(random_state=0, tol=1e-5)
svm_clf.fit(X_train, y_train)
```

```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=0, tol=1e-05,
          verbose=0)
```

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

```
array([1., 0., 2., ..., 1., 0., 2.])
```

```
accuracy_score( y_test, y_pred )
```

```
0.8100519008117819
```

Figure 4.18: Result of the Linear Svc Model

Source: Researcher's Computation, 2022

4.4.5 Neural Network Model

Multi-layer Perceptron classifier is a class of Neural Network model that is suitable for classification task. It optimizes the log-loss function using LBFGS or stochastic gradient descent. The MLPClassifier was used to build the intrusion detection model. The important parameter that is set here is the hidden_layer_sizes represents the number of neurons in the ith hidden layer. This parameter value was set to a tuple (30, 30, 30), because this value gave optimal result from various experiments. The resulting model was used predict the intrusion class, y_pred. From Figure 4.19, the accuracy score of the MLPClassifier model was approximately 0.8369.

```
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
scaler = StandardScaler()
# Fit only to the training data
scaler.fit(X_train)
```

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

```
# Now apply the transformations to the data:
train_X = scaler.transform(X_train)
test_X = scaler.transform(X_test)
```

```
mlp = MLPClassifier(hidden_layer_sizes=(30,30,30))
mlp.fit(train_X,y_train)
```

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(30, 30, 30), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

```
accuracy_score( y_test, y_pred )
```

```
0.8368895000665395
```

Figure 4.19: Result of the Neural Network Model

Source: Researcher's Computation, 2022

4.4.6 Random Forest

The RandomForestClassifier is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree. The resulting model was used predict the intrusion class, y_pred. From Figure 4.20, the accuracy score of the RandomForestClassifier model was approximately 0.8298.

```
model.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features=3,
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

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

```
array([1., 0., 2., ..., 1., 0., 2.])
```

```
accuracy_score( y_test, y_pred )
```

```
0.8297919531561904
```

Figure 4.20: Result of the Random Forest Model

Source: Researcher's Computation, 2022

4.5 The Improved Network Intrusion Detection System (ImNIDS)

The LogisticRegression outperformed all other classifiers that were examined during the experiments, as shown in Figure 4.21, and because of its performance, it was employed in developing the improved Network Intrusion Detection Systems. Figure 4.21 is a summary of all the classifiers with their corresponding accuracy scores. While LogisticRegression has the best accuracy score, kneighbors classifier has the least score.

4.6 Evaluation (Discussion of Result)

The Improved Network Intrusion Detection System (ImNIDS) is a web-based application that can be deployed from any node or host computer in an organizational/ enterprise network. The interface and the mode of its operation are shown in Figure 4.22 and Figure 4.23.

