

Liverpool John Moores University

School of Computer Science and Mathematics

**6200COMP Project**

Final Year Dissertation

submitted by

**OLUWADARA OMIWALE**

**1001281**

**COMPUTER SCIENCE**

Title

**Using Ai to increase the accuracy of intrusion detection systems in key infrastructure.**

Supervised by

**Fariba Sharifian**

Submitted on

**11 April 2024**

Abstract

Cyber security has struggled against the raging war of cyber-crimes such as cyber-attacks including worms and viruses and, in a world, where key infrastructures is dependent on the security of network systems it is important cyber security remains strong. This study aims at increasing the strength of cyber security by utilising Ai to increase the accuracy when detecting abnormal behaviour.

This study contains an introduction of cyber security and Ai and the problem statement in which this study aims at solving which is ‘using Ai to increase the accuracy of Intrusion Detection Systems in key infrastructures’ and finally my motivation, which was my profound interest in cyber security but also my growing interest in the capabilities of Ai

Furthermore, in this study were things such as objectives that needed to be considered for the project to be successful and steps taken to ensure the success of the project. In conclusion from this study, I was able to receive an impressive result of 100% from a model and conclude that we can indeed use AI to improve the accuracy of IDS.

Acknowledgement

This report was completed on my own however I would like to acknowledge the following people for their help in the completion of this report

Family and friends – for just providing me with their presence and helping me keep a sane mind

Partner – my Gf Calli for motivating me to do work all the time even on days that I was demotivated and lost

Supervisor – my supervisor Fariba for just being available to help me when I needed advice and/ or tips

Table of Contents

[1. Introduction 5](#_Toc195175965)

[1.1 Background Information 5](#_Toc195175966)

[1.2 Problems Statement 6](#_Toc195175967)

[1.3 Motivation 7](#_Toc195175968)

[2. Background Research and Domain Analysis 8](#_Toc195175969)

[2.1 Background Research 8](#_Toc195175970)

[2.1.1 Cyber security 8](#_Toc195175971)

[2.1.2 Machine learning 9](#_Toc195175972)

[2.1.3 Integration of Machine Learning into cyber security 14](#_Toc195175973)

[2.2 Domain Analysis 16](#_Toc195175974)

[3. Requirement Analysis and Methodology 20](#_Toc195175975)

[3.1 Requirement Analysis 20](#_Toc195175976)

[3.1.1 Objectives 20](#_Toc195175977)

[3.1.2 Model requirements. 20](#_Toc195175978)

[3.1.3 Data requirements 21](#_Toc195175979)

[3.1.4 Evaluation requirements 21](#_Toc195175980)

[3.1.5 Functional and technical requirements 21](#_Toc195175981)

[3.1.6 Non-Functional 22](#_Toc195175982)

[3.1.7 Challenges and Considerations. 22](#_Toc195175983)

[3.2 Methodology 23](#_Toc195175984)

[3.2.1 Data Collection 23](#_Toc195175985)

[3.2.2 Explanatory Data analysis 23](#_Toc195175986)

[3.2.3 Feature selection 23](#_Toc195175987)

[3.2.4 Cleaning and preprocessing 24](#_Toc195175988)

[3.2.5 Machine learning 24](#_Toc195175989)

[3.2.6 Evaluation measures 25](#_Toc195175990)

[3.2.7 Hypothesis: 26](#_Toc195175991)

[4. Design of Artefact 27](#_Toc195175992)

[4.1 Finding the Dataset 27](#_Toc195175993)

[4.2 Preprocessing the data 29](#_Toc195175994)

[4.3 Machine learning algorithms 31](#_Toc195175995)

[5. Development of Artefact 33](#_Toc195175996)

[5.1 Decision tree: 33](#_Toc195175997)

[5.2 Random Forest: 39](#_Toc195175998)

[5.3 Neural network 42](#_Toc195175999)

[6. Testing and Evaluation of Artefact 48](#_Toc195176000)

[6.1 Metric evaluation: 48](#_Toc195176001)

[6.2 non-metrics evaluation 49](#_Toc195176002)