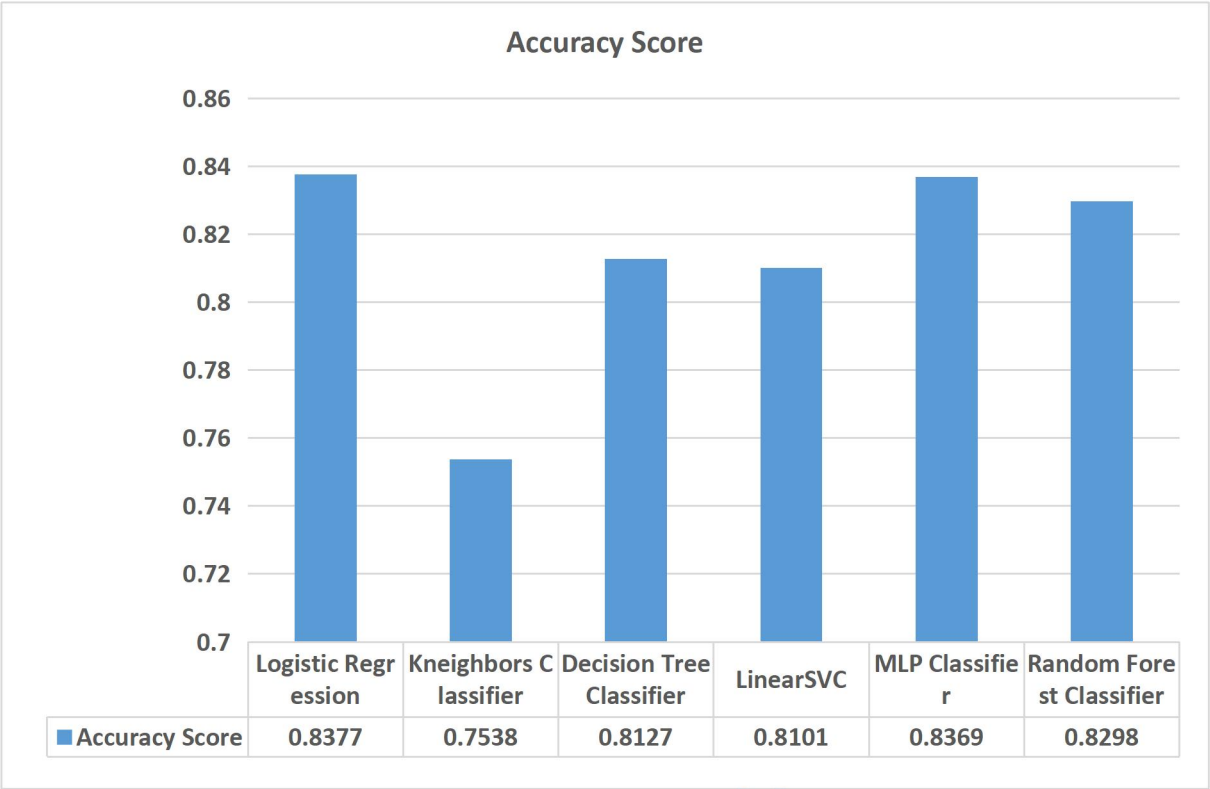


Figure 4.21 Classifiers Verses Accuracy Score

Source: Researcher's Computation, 2022

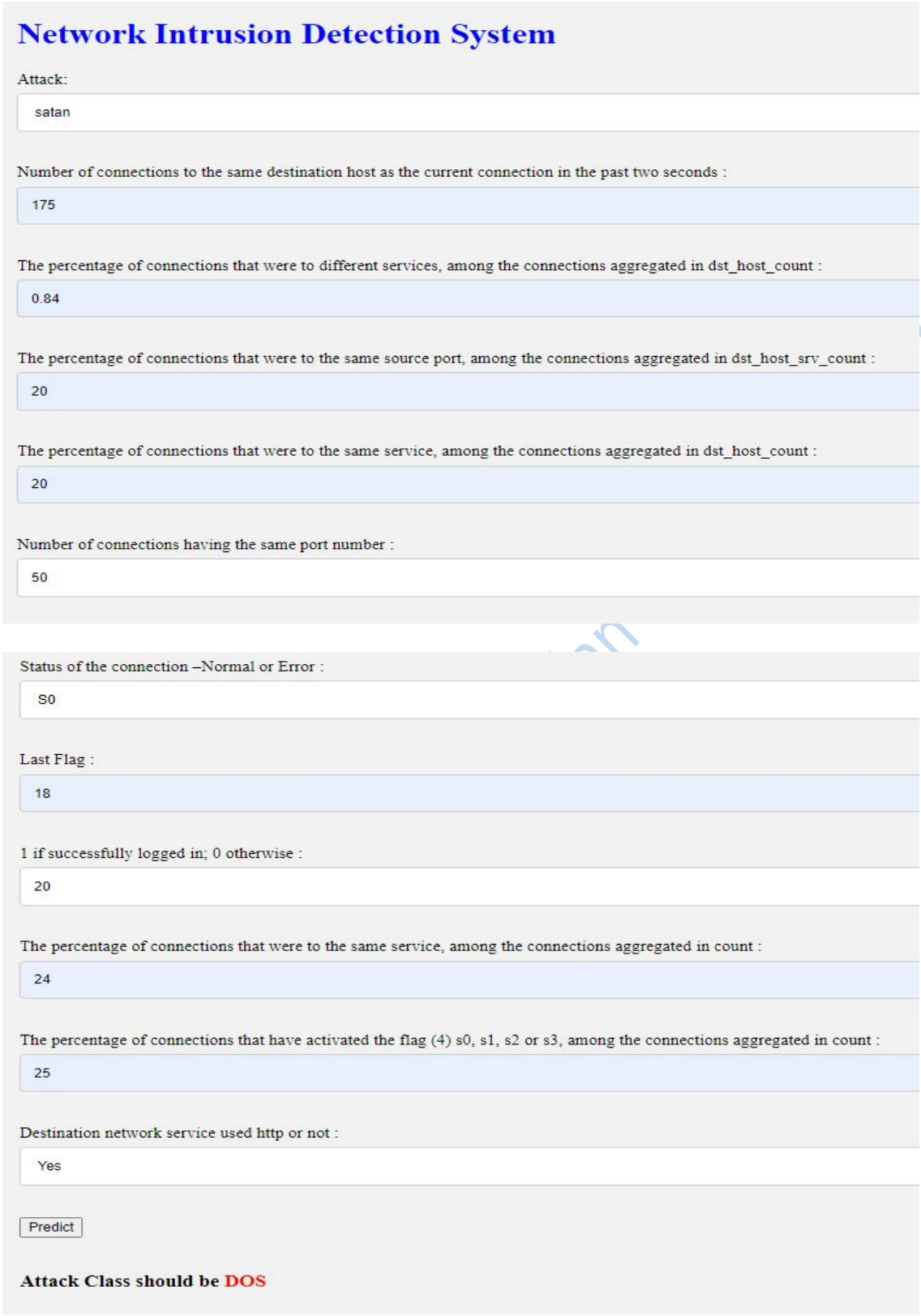


Figure 4.22: Detection of DOS Attack by ImNIDS

Source: Researcher's Computation, 2022

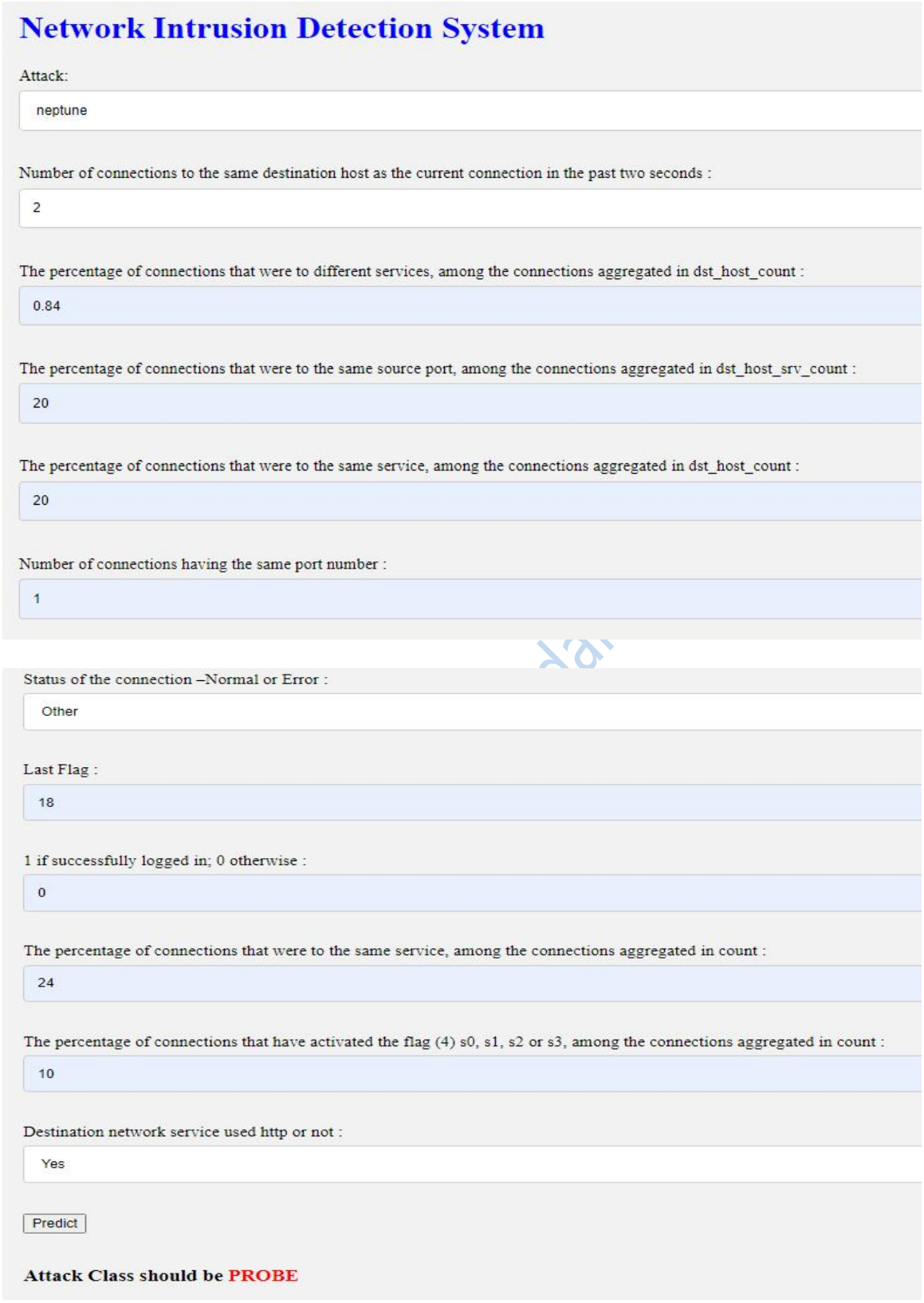


Figure 4.23: Detection of PROBE Attack by ImNIDS

Source: Researcher's Computation, 2022

Chapter Five

Conclusion

5.1 Summary of Findings

In many researches about the machine learning in network security, the significance of feature selection solution has been emphasized repeatedly. Machine learning models perform better with numerical data than the categorical data. Hence the features with categorical values were transformed into numerical value. Applying the advanced feature selection approaches has brought significant improvement to the performance of Network Intrusion Detection System, as good feature selection results improve learning accuracy, reduce learning time, and simplify learning results. On one hand, using key features that are interrelated with each other with machine learning approach effectively reduce the iterations of the experiment. On the other hand, obtaining the optimal feature subset helped to effectively shrink the size of features that are used for training the machine learning model, which further lower the impact of over-fitting and provide better generalization of models. Besides, less features fed into machine learning model save a lot of time on data processing and model training. Eliminating the irrelevant features by applying the feature selection techniques improves the classification accuracy of machine learning model.

The performance comparison among different classifiers was made in order to understand their effectiveness in terms of accuracy scores. From results, it is clear that every attributes in data set is not of equal importance, as we can ignore some attributes over others which does not involve much in intrusion detection.

5.2 Conclusion

This thesis has been able to deliver an intrusion detection system which is resistant to noise and have speedy detection rate against attacks in the network. The elimination of many unwanted features which constitutes 'noise' were eliminated, leaving the most important features. Unlike many traditional IDS which is laden with many features beyond necessary, with the mindset to enhance the performance of its operation but, eventually slows, inhibits and reduce the effectiveness of its operations. This has greatly improved the speed of processing the detection and raising the alert of unwanted intruder.

5.3 Contribution to Knowledge

This thesis will add to the lists of unending Intrusion Detection System with a major goal in mind; to improve in the area of noise interfering with the performance of the IDS. Thus, an IDS with a faster response time to intrusion and ability to categorize the intrusion more accurately is thereby created. This is achieved by streamlining the number of features the IDS would be working with to only the most important ones

5.4 Recommendation

This work can be improved on by using more related datasets other than the NSL-KDD dataset this will make comparison across multiple datasets possible. Also, other machine learning techniques and deep learning techniques could be used. The implementation of other attributes and functions can be added to further improve the future IDS better than where the thesis covered. Furthermore, the effect of optimization techniques such as Ant-colony, bee colony, genetic

algorithm etc. could be observed to either justify or discourage their continued usage in this area of study.

5.5 Suggestion for Further Studies

This thesis used one type of dataset for its training and test. It is therefore suggested that more than one dataset can be employed to get a better accuracy that what this thesis has accomplished.

Bibliography

Conference Proceedings

Abdulrahman K. A. A. & Muhammad I., *Distributed Denial of Service Attack Alleviated and Detected by Using Mininet and Software Defined Network*, Webology, vol. 19, 2022, pp. 1 – 16, DOI: 10.14704/WEB/V19I1/WEB19272.

Ahn S., Yi H., Lee Y., Ha W. R., Kim G. & Paek Y., *Hawkware: Network Intrusion Detection Based on Behavior Analysis with ANNs on an IoT Device*, in *Proc. 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1-6.

Alani M. M. & Awad A. I., *An Intelligent Two-Layer Intrusion Detection System for the Internet of Things*, IEEE Transactions *on* Industrial Informatics, vol. 19, no. 1, Jan. 2023, pp. 683-692, doi: 10.1109/TII.2022.3192035.

Alavizadeh H., Alavizadeh H. & Jang-Jaccard J., *Deep Q-Learning Based Reinforcement Learning Approach for Network Intrusion Detection*. Computers**,** vol. 11, 2022, pp. 41. https://doi.org/10.3390/computers11030041.

Ali M.H., Mohammed B. A. D. A, Ismail A., & Zolkipli M. F., *A New Intrusion Detection System Based on Fast Learning Network and Particle Swarm Optimization.* IEEE Access, vol. 6, 2018, pp. 20255–20261. https://doi.org/10.1109/ACCESS.2018.2820092.

Alqahtani H., Sarker I. H., Kalim A., Minhaz S. M. H., Ikhlaq S., & Hossain S., *Cyber Intrusion Detection using Machine Learning Classification Techniques*, In Computing Science, Communication and Security: First International Conference, COMS2 2020*,* 2020*,* pp. 121-131.

Andresini G., Appice A., Mauro N. D., Loglisci C. & Malerba D., *Multi-Channel Deep Feature Learning for Intrusion Detection*. IEEE Access, vol. 8, 2020, pp. 53346–53359. https://doi.org/10.1109/ACCESS.2020.2980937.

Aslan O., Ozkan-Okay M. & Gupta D., *Intelligent Behavior-based Malware Detection System on Cloud Computing Environment*, IEEE Access, vol. 9, 2021, pp. 83252-83271.

Awad M., Fraihat S., Salameh K. & Al Redhaei A. *Examining the Suitability of NetFlow Features in Detecting IoT Network Intrusions*, Sensors, *vol. 22, no. 16, 2022,* https://doi.org/10.3390/s22166164.

Bakshi T., *State of the Art and Recent Research Advances in Software Defined Networking*. In Wireless Communications and Mobile Computing, vol. 2017, 2017, https://doi.org/10.1155/2017/7191647.

Beachy A., Bae H., Camberos J. A., Grandhi R. V., *Epistemic Modeling Uncertainty of Rapid Neural Network Ensembles for Adaptive Learning*, Finite Elements in Analysis and Design, vol. 228, 2023, https://doi.org/10.1016/j.finel.2023.104064.

Bedi P., Gupta N., & Jindal V., *Siam-IDS: Handling Class Imbalance Problem in Intrusion Detection Systems using Siamese Neural Network*, Procedia Computer Science, Vol 171, 2020, pp 780-789.

Bhoopesh S. B., Dikshita, Nitesh S. B., & Garvit C., *A Comprehensive Study of Intrusion Detection and Prevention Systems* , Wireless Communication Security, vol. 7, 2022, pp. 115-142, doi: https://doi.org/10.1002/9781119777465.ch7.

Byrnes J., Hoang T., Mehta N. N. & Cheng Y., *A Modern Implementation of System Call Sequence Based Host-based Intrusion Detection Systems*, in *Proc. Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, Oct. 2020, pp. 218-225.

Calugar A. N., Meng W. & Zhang H., *Towards Artificial Neural Network Based Intrusion Detection with Enhanced Hyperparameter Tuning,* GLOBECOM 2022 - 2022 IEEE Global Communications Conference, 2022, pp. 2627-2632, doi: 10.1109/GLOBECOM48099.2022.10000809.

Cao B., Chenghai L., Yafei S., Yueyi Q. & Chen C., *Network Intrusion Detection Model Based on CNN and GRU*, Applied Sciences, vol. 12, 2022, 4184, https://doi.org/10.3390/app12094184.

Chen L., Gao S., & Liu B., *An Improved Density Peaks Clustering Algorithm Based on Grid Screening and Mutual Neighborhood Degree for Network Anomaly Detection*. Scientist Report, vol. 12, 2022, 1409. https://doi.org/10.1038/s41598-021-02038-z.

Chen L., Kuang X., Xu A., Suo S., & Yang Y., *A Novel Network Intrusion Detection System Based on CNN.* In Proceedings of the 2020 Eighth International Conference on Advanced Cloud and Big Data (CBD), Taiyuan, China, 2020, pp. 243–247. https://doi.org/10.1109/CBD51900.2020.00051.

Chen Y. & Yuan F., *Dynamic Detection of Malicious Intrusion in Wireless Network Based on Improved Random Forest Algorithm.* In Proceedings of the 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), 2022 pp. 27-32. https://doi.org/10.1109/IPEC54454.2022.9777557.

Chen Y., Lin Q., Wei W., Ji J., Ka-Chun W. & Carlos A. C., *Intrusion Detection Using Multi-Objective Evolutionary Convolutional Neural Network for Internet of Things in Fog*

Computing, Knowledge-Based Systems, vol. 244, 2022, pp. 108505, https://doi.org/10.1016/j.knosys.2022.108505.

David J. & Thomas C., *Efficient DDoS flood attack detection using dynamic thresholding on flow-based network traffic*, Computers & Security, vol. 82, no. 284, 2019, pp. 0167-4048, doi: https://doi.org/10.1016/j.cose.2019.01.002.

Denning D. E., *An Intrusion-detection Model*, IEEE Transactions on Software Engineering, no. 2, 1987, pp. 222–232.

Disha R.A., & Waheed S., *Performance Analysis of Machine Learning Models for Intrusion Detection System Using Gini Impurity-based Weighted Random Forest (GIWRF) Feature Selection Technique*, Cybersecurity, vol. 5, no, 1, 2022, https://doi.org/10.1186/s42400-021-00103-8.

Dixit U., Bhatia S. & Bhatia P., *Utilizing ML and DL Algorithms for Alert Classification in Intrusion Detection and Prevention Systems: A Detailed Review*, *2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 2022, pp. 1199-1205, doi: 10.1109/ICACITE53722.2022.9823605.

Dwivedi S., Vardhan M., Tripathi S. & Shukla A. K., *Implementation of Adaptive Scheme in Evolutionary Technique for Anomaly-based Intrusion Detection*, Evolutionary Intelligence, vol. 13, no. 1, 2020, pp. 103-117.

Effendy, D. A., Kusrini, K., & Sudarmawan, S., *Classification of Intrusion Detection System (IDS) based on Computer Network*, 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), 2017, pp. 90-94, doi: 10.1109/ICITISEE.2017.8285566.

El Kafhali S. & El Mir I., *Security Threats, Defense Mechanisms, Challenges, and Future Directions in Cloud Computing*, Archives of Computational Methods in Engineering, vol. 29, no. 223, 2022, doi: 10.1007/s11831-021-09573-y.

Elhefnawy R., Abounaser H., & Badr A., *A Hybrid Nested Genetic-Fuzzy Algorithm Framework for Intrusion Detection and Attacks.* IEEE Access, vol. 8, 2020, pp. 98218–98233. https://doi.org/10.1109/ACCESS.2020.2996226.

Elmasry W., Akbulut A., & Zaim A. H., *Evolving Deep Learning Architectures for Network Intrusion Detection Using a Double PSO Metaheuristic*. Computer Networks, vol. 168, 2020, 107042.

Emad-ul-Haq Q., Muhammad I., Noman H., Muhammad S. & Imran R., *An Intelligent and Efficient Network Intrusion Detection System Using Deep Learning*, Computers and Electrical Engineering, Vol. 99, 2022, https://doi.org/10.1016/j.compeleceng.2022.107764.

Ganeshan R, kolli .C.S., kumar M. & Daniya T., *A Systematic Review on Anomaly Based Intrusion Detection System* , IOP Conference Series: Materials Science and Engineering, vol. 981, no. 2, 2020, doi: 10.1088/1757-899X/981/2/022010.

Garg S., Kaur K., Kumar N., Kaddoum G., Zomaya A. Y., & Ranjan R., *A Hybrid Deep Learning-Based Model for Anomaly Detection in Cloud Datacenter Networks*, IEEE Transactions on Network and Service Management, vol. 16, no. 3, 2019, pp. 924-935, doi: 10.1109/TNSM.2019.2927886.

Gautam S., Henry A., Zuhair M., Rashid M., Javed A. R. & Maddikunta P. K. R., *A Composite Approach of Intrusion Detection Systems: Hybrid RNN and Correlation-Based Feature Optimization*. Electronics, vol. 11, 2022, 3529. https://doi.org/10.3390/electronics11213529.

Ghanshala K. K., Mishra P., Joshi R. C. & Sharma S., *BNID: A Behavior-based Network Intrusion Detection at Network-layer in Cloud Environment*, *Proc. First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, 2018, pp. 100-105.

Giuseppina A., Annalisa A., Paolo .C. F., Donato M., Gennaro V., *ROULETTE: A neural attention multi-output model for explainable Network Intrusion Detection*, Expert Systems with Applications, vol. 201, no. 117144, 2022, https://doi.org/10.1016/j.eswa.2022.117144.

Gu J. & Lu S., *An Effective Intrusion Detection Approach Using SVM with Naïve Bayes Feature Embedding.* Computers & Security. Vol. 103, 2021, 102158. https://doi.org/10.1016/j.cose.2020.102158.

Gunduz M. Z. & Das R., *Cyber-security on Smart Grid: Threats and Potential Solutions*, Computer Network, vol. 169, 2020, https://doi.org/10.1016/j.comnet.2019.107094.

Gupta D., Bhatt S., Gupta M., Kayode O., & Tosun A. S., *Access Control Model for Google Cloud IoT*, *Proc.* IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), 2020, pp. 198-208.

Gupta S. K., Tripathi M. & Grover J., *Hybrid Optimization and Deep Learning Based Intrusion Detection System*, Computers and Electrical Engineering, vol. 100, no. 107876, 2022, pp. 0045 – 7906, doi: https://doi.org/10.1016/j.compeleceng.2022.107876.

Hadem P., Saikia D. K. & Moulik S., *An SDN-based Intrusion Detection System Using SVM with Selective Logging for IP Traceback*, Computer & Network, vol. 191, 2021, Art. no. 108015. https://doi.org/10.1016/j.comnet.2021.108015.

Halbouni A., Gunawan T. S., Habaebi M. H., Halbouni M., Kartiwi M. & Ahmad R., *Machine Learning and Deep Learning Approaches for CyberSecurity: A Review*, IEEE Access, vol. 10, 2022, pp. 19572-19585, doi: 10.1109/ACCESS.2022.3151248.

Heidari A., Jabraeil J. & Mohammad A., *Internet of Things Intrusion Detection Systems: A Comprehensive Review and Future Directions*, Cluster Computing, 2022, 10.1007/s10586-022-03776-z.

Heidari A., & Jabraeil J. M.A., *Internet of Things Intrusion Detection Systems: A Comprehensive Review and Future Directions*. Cluster Computing**,** vol. 26, 2023, 3753–3780. https://doi.org/10.1007/s10586-022-03776-z.

Hochberg J., Jackson K., Stallings C., McClary J., DuBois D. & Ford J., *Nadir: An Automated System for Detecting Network Intrusion and Misuse*, Computers & Security, vol. 12, no. 3, 1993, pp. 235–248.

Horchulhack P., Viegas E. K. & Santin A. O., *Toward Feasible Machine Learning Model Updates in Network-based Intrusion Detection*, Computer Networks, vol. 202, 2022, pp. 1389-1286, https://doi.org/10.1016/j.comnet.2021.108618.

Ibrahim W. N. H., Anuar S., Selamat A., Krejcar O., Crespo R. G., & Fujita H., *Multilayer Framework for Botnet Detection Using Machine Learning Algorithms*, IEEE, vol. 9, 2021, pp. 48753-48768, doi: 10.1109/ACCESS.2021.3060778.

Ioannis M., Ioannis R., Ioannis G., George K., Anastasios D. & Nikolaos D., *Multiclass Confusion Matrix Reduction Method and Its Application on Net Promoter Score Classification Problem,* School of Rural and Surveying Engineering, National Technical University of Athens, 9th, Heroon Polytechniou Str., 15773 Athens, Greece: An Article (PETRA Conference Series, vol. 5, 2018, pp. 1-23.

Jabez J. & Muthukumar B., *Intrusion Detection System (IDS): Anomaly Detection Using Outlier Detection Approach*. Procedia Computer Science. International Conference on Intelligent Computing, Communication & Convergence (ICCC-2014) Conference Organized by Interscience Institute of Management and Technology**,** vol. 48, 2015, pp. 338–346. https://doi.org/10.1016/j.procs.2015.04.191.