[6.3 Chosen model: 50](#_Toc195176003)

[7. Project Evaluation 52](#_Toc195176004)

[8. Conclusion and Future Work 53](#_Toc195176005)

[Conclusion 53](#_Toc195176006)

[Future Work 54](#_Toc195176007)

[9. Reference list 55](#_Toc195176008)

[Appendix 60](#_Toc195176009)

[Project Specification 60](#_Toc195176010)

[Signed Monthly Reports and other documentation 60](#_Toc195176011)

[Figure 1: flowchart of proposed plan 27](#_Toc195260053)

[Figure 2: code showing how I separated my features 30](#_Toc195260054)

[Figure 3 : image of dataset after features Selecetion 30](#_Toc195260055)

[Figure 4: code showing removal of empty values 31](#_Toc195260056)

[Figure 5: Structure of neural network (Taken from google images ) 31](#_Toc195260057)

[Figure 6: Structure of Random Forest (Taken from google images) 32](#_Toc195260058)

[Figure 7: Structure of Decision Tree (Taken from google images) 32](#_Toc195260059)

[Figure 8: image of features and their data types 35](#_Toc195260060)

[Figure 9: image of decision tree 35](#_Toc195260061)

[Figure 10 : results for decision tree 38](#_Toc195260062)

[Figure 11:test1 40](#_Toc195260063)

[Figure 12:test2 40](#_Toc195260064)

[Figure 13:test3 40](#_Toc195260065)

[Figure 14:test4 40](#_Toc195260066)

[Figure 15:test5 40](#_Toc195260067)

[Figure 16: best parameters 41](#_Toc195260068)

[Figure 17: error on neural network 43](#_Toc195260069)

[Figure 18: first run of neural network 43](#_Toc195260070)

[Figure 19: second run on neural network 43](#_Toc195260071)

[Figure 20: image of accuracy 44](#_Toc195260072)

[Figure 21: same accuracy, worse mode loss 46](#_Toc195260073)

[Figure 22: decreasing learning rate 46](#_Toc195260074)

[Figure 23: image showing overfitting 47](#_Toc195260075)

[Figure 24: image showing no more overfitting 47](#_Toc195260076)

[Figure 25: cross matrix 50](#_Toc195260077)

[Figure 26: cross-validation 51](#_Toc195260078)

[Figure 27:ROC curve 51](#_Toc195260079)

# Introduction

## 1.1 Background Information

As time goes on the need for the internet increases even more, with people’s lives being dependant on the internet in many ways such as communicating with each other and doing their jobs and the more importantly self-healthcare and lives being monitored by devices connected to the internet. The internet has become a foothold in humanity’s existence and more organizations have introduced the concept of the internet, and this comes with a downfall as more organisations become vulnerable to cyber-crimes (Kumar, 2010) . Ever since the introduction of the internet cybercrimes have been existent even to the point that it’s suggested that the first cyber-crime took place before the internet existed in 1834. So ever since then Cyber attackers have been one step ahead and as we may know it is impossible to be 100% un-hackable but what if we could make it so cyber-crime did not even get the chance to be committed in the first place. This project aims at exploring the possibility of this by using AI to increase the accuracy of intrusion detection system in detecting any anomalies among the network caused by security threats.

AI first came to existence in the 1950s. 1952 was when the first ever chequers game played by a program took place and over time innovative programs were in creation for example in 1981 the Japanese government allocated over 2 billion (in today’s currency) to the ‘fifth generation computer’ (intelligence, 2024). The aim of this was to create computers able to translate human language. Over time AI gradually grew and became significantly important as it could do many more things and very much better than any human because it was trained, and it had a higher success rate than a human could reach on their own. As time went on Ai became a significant aspect of the internet and many new features were discovered alongside. One main feature that is focused on in this project is Machine learning algorithms.