Janabi A. H., Kanakis T. & Johnson M., *Convolutional Neural Network Based Algorithm for Early Warning Proactive System Security in Software Defined Networks*, *in* IEEE Access, vol. 10, 2022, pp. 14301-14310, doi: 10.1109/ACCESS.2022.3148134.

Jin S., Diao R., & Shen Q., *Backward Fuzzy Interpolation and Extrapolation With Multiple Multi-Antecedent Rules*. In Proceedings of IEEE International Conference on Fuzzy Systems, 2012, pp. 1170–1177.

Kanimozhi P. & Victoire T. A. A, *Oppositional Tunicate Fuzzy C-means Algorithm and Logistic Regression for Intrusion Detection on Cloud*. Concurrency and Computation Practice and Experience., vol. 34, 2022, e6624. https://doi.org/10.1002/cpe.6624.

Kanimozhi P., & Aruldoss V. T., *Oppositional Tunicate Fuzzy C-means Algorithm and Logistic Regression for Intrusion Detection on Cloud*, Concurrency and Computation Practice and Experience, vol. 34, no. 4, 2022, e6624. doi:10.1002/cpe.6624.

Khalil A. A. & Rahman M. A., *Adaptive Neuro-Fuzzy Inference System-based Lightweight Intrusion Detection System for UAVs*," 2023 IEEE 48th Conference on Local Computer Networks (LCN), 2023, pp. 1-9, doi: 10.1109/LCN58197.2023.10223340.

Khammassi C. & Krichen S., *A NSGA2-LR Wrapper Approach for Feature Selection in Network Intrusion Detection*, Computer Networks, vol. 172, 2020, pp. 1389-1286, https://doi.org/10.1016/j.comnet.2020.107183.

Khan F. A., Gumaei A., Derhab A., & Hussain A., *A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection.* IEEE Access, vol. 7, 2019, pp. 30373–30385. https://doi.org/10.1109/ACCESS.2019.2899721.

Khraisat A. & Alazab A., *A Critical Review of Intrusion Detection Systems in the Internet of Things: Techniques, Deployment Strategy, Validation Strategy, Attacks, Public Datasets and Challenges*, Cybersecurity, vol. 4, no. 1, 2021, 10.1186/s42400-021-00077-7.

Kilincer I. F., Ertam F. & Sengur A., *Machine Learning Methods for Cyber Security Intrusion Detection: Datasets and Comparative Study*, Computer Networks, Vol. 188, ISSN 1389-1286, 2021, https://doi.org/10.1016/j.comnet.2021.107840.

Kumar S., Gupta S. & Arora S., *Research Trends in Network-Based Intrusion Detection Systems: A Review*, *IEEE*, vol. 9, 2021, pp. 157761-157779, doi: 10.1109/ACCESS.2021.3129775.

Kumar V., Sinha D., Das A. K, Subhash C. P. & Radha T. G., *An Integrated Rule Based Intrusion Detection System: Analysis on UNSW-NB15 Data Set and the Real Time Online Dataset*, Cluster Computing, vol. 23, 2020, pp 1397–1418, https://doi.org/10.1007/s10586-019-03008-x.

Leon M., Markovic T. & Punnekkat S., *Comparative Evaluation of Machine Learning Algorithms for Network Intrusion Detection and Attack Classification*, 2022 International Joint Conference on Neural Networks (IJCNN), 2022, doi: 10.1109/IJCNN55064.2022.9892293.

Li Y,.Xu Y, Liu Z., Hou H., Zheng Y., Xin Y., *Robust Detection for Network Intrusion of Industrial IoT Based on Multi-CNN Fusion*. Measurement, vol. 154, 2020, 107450.

Liang D., Liu Q., Zhao B., Zhu Z. & Liu D., *A Clustering-SVM Ensemble Method for Intrusion Detection System.* In Proceedings of the 2019 8th International Symposium on Next Generation Electronics (ISNE), 2019, pp. 1–3. https://doi.org/10.1109/ISNE.2019.8896514.

Lin Y., Zhang Y., & Ou Y., *The Design and Implementation of Host-based Intrusion Detection System.* In Proceedings of the 3rd International Symposium on Intelligent Information Technology and Security Informatics, 2010, pp. 595–598.

Long J., Fang F. & Luo H., *A Survey of Machine Learning-based IoT Intrusion Detection Techniques*, IEEE 6th International Conference on Smart Cloud (SmartCloud), 2021, pp. 7-12, doi: 10.1109/SmartCloud52277.2021.00009.

Malek Z. S., Trivedi B. & Shah A., *User Behavior Pattern –Signature Based Intrusion Detection*, in *Proc. Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2020, pp. 549-552, doi. 10.1109/WorldS450073.2020.9210368.

Mani S., Sundan B., Thangasamy A., & Govindaraj L, *A New Intrusion Detection and Prevention System Using a Hybrid Deep Neural Network in Cloud Environment*. Computer Networks, Big Data and IoT. Lecture Notes on Data Engineering and Communications Technologies, vol 117, 2022, https://doi.org/10.1007/978-981-19-0898-9_73.

Marir N., Wang H., Feng G., Li B. & Jia M., *Distributed Abnormal Behavior Detection Approach Based on Deep Belief Network and Ensemble SVM Using Spark*. IEEE Access, vol. 6, 2018, pp. 59657–59671. https://doi.org/10.1109/ACCESS.2018.2875045.

Martins I., Resende J. S., Sousa P. R., Silva S., Antunes L., & Gama J., *Host-based IDS: A Review and Open Issues of an Anomaly Detection System in IoT*, Future Generation Computer Systems, vol. 133, no. 95, 2022, 0167-739X, https://doi.org/10.1016/j.future.2022.03.001.

Mayuranathan M., Saravanan S.K., Muthusenthil B. & Samydurai, A., *An Efficient Optimal Security System for Intrusion Detection in Cloud Computing Environment Using Hybrid Deep Learning Technique*, Advances in Engineering Software, vol. 173, no. 103236, 2022, pp. 0965-9978, doi: https://doi.org/10.1016/j.advengsoft.2022.103236.

Mhawi D.N, Aldallal A. & Hassan S., *Advanced Feature-Selection-Based Hybrid Ensemble Learning Algorithms for Network Intrusion Detection Systems*. Symmetry, vol. 14, 2022, 1461. https://doi.org/10.3390/sym14071461.

Muhammad S., Zhaoquan G., Omar C., Wajdi A., & Habib H., *The Rise of "Internet of Things": Review and Open Research Issues Related to Detection and Prevention of IoT-Based Security Attacks*, Hindawi Wireless Communications and Mobile Computing, vol. 2022, 2022, pp. 12, https://doi.org/10.1155/2022/8669348.

Murat Ö., Cihan K., & Ahmet Z., *Distributed Intrusion Detection Systems: A Survey*, The Academic Perspective Procedia, vol. 2, no. 2, 2019, pp. 400 – 407, https://doi.org/10.33793/acperpro.02.03.18.

Muzafar S. & Jhanjhi N., *DDoS Attacks on Software Defined Network: Challenges and Issues,* International Conference on Business Analytics for Technology and Security (ICBATS), 2022, pp. 1-6, doi: 10.1109/ICBATS54253.2022.9780662.

Omar A. *A Feature Selection Model for Network Intrusion Detection System Based on PSO, GWO, FFA and GA Algorithms Symmetry*, vol. 12, no. 6, 2020, https://doi.org/10.3390/sym12061046.

Ozkan-Okay M., Samet R., Aslan Ö. & Gupta D., *A Comprehensive Systematic Literature Review on Intrusion Detection Systems*, IEEE, vol. 9, pp. 157727-157760, 2021, pp. 3-4, doi: 10.1109.

Park D., Kim S., Kwon H., Shin D. & Shin D., *Host-based Intrusion Detection Model Using Siamese Network*, IEEE Access, vol. 9, 2021, pp. 76614-76623.

Paxson V., *Bro: A System for Detecting Network Intruders in Real-time*, Computer Networks, vol. 31, no. 23, 1999, pp. 2435–2463.

Pradhan, M., Nayak, C. K., & Pradhan, S. K., *Intrusion Detection System (IDS) and Their Types In Securing the Internet of Things: Concepts, Methodologies, Tools, and Applications* (IGI Global, 2020), 2020, pp. 481-497, https://doi.org/10.4018/978-1-5225-9866-4.ch026.

Prasad S., Alamuru S., Arun S & Alamuru S., *Intrusion Detection and Prevention Systems in Smart Environments – A Survey*, *2022 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), 2022,* pp. 1-5, doi: 10.1109/SMARTGENCON56628.2022.10084309.

Prashanth S.K., Shitharth S., Praveen K. B., Subedha V. & Sangeetha K., *Optimal Feature Selection Based on Evolutionary Algorithm for Intrusion Detection*. SN Computer Science. Vol. 3, no. 439, 2022, https://doi.org/10.1007/s42979-022-01325-4.

Protic D., *Review of KDD Cup '99, NSL-KDD and Kyoto 2006 and datasets.* Vojnoteh. Glas. vol. 66, 2018, pp. 580–596. https://doi.org/10.5937/vojtehg66-16670.

Qiu W., Ma Y., Chen X., Yu H. & Chen L., *Hybrid Intrusion Detection System Based on Dempster-Shafer Evidence Theory*, Computers & Security, vol. 117, 2022, https://doi.org/10.1016/j.cose.2022.102709.

Quincozes S. E., Albuquerque C., Passos D., & Mossé D., *A Survey on Intrusion Detection and Prevention Systems in Digital Substations*, Computer Networks, 184, 2021, https://doi.org/10.1016/j.comnet.2020.107679.

Raja, N.M. & Vegad, S., *An Empirical Study for the Traffic Flow Rate Prediction-based Anomaly Detection in Software-defined Networking: A Challenging Overview.* Social Network Analysis and Mining, vol. 13, 2023, https://doi.org/10.1007/s13278-023-01057-0.

Rawat S., Srinivasan A., Ravi V. & Ghosh U., *Intrusion Detection Systems Using Classical Machine Learning Techniques vs Integrated Unsupervised Feature Learning and Deep Neural Network,* Internet Technology Letters. Vol. 5, no.1, 2022, pp. e232, https://doi.org/10.1002/itl2.232

Sharma S., Nand P., & Sharma P., *Intrusion Detection and Prevention Systems Using SNORT*, Advances in Data Science and Management, vol 86, 2022, doi: org/10.1007/978-981-16-5685-9_46.

Sharon A., Mohanraj P., Abraham T. E., Sundan B. & Thangasamy A., *An Intelligent Intrusion Detection System Using Hybrid Deep Learning Approaches in Cloud Environment*. Computer, Communication, and Signal Processing. (ICCCSP), vol. 651, 2022. https://doi.org/10.1007/978-3-031-11633-9_20.

Smith A., Ramotsoela T. D. & Hancke G. P., *Behavioural Intrusion Detection for Wireless Sensor Networks*, *2021* IEEE 30th International Symposium on Industrial Electronics (ISIE)*, 2021*, pp. 01-06, doi: 10.1109/ISIE45552.2021.9576349.

Song H. M, Woo J., & . Kim H. K, *In-vehicle Network Intrusion Detection Using Deep Convolutional Neural Network*. Vehicular Communications, vol. 21, 2020, 100198.

Stavroulakis P., & Stamp M., *Handbook of Information and Communication Security*. New York: Springer. 2010.

Subba B., Biswas S. & Karmakar S., *Host Based Intrusion Detection System using Frequency Analysis of N-gram Terms*, *in Proc.* TENCON 2017 - 2017 IEEE Region 10 Conference*,* Nov. 2017, pp. 2006-2011.

Sundararajan V., & Dietz E., *Centralized Hierarchical Cybersecurity Monitoring Towards Securing the Defense Industrial Base Supply Chain*. Series Of Research Network, 2023, p. 8, http://dx.doi.org/10.2139/ssrn.4603578.

Tavallaee M., Bagheri E., Lu W. & Ghorbani A. A., *A Detailed Analysis of the KDD CUP 99 Data Set, 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1-6, doi: 10.1109/CISDA.2009.5356528.

Thakkar A., & Lohiya R., *Attack Classification Using Feature Selection Techniques: A Comparative Study*, Journal of Ambient Intelligence and Humanized Computing, vol. 12, 2021, pp. 1249–1266, https://doi.org/10.1007/s12652-020-02167-9.

Tuan-Hong C. & Iftekhar S., *Evaluation of Machine Learning Algorithms in Network-Based Intrusion Detection Using Progressive Dataset*, Symmetry, Vol. 15, 2023, pp. 1251, 10.3390/sym15061251.

Vinolia A., Kanya N. & Rajavarman V. N., *Machine Learning and Deep Learning-based Intrusion Detection in Cloud Environment: A Review*, 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), 2023, pp. 952-960, doi: 10.1109/ICSSIT55814.2023.10060868.

W. A. H. M. Ghanem, *Cyber Intrusion Detection System Based on a Multiobjective Binary Bat Algorithm for Feature Selection and Enhanced Bat Algorithm for Parameter Optimization in Neural Networks*, in IEEE Access**,** vol. 10, pp. 76318-76339, 2022, doi: 10.1109/ACCESS.2022.3192472.

Wintrode J. & DeTienne D., *Adaptive Encrypted Traffic Characterization via Deep Representation Learning*, 2022 Intermountain Engineering, Technology and Computing (IETC), 2022, pp. 1-6, doi: 10.1109/IETC54973.2022.9796734.

Wisanwanichthan T., & Thammawichai M., *A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM*. IEEE Access, vol. PP, 2021, pp. 1 – 1, https://doi.org/10.1109/ACCESS.2021.3118573.

Yan B., & Han G., *Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System*. IEEE Access. Vol. 6, 2018, pp. 41238–41248. https://doi.org/10.1109/ACCESS.2018.2858277.

Yang S., *Research on Network Malicious Behavior Analysis Based on Deep Learning*, in *Proc. I**EEE 5th* Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2021, pp. 2609-2612.

Yang Z., Liu X., Li T., Wu D., Wang J., Zhao Y. & Han H., *A Systematic Literature Review of Methods and Datasets for Anomaly-based Network Intrusion Detection*, Computers & Security, vol. 116, no. 102675, 2022, pp. 0167-4048, doi. https://doi.org/10.1016/j.cose.2022.102675.

Yu Y., & Bian N., *An Intrusion Detection Method Using Few-Shot Learning*. IEEE Access, vol. 8, 2020, pp. 49730–49740. https://doi.org/10.1109/ACCESS.2020.2980136.

Zheng L., Zhang J., Lin F. & Wang X., *Feature-Fusion-Based Abnormal-Behavior-Detection Method in Virtualization Environment.* Electronics, vol. 12, 2023, 3386. https://doi.org/10.3390/electronics12163386.

Zhongjun Y., Zhi L., Xuejun Z., & Guogang W., *An Optimized Adaptive Ensemble Model with Feature Selection for Network Intrusion Detection*, Concurrency and Computation: Practice and Experience, vol. 35, 2023, https://doi.org/10.1002/cpe.7529.

Zhongyun T., Haiyang H. & Chonghuan X., *A Federated Learning Method for Network Intrusion Detection*, Concurrency and Computation: Practice and Experience, vol. 24, 2021, https://doi.org/10.1002/cpe.6812.

Journals

Alaa O. & Mohammed A., *An Efficient Approach towards Network Routing using Genetic Algorithm*, **International Journal of Computers Communications and Control**, vol. 17, 2022, https://doi.org/10.15837/ijccc.2022.5.4815.

Alkasassbeh M. & Al-Haj B. S., *Intrusion Detection Systems: A State-of-the-Art Taxonomy and Survey*. **Arabian Journal for Science and Engineering,** vol. 48, 2023, pp.10021–10064 https://doi.org/10.1007/s13369-022-07412-1.

Arunkumar M. & Kumar, K.A., GOSVM: *Gannet Optimization Based Support Vector Machine for Malicious Attack Detection in Cloud Environment.* **International Journal of Information Technology**, vol. 15, 2023, pp. 1653–1660. https://doi.org/10.1007/s41870-023-01192-z.

Bedi P., Gupta N. & Jindal V., *I-SiamIDS: An Improved Siam-IDS for Handling Class Imbalance in Network-based Intrusion Detection Systems,* **International Journal of Speech Technology**, vol. 51, no. 2, 2021, pp. 1133-1151.

Besharati, E., Naderan, M. & Namjoo, E., LR-HIDS: Logistic Regression Host-based Intrusion Detection System for Cloud Environments, **Journal of Ambient Intelligence and Humanized Computing**, vol. 10, 2019, pp. 3669–3692. doi:org/10.1007/s12652-018-1093-8.

Boopathi M. & Seetha, R. *Accurate Detection of Multi-layer Packet Dropping Attacks Using Distributed Mobile Agents in MANET*, **Journal of Physics: Conference Series**, vol. 1979, no. 1, 2021, pp. 1742-6596, doi: 10.1088/1742-6596/1979/1/012040.

Choudhary S. & Kesswani N., *A Hybrid Classification Approach for Intrusion Detection in IoT Network*, **Journal of Scientific & Industrial Research**, vol. 80, no. 9, 2021, pp. 809-816, https://doi.org/10.1155/2022/4553502

Chuan F., Pengchao H., Xu Z., Bowen Y. & Yejun L. G., *Computation Offloading in Mobile Edge Computing Networks: A Survey*, **Journal of Network and Computer Applications**, vol. 202, 2022, doi.org/10.1016/j.jnca.2022.103366.

Doaa M. & Osama I., *Enhancement of an IoT Hybrid Intrusion Detection System Based on Fog-to-cloud Computing*, **Journal of Cloud Computing**, vol. 12, no. 1, 2023, 41, https://doi.org/10.1186/s13677-023-00420-y.

Geeta S. & Neelu K., *A Survey of Intrusion Detection From the Perspective of Intrusion Datasets and Machine Learning Techniques,* **International Journal of Computers and Applications**, vol. 44, 2022, 659-669, DOI: 10.1080/1206212X.2021.1885150.

Ibrahim A., Abdelghani D., Samia A. C., Mohammed, A. A. A, & Mohamed A. E., *Feature Selection Model Based on Gorilla Troops Optimizer for Intrusion Detection Systems*. **Journal of Sensors**, vol. 2022, 2022, pp. 1-12, https://doi.org/10.1155/2022/6131463.

Kesswani N., & Agarwal B., *SmartGuard: An IoT-based Intrusion Detection System for Smart Homes.* **International Journal of Intelligent Information and Database Systems**, vol. 13(1), 2020, pp. 61–71.

Lata S. & Singh D., *Intrusion Detection System in Cloud Environment: Literature Survey & Future Research Directions*, **International Journal of Information Management Data Insights**, vol. 2, no. 2, 2022, pp. 2667-0968, doi: https://doi.org/10.1016/j.jjimei.2022.100134.

Li H. W., Wu Y. S., & Huang Y., *On the Feasibility of Anomaly Detection with Fine-Grained Program Tracing Events*. **Journal of Network and Systems Management**, vol. 30, 2022, pp.28,. https://doi.org/10.1007/s10922-021-09635-3.

Mazini M., Shirazi B. & Mahdavi I., *Anomaly Network-based Intrusion Detection System Using a Reliable Hybrid Artificial Bee Colony and AdaBoost Algorithms*, **Journal of King Saud University-Computer and Information Sciences**, vol. 31, no. 4, 2019, pp. 541-553,

Meftah S., Rachidi T. & Assem N., *Network Based Intrusion Detection Using the UNSW-NB15 Dataset*, **International Journal of Computing and Digital Systems**., vol. 8, no. 5, 2019, pp. 478-487.

Monis T., & Mohd S., *A Review on Intrusion Detection in Cloud Computing*. **International Journal of Engineering and Management Research**, vol. *13*(2), 2023, 207–215. https://doi.org/10.31033/ijemr.13.2.35l.

Mousavi S.M., Majidnezhad V. & Naghipour A., *A New Intelligent Intrusion Detector Based on Ensemble of Decision Trees,* **Journal of Ambient Intelligence and Humanized Computing**, vol. 13, 2022, pp. 3347–3359, https://doi.org/10.1007/s12652-019-01596-5.

Mughal A. A., *Well-Architected Wireless Network Security*, **Journal of Humanities and Applied Science Research,** vol. 5, no. 1, 2022, doi: https://journals.sagescience.org/index.php/JHASR/article/view/52.

Otoum Y. & Nayak A., *AS-IDS: Anomaly and Signature Based IDS for the Internet of Things*, **Journal of Network and Systems Management**, vol. 29, no. 23, 2021, pp. 1-26, , doi: https://doi.org/10.1007/s10922-021-09589-6.

Ouarda L., Malika B., & Brahim B., *Towards a Better Similarity Algorithm for Host-based Intrusion Detection System*. **Journal of Intelligent Systems**, vol. *32*, 2023, pp. 20220259. https://doi.org/10.1515/jisys-2022-0259.

Pan J. S., Fan F., Chu S.C., Zhao H. & Liu G. A., *Lightweight Intelligent Intrusion Detection Model for Wireless Sensor Networks*. **Security and Communication Networks**, vol. 2021, 2021, pp. 1–15. https://doi.org/10.1155/2021/5540895.

Parganiha V., Shukla S. P., & Sharma L. K., *Cloud Intrusion Detection Model Based on Deep Belief Network and Grasshopper Optimization*. **International Journal of Ambient Computing and Intelligence (IJACI)**, vol. 13(1), 2022, pp. 1-24. http://doi.org/10.4018/IJACI.293123.

Sicato, J. C. S., Singh, S. K., Rathore, S., & Park, J. H, *A Comprehensive Analysis of Intrusion Detection System for IoT Environment*. (**Journal of Information Processing Systems**, 2020), 16(4), 2020, pp. 975-990.

Suman L. & Dheerendra S., *Intrusion Detection System in Cloud Environment: Literature Survey & Future Research Directions*, **International Journal of Information Management Data Insights**, vol. 2, 2022, https://doi.org/10.1016/j.jjimei.2022.100134.

Yuhua Y., Jang-Jaccard J., Wen X. , Amardeep S., Jinting Z., Fariza S. & Jin K., *IGRF-RFE: A Hybrid Feature Selection Method for MLP-based Network Intrusion Detection on UNSW-NB15 Dataset*. **Journal of Big Data,** vol. 10, 2023, https://doi.org/10.1186/s40537-023-00694-8

Appendix

1.

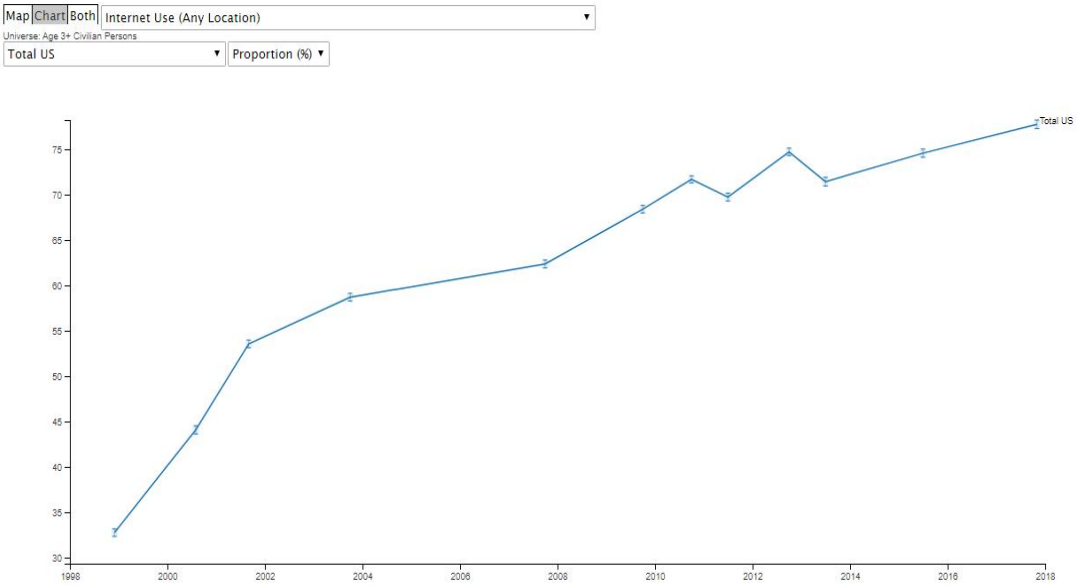


Figure 1.1 Internet Use in the U.S

(Source: http://www.internetworldstats.com)

Source Code

```
import numpy as np
from flask import Flask, request, jsonify, render_template
import joblib
app = Flask(__name__)
model = joblib.load('model.pkl')
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict',methods=['POST'])
def predict():
    int_features = [float(x) for x in request.form.values()]
    if int_features[0]==0:
        f_features=[0,0,0]+int_features[1:]
    elif int_features[0]==1:
        f_features=[1,0,0]+int_features[1:]
    elif int_features[0]==2:
        f_features=[0,1,0]+int_features[1:]
    else:
        f_features=[0,0,1]+int_features[1:]
    if f_features[6]==0:
        fn_features=f_features[:6]+[0,0]+f_features[7:]
    elif f_features[6]==1:
        fn_features=f_features[:6]+[1,0]+f_features[7:]
    else:
        fn_features=f_features[:6]+[0,1]+f_features[7:]
    final_features = [np.array(fn_features)]
    predict = model.predict(final_features)
    if predict==0:
        output='Normal'
    elif predict==1:
        output='DOS'
```

```
    elif predict==2:
        output='PROBE'
    elif predict==3:
        output='R2L'
    else:
        output='U2R'
    return render_template('index.html', output=output)

@app.route('/results',methods=['POST'])
def results():
    data = request.get_json(force=True)
    predict = model.predict([np.array(list(data.values()))])
    if predict==0:
        output='Normal'
    elif predict==1:
        output='DOS'
    elif predict==2:
        output='PROBE'
    elif predict==3:
        output='R2L'
    else:
        output='U2R'
    return jsonify(output)
if __name__ == "__main__":
    app.run()

"""Network Intrusion Detection System.ipynb
# Commented out IPython magic to ensure Python compatibility.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# %matplotlib inline
```

```
import itertools
import seaborn as sns
import pandas_profiling
import statsmodels.formula.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrices
from sklearn import datasets
from sklearn.feature_selection import RFE
import sklearn.metrics as metrics
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_classif, mutual_info_classif

train=pd.read_csv('NSL_Dataset\Train.txt',sep=',')
test=pd.read_csv('NSL_Dataset\Test.txt',sep=',')
train.head()
columns=["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_frag
ment","urgent","hot","num_failed_logins","logged_in","num_compromised","root_shell","su_atte
mpted","num_root","num_file_creations","num_shells","num_access_files","num_outbound_cmd
s","is_host_login","is_guest_login","count","srv_count","serror_rate","srv_serror_rate","rerror_ra
te","srv_rerror_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_
host_srv_count","dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rat
e","dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate","dst_host_rerr
or_rate", "dst_host_srv_rerror_rate","attack","last_flag"]
len(columns)
train.columns=columns
test.columns=columns
train.head()
test.head()
train.info()
test.info()
train.describe().T
```

"""In attack_class normal means 0, DOS means 1, PROBE means 2, R2L means 3 and U2R means 4."""

```
train.loc[train.attack=='normal','attack_class']=0
train.loc[(train.attack=='back') | (train.attack=='land') | (train.attack=='pod') | (train.attack=='neptune') |
    (train.attack=='smurf') | (train.attack=='teardrop') | (train.attack=='apache2') | (train.attack=='udpstorm') |
    (train.attack=='processtable') | (train.attack=='worm') | (train.attack=='mailbomb'),'attack_class']=1

train.loc[(train.attack=='satan') | (train.attack=='ipsweep') | (train.attack=='nmap') | (train.attack=='portsweep') |
    (train.attack=='mscan') | (train.attack=='saint'),'attack_class']=2

train.loc[(train.attack=='guess_passwd') | (train.attack=='ftp_write') | (train.attack=='imap') | (train.attack=='phf') |
    (train.attack=='multihop') | (train.attack=='warezmaster') | (train.attack=='warezclient') | (train.attack=='spy') |
    (train.attack=='xlock') | (train.attack=='xsnoop') | (train.attack=='snmpguess') | (train.attack=='snmpgetattack') |
    (train.attack=='httptunnel') | (train.attack=='sendmail') | (train.attack=='named'),'attack_class']=3

train.loc[(train.attack=='buffer_overflow') | (train.attack=='loadmodule') | (train.attack=='rootkit') | (train.attack=='perl') |
    (train.attack=='sqlattack') | (train.attack=='xterm') | (train.attack=='ps'),'attack_class']=4

test.loc[test.attack=='normal','attack_class']=0
test.loc[(test.attack=='back') | (test.attack=='land') | (test.attack=='pod') | (test.attack=='neptune') |
    (test.attack=='smurf') | (test.attack=='teardrop') | (test.attack=='apache2') | (test.attack=='udpstorm') |
```

```
(test.attack=='processtable') | (test.attack=='worm') |
(test.attack=='mailbomb'),'attack_class']=1
test.loc[(test.attack=='satan') | (test.attack=='ipsweep') | (test.attack=='nmap') |
(test.attack=='portsweep') |
(test.attack=='mscan') | (test.attack=='saint'),'attack_class']=2
test.loc[(test.attack=='guess_passwd') | (test.attack=='ftp_write') | (test.attack=='imap') |
(test.attack=='phf') |
(test.attack=='multihop') | (test.attack=='warezmaster') | (test.attack=='warezclient') |
(test.attack=='spy') |
(test.attack=='xlock') | (test.attack=='xsnoop') | (test.attack=='snmpguess') |
(test.attack=='snmpgetattack') |
(test.attack=='httptunnel') | (test.attack=='sendmail') |
(test.attack=='named'),'attack_class']=3
test.loc[(test.attack=='buffer_overflow') | (test.attack=='loadmodule') | (test.attack=='rootkit') |
(test.attack=='perl') |
(test.attack=='sqlattack') | (test.attack=='xterm') | (test.attack=='ps'),'attack_class']=4

train.head()
train.shape

output=pandas_profiling.ProfileReport(train)
output

"""### Exporting pandas profiling output to html file"""
output.to_file('pandas_profiling.html')

"""### Basic Exploratory Analysis"""
# Protocol type distribution
plt.figure(figsize=(6,3))
sns.countplot(x="protocol_type", data=train)
plt.show()

# service distribution
```

```
plt.figure(figsize=(6,10))
sns.countplot(y="service", data=train)
plt.show()

# flag distribution
plt.figure(figsize=(6,3))
sns.countplot(x="flag", data=train)
plt.show()

# attack distribution
plt.figure(figsize=(6,4))
sns.countplot(y="attack", data=train)
plt.show()

# attack class distribution
plt.figure(figsize=(6,3))
sns.countplot(x="attack_class", data=train)
plt.show()

"""#### identifying relationships (between Y & numerical independent variables by comparing means)"""
train.groupby('attack_class').mean().T

"""##### Observations:
- The length of time duration of connection for attack is higher than  normal.
- Wrong fragments in the connection is only present in attack.
- Number of outbound commands in an ftp session  are 0 in both normal and attack.
"""
numeric_var_names=[key for key in dict(train.dtypes) if dict(train.dtypes)[key] in ['float64', 'int64', 'float32', 'int32']]
cat_var_names=[key for key in dict(train.dtypes) if dict(train.dtypes)[key] in ['object', 'O']]

numeric_var_names
```

```
cat_var_names

train_num=train[numeric_var_names]
test_num=test[numeric_var_names]
train_num.head(5)

train_cat=train[cat_var_names]
test_cat=test[cat_var_names]
train_cat.head(5)

"""### Data Audit Report"""
# Creating Data audit Report
def var_summary(x):
    return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.median(), x.std(), x.var(),
x.min(), x.dropna().quantile(0.01),
x.dropna().quantile(0.05),x.dropna().quantile(0.10),x.dropna().quantile(0.25),x.dropna().quantile(
0.50),x.dropna().quantile(0.75), x.dropna().quantile(0.90),x.dropna().quantile(0.95),
x.dropna().quantile(0.99),x.max()],
                  index=['N', 'NMISS', 'SUM', 'MEAN','MEDIAN', 'STD', 'VAR', 'MIN', 'P1' ,
'P5' ,'P10' ,'P25' ,'P50' ,'P75' ,'P90' ,'P95' ,'P99' ,'MAX'])

num_summary=train_num.apply(lambda x: var_summary(x)).T
num_summary
num_summary.to_csv('num_summary.csv')

"""### Handling Outlier"""
#Handling Outliers
def outlier_capping(x):
    x = x.clip(upper=x.quantile(0.99))
    x = x.clip(lower=x.quantile(0.01))
    return x
```

```
train_num=train_num.apply(outlier_capping)

"""#### No missing in train dataset . So , Missing treatment not required ."""
def cat_summary(x):
    return pd.Series([x.count(), x.isnull().sum(), x.value_counts()],
                     index=['N', 'NMISS', 'ColumnsNames'])
cat_summary=train_cat.apply(cat_summary)
cat_summary

"""### Dummy Variable Creation"""
# An utility function to create dummy variable
def create_dummies( df, colname ):
    col_dummies = pd.get_dummies(df[colname], prefix=colname, drop_first=True)
    df = pd.concat([df, col_dummies], axis=1)
    df.drop( colname, axis = 1, inplace = True )
    return(df)

#for c_feature in categorical_features
for c_feature in ['protocol_type', 'service', 'flag', 'attack']:
    train_cat = create_dummies(train_cat,c_feature)
    test_cat = create_dummies(test_cat,c_feature)
train_cat.head()

"""### Final file for analysis"""

train_new = pd.concat([train_num, train_cat], axis=1)
test_new = pd.concat([test_num, test_cat], axis=1)
train_new.head()

# correlation matrix (ranges from 1 to -1)
corrm=train_new.corr()
corrm
corrm.to_csv('corrm.csv')
```

```
# visualize correlation matrix in Seaborn using a heatmap
sns.heatmap(corrm)

"""#### Dropping columns based on data audit report
      - Based on low variance (near zero variance)
      - High missings (>25% missings)
      - High correlations between two numerical variables
"""
train_new.drop(columns=['land','wrong_fragment','urgent','num_failed_logins',"root_shell","su_at
tempted","num_root","num_file_creations","num_shells","num_access_files","num_outbound_c
mds","is_host_login","is_guest_login",'dst_host_rerror_rate','dst_host_serror_rate','dst_host_srv_r
error_rate','dst_host_srv_serror_rate','num_root','num_outbound_cmds','srv_rerror_rate','srv_serro
r_rate'], axis=1, inplace=True)
sns.heatmap(train_new.corr())

"""#### Variable reduction using Select K-Best technique"""
X = train_new[train_new.columns.difference(['attack_class'])]
X_new = SelectKBest(f_classif, k=15).fit(X, train_new['attack_class'] )
X_new.get_support()
X_new.scores_

# capturing the important variables
KBest_features=X.columns[X_new.get_support()]
KBest_features

"""### Final list of variable selected for the model building using Select KBest
attack_neptune, attack_normal, attack_satan, count, dst_host_diff_srv_rate,
dst_host_same_src_port_rate, dst_host_same_srv_rate, dst_host_srv_count, flag_S0, flag_SF,
last_flag, logged_in, same_srv_rate, serror_rate, service_http
"""
train=train_new
test=test_new
```

```
"""## Model Building"""
top_features=['attack_neptune','attack_normal','attack_satan','count','dst_host_diff_srv_rate','dst_h
ost_same_src_port_rate','dst_host_same_srv_rate','dst_host_srv_count','flag_S0','flag_SF','last_fla
g','logged_in','same_srv_rate','serror_rate','service_http']
X_train = train[top_features]
y_train = train['attack_class']
X_test = test[top_features]
y_test = test['attack_class']

"""### Building logistic Regression
#### 1) LogisticRegression
"""
lr_clf = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomial').fit(X_train,
y_train)
y_pred=lr_clf.predict(X_test)
y_pred

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

"""#### 2) RidgeClassifier"""
from sklearn.linear_model import RidgeClassifier
rc_clf = RidgeClassifier().fit(X_train, y_train)
y_pred=rc_clf.predict(X_test)
y_pred

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

"""### K-Nearest Neighbors
#### 1) KNeighborsClassifier
"""
```

```
from sklearn.neighbors import KNeighborsClassifier
k_neigh = KNeighborsClassifier(n_neighbors=3)
k_neigh.fit(X_train, y_train)
y_pred=k_neigh.predict(X_test)
y_pred

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

"""#### 3) NearestCentroid"""
from sklearn.neighbors.nearest_centroid import NearestCentroid

nc = NearestCentroid()
nc.fit(X_train, y_train)
y_pred=nc.predict(X_test)
y_pred

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

"""### Discriminant Analysis
#### 1) LinearDiscriminantAnalysis
"""
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)

y_pred=lda.predict(X_test)
y_pred

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
"""#### 2) QuadraticDiscriminantAnalysis"""
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
qda = QuadraticDiscriminantAnalysis()
qda.fit(X_train, y_train)

y_pred=qda.predict(X_test)
y_pred

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

"""### Decision Trees"""
from sklearn.model_selection import cross_val_score
from sklearn import metrics
import sklearn.tree as dt
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz, export
from sklearn.model_selection import GridSearchCV

clf_tree = DecisionTreeClassifier( max_depth = 5)
clf_tree=clf_tree.fit( X_train, y_train )
y_pred=qda.predict(X_test)
y_pred

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)

"""#### Fine Tuning the parameters"""

param_grid = {'max_depth': np.arange(3, 9),
              'max_features': np.arange(3,9)}

tree = GridSearchCV(DecisionTreeClassifier(), param_grid, cv = 5)
tree.fit( X_train, y_train )
```

```
tree.best_score_
tree.best_estimator_
tree.best_params_

"""### Building Final Decision Tree Model"""
clf_tree = DecisionTreeClassifier( max_depth = 8, max_features=8 )
clf_tree.fit( X_train, y_train )

"""#### Feature Relative Importance"""
clf_tree.feature_importances_

# summarize the selection of the attributes
import itertools
feature_map = [(i, v) for i, v in itertools.zip_longest(X_train.columns,
clf_tree.feature_importances_)]

feature_map
Feature_importance = pd.DataFrame(feature_map, columns=['Feature', 'importance'])
Feature_importance.sort_values('importance', inplace=True, ascending=False)
Feature_importance

tree_test_pred = pd.DataFrame( { 'actual':  y_test,
                    'predicted': clf_tree.predict( X_test ) } )
tree_test_pred.sample( n = 10 )

accuracy_score( tree_test_pred.actual, tree_test_pred.predicted )
tree_cm = metrics.confusion_matrix( tree_test_pred.predicted,
                        tree_test_pred.actual,
                        [1,0] )
sns.heatmap(tree_cm, annot=True,
        fmt='.2f',
        xticklabels = ["Left", "No Left"] , yticklabels = ["Left", "No Left"] )
```

```
plt.ylabel('True label')
plt.xlabel('Predicted label')

"""## Naive Bayes Model
#### 1) BernoulliNB
"""

from sklearn.naive_bayes import BernoulliNB
bnb_clf = BernoulliNB()
bnb_clf.fit(X_train, y_train)

y_pred=bnb_clf.predict(X_test)
y_pred

nb_cm = metrics.confusion_matrix( y_test,y_pred )
sns.heatmap(nb_cm, annot=True, fmt='.2f', xticklabels = ["no", "Yes"] , yticklabels = ["No", "Yes"] )
plt.ylabel('True label')
plt.xlabel('Predicted label')

accuracy_score( y_test, y_pred )

"""#### 2) GaussianNB"""
from sklearn.naive_bayes import GaussianNB
gnb_clf = GaussianNB()
gnb_clf.fit(X_train, y_train)

y_pred=gnb_clf.predict(X_test)
y_pred

nb_cm = metrics.confusion_matrix( y_test, y_pred )
```

```
sns.heatmap(nb_cm, annot=True, fmt='.2f', xticklabels = ["no", "Yes"] , yticklabels = ["No", "Yes"] )
plt.ylabel('True label')
plt.xlabel('Predicted label')

accuracy_score( y_test, y_pred )

"""### Support Vector Machine (SVM)

#### 1) LinearSVC
"""
from sklearn.svm import LinearSVC
svm_clf = LinearSVC(random_state=0, tol=1e-5)
svm_clf.fit(X_train, y_train)

y_pred=svm_clf.predict(X_test)
y_pred
accuracy_score( y_test, y_pred )

"""#### 2) SVC"""
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline

model = SVC(kernel='rbf', class_weight='balanced',gamma='scale')
model.fit(X_train, y_train)

y_pred=model.predict(X_test)
y_pred

accuracy_score( y_test, y_pred )
```

```
"""### Stochastic Gradient Descent (SGD)"""
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

model = SGDClassifier(loss="hinge", penalty="l2")
model.fit(X_train, y_train)

y_pred=model.predict(X_test)
y_pred

accuracy_score( y_test, y_pred )

n_iters = [5, 10, 20, 50, 100, 1000]
scores = []
for n_iter in n_iters:
    model = SGDClassifier(loss="hinge", penalty="l2", max_iter=n_iter)
    model.fit(X_train, y_train)
    scores.append(model.score(X_test, y_test))

plt.title("Effect of n_iter")
plt.xlabel("n_iter")
plt.ylabel("score")
plt.plot(n_iters, scores)

# losses
losses = ["hinge", "log", "modified_huber", "perceptron", "squared_hinge"]
scores = []
for loss in losses:
    model = SGDClassifier(loss=loss, penalty="l2", max_iter=1000)
    model.fit(X_train, y_train)
```

```
    scores.append(model.score(X_test, y_test))
plt.xlabel("loss")
plt.ylabel("score")
plt.title("Effect of loss")
x = np.arange(len(losses))
plt.xticks(x, losses)
plt.plot(x, scores)

from sklearn.model_selection import GridSearchCV
params = {
    "loss" : ["hinge", "log", "squared_hinge", "modified_huber"],
    "alpha" : [0.0001, 0.001, 0.01, 0.1],
    "penalty" : ["l2", "l1", "none"],
}

model = SGDClassifier(max_iter=100)
clf = GridSearchCV(model, param_grid=params)

clf.fit(X_train, y_train)
print(clf.best_score_)

y_pred=clf.predict(X_test)
y_pred
accuracy_score( y_test, y_pred )

"""### Neural Network Model"""
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
scaler = StandardScaler()
# Fit only to the training data
scaler.fit(X_train)

# Now apply the transformations to the data:
```

```
train_X = scaler.transform(X_train)
test_X = scaler.transform(X_test)

mlp = MLPClassifier(hidden_layer_sizes=(30,30,30))
mlp.fit(train_X,y_train)

y_pred=mlp.predict(test_X)
y_pred

from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))

mlp.coefs_
accuracy_score( y_test, y_pred )

#### 1. Bagged Decision Trees
Bagging performs best with algorithms that have high variance. A popular example are decision trees, often constructed without pruning.
"""

from sklearn import model_selection
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
cart = DecisionTreeClassifier()
num_trees = 100
model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())
```

```
model.fit(X_train, y_train)

y_pred=model.predict(X_test)
y_pred

accuracy_score( y_test, y_pred )

"""#### 2. Random Forest
Random forest is an extension of bagged decision trees.
"""
from sklearn.ensemble import RandomForestClassifier

seed = 7
num_trees = 100
max_features = 3
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())

model.fit(X_train, y_train)

y_pred=model.predict(X_test)
y_pred

accuracy_score( y_test, y_pred )

"""#### 3. Extra Trees
Extra Trees are another modification of bagging where random trees are constructed from samples of the training dataset.
"""
from sklearn.ensemble import ExtraTreesClassifier
```

```
seed = 7
num_trees = 100
max_features = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = ExtraTreesClassifier(n_estimators=num_trees, max_features=max_features)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())

model.fit(X_train, y_train)

y_pred=model.predict(X_test)
y_pred

accuracy_score( y_test, y_pred )
```

1. AdaBoost

AdaBoost was perhaps the first successful boosting ensemble algorithm. It generally works by weighting instances in the dataset by how easy or difficult they are to classify, allowing the algorithm to pay or or less attention to them in the construction of subsequent models.

```
"""
from sklearn.ensemble import AdaBoostClassifier

seed = 7
num_trees = 30
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())

model.fit(X_train, y_train)
y_pred=model.predict(X_test)
y_pred
```

```
accuracy_score( y_test, y_pred )

"""#### 2. Stochastic Gradient Boosting
Stochastic Gradient Boosting (also called Gradient Boosting Machines) are one of the most sophisticated ensemble techniques. It is also a technique that is proving to be perhaps of the the best techniques available for improving performance via ensembles.
"""
from sklearn.ensemble import GradientBoostingClassifier

seed = 7
num_trees = 100
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = GradientBoostingClassifier(n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
print(results.mean())

model.fit(X_train, y_train)

y_pred=model.predict(X_test)
y_pred

accuracy_score( y_test, y_pred )

"""### Voting Ensemble
"""
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier

seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
# create the sub models
estimators = []
```

```
model1 = LogisticRegression()
estimators.append(('logistic', model1))
model2 = DecisionTreeClassifier()
estimators.append(('cart', model2))
model3 = SVC()
estimators.append(('svm', model3))
# create the ensemble model
ensemble = VotingClassifier(estimators)
results = model_selection.cross_val_score(ensemble, X_train, y_train, cv=kfold)
print(results.mean())

ensemble.fit(X_train, y_train)

y_pred=ensemble.predict(X_test)
y_pred

accuracy_score( y_test, y_pred )

"""# Save Model"""

import pickle
# Saving model to disk of random forest
pickle.dump(lr_clf, open('model.pkl','wb'))

"""# Load Model and Predict"""
import pickle
model=pickle.load(open('model.pkl', 'rb'))
model.predict([[1,0,0,229,0.06,0.00,0.04,10,0,0,21,0,0.04,0.00,0]])
```

Biodata

A. Personal Data

1.	**Full Name:**	FADEYI TITUS OLAKUNLE
	Address:	No 25, Akintujioye, Street, Ikotun, Ejigbo, Lagos.
	Email Address:	kunletexs@gmail.com
	Phone Number:	07032306377/08024206799
2.	**Date and Place of Birth:**	09/03/1982 - Ibadan
3.	**Nationality:**	Nigerian
4.	**Marital Status:**	Single
5.	**Name and Address of Next of Kin:**	Mr. & Mrs. Olajubu, 31, Ibrahim Babatude Street, Olive Park Estate, Ajah, Lagos.

6. Date of Assumption of Duty in Current Establishment: 4th September, 2021
7. Status on First Appointment in Current Establishment: Teacher
8. Present Position: Teacher
9. Date of Confirmation of Appointment: September, 2023

B. Educational Background

1. Educational Institutions Attended with Dates and Qualification:

i. Primary Education: Ebire Nursery/Primary School, Ibadan
(1985 – 1992)
Primary Leaving Certificate

ii. Secondary Education: Unity School, Ejigbo
(1993 – 2000)
O' Level Certificate

iii. Higher Educational Institutions Attended with Dates & Qualification:
Babcock University, Ogun State
(2003 – 2007)
B.Sc Computer Science (Information Systems)

C. 1. Work Experience: With Dates (including Courses Taught Where Relevant)

- Awesome college, Lagos
(2009 – 2015)
Information and Communication Technology
- Caleb British International School,
(2016 – 2023)
Computer Studies

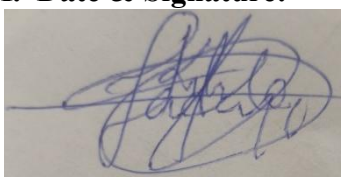
2. Courses taught within the current academic sessions: Computer Studies

D. Extra-Curricular Activities: Watching news

E. Names and Addresses of Referees:

- Mrs E. Ayodeji
Head of Department (HoD) English
Caleb British International School, Lagos
- Mr O. Owolabi,
Principal
Plumbreed School, Lagos

M. Date & Signature:



Signature

05th October, 2023.

Date

The University Compliance Certification

This is to certify that this thesis is Olakunle Titus FADEYI with Matriculation Number LCU/PG/000824 in the Department of Computer Science, Faculty of Natural and Applied Science, Lead City University, Ibadan is in full compliance with the approval of the University's format and style.

_____ _____

Signature Date