Machine learning became an extremely useful tool used to leverage technologies around AI. It was a part of AI until the late 1970s in which it branched of and evolved on its own (Foote, 2021). Machine learning algorithms enable a machine or system to learn and improve from experience. There are four types of machines learning, supervised, unsupervised, semi-supervised and reinforcement (Wakefield, 2024) and many examples of ML Models. With how machine learning algorithms work they need data to train thus this introduces the idea of data mining. data mining is the process of analysing large datasets to discover nontrivial patterns, Trends, and insights. Data mining techniques can either target data sets or can predict outcomes by using ML algorithms and these methods have to ability to organise and filter data from fraud and even security breaches (Holdsworth, 2024).

Intrusion Detection system (IDS) is a set of methods or techniques that is used to detect suspicious activity in a network, the monitor network or information system for any signs of malicious activities and they respond by warning the system administrator displaying alerts and logging events. They communicate to the host with two ways, one is an absence of alert, and the other is the presence of an alert (Stallings et al., 2008). The earlier forms of IDS were developed in the 1980s (Okta, 2024). IDS can use signatured based techniques, anomaly-based techniques or hybrid-based techniques.

The integration of machine learning and intrusion detection have been highly significant and beneficial to cyber security as traditional methods lack in effectiveness machine learning offers a solution by enabling the systems to learn.

## 1.2 Problems Statement

Intrusion detection systems are commonly used in network systems to detect and prevent malicious intruders. As mentioned they are able to monitor network traffic and identify any unauthorised access and they can do this by the following ways: 1) They monitor operations of Firewalls, routers and key management servers that security controls aim at detection , preventing or recovering from cyber-attacks, 2) Provide ways for the administrator to further understand relevant log files providing a user-friendly interface so both nonexperts and expert can assist in managing system security and finally 3) Recognise and report when the IDS detect system files being altered and generate alarms to notify of breaches (Hashemi-Pour, 2024). With those steps IDS can provide an in-depth protection to network systems if functioning to its intended purpose.

However, as many things it also has its cons and problems. As time goes on new malware is created and developed and this becomes a problem, considering the types of IDS there are some such as signature based – behaviour that have limitations and face challenges in the environment of new malware this is because it is unable to detect new patterns or indicators of new threats that aren’t already known. While other such as anomaly-based can it still leads to the main problem of how accurate they will be able to detect malware attacks.

The Problem that this report focuses on is increasing the accuracy of Intrusion detection systems. To evaluate on this statement, I plan to reduce the number of false positives and false negatives reported by using Ai to train the IDS. A **False Positive** is when the IDS identifies an activity as a harmful attack when the activity is acceptable behaviour to the network and a **False Negative** which is the highly more dangerous one is when the IDS identifies an activity as acceptable behaviour to a network, but it is a harmful attack to the network (Hashemi-Pour, 2024). The reason a False negative is more dangerous is because, unlike the false positive in which no harm is caused, when it occurs IT teams have no idea that their network has been or is being infiltrated and do not find out till too late. In a sense it is better for an IDS to be oversensitive and bear more false positives than negatives. This is liable to occur in IDS that are used to searching known patterns.

False negatives are becoming a bigger issue for Intrusion detection system as malware is always evolving and changing and for IDS, regardless if signature or anomaly based, this makes it harder to detect intrusions as the evolved malware may not display already known patterns of suspicious behaviour that they are designed to detect (Hashemi-Pour, 2024).

Therefore, this project Aims to use Ai to train an IDS to significantly reduce the number of false negatives (and positives). Other Expected problems that will arise is finding a good data set and training sample for the machine learning to construct our model and evaluating the efficiency of its learning.

## 1.3 Motivation

The motivation for this project was created from two reasons, the first being I wanted to develop new skills and increase my knowledge. For a little while cyber security has always been an interesting concept to me in the fact that our world currently relies on it for more reasons that we may think such as preventing authorised access into bank accounts or even government armoury. I want to follow a career in cyber security as while it is a huge responsibility it's also a thing that is needed as it aids in protecting people’s lives. Furthermore, with the addition of Artificial intelligence and its capabilities to aid and develop humans’ confidence in solving complex problems it became an equal if not more interesting concept. There have been many studies of introducing AI into cyber defence to develop cyber security and as someone who finds both topics interesting, I was inspired to fulfil my report based of this hence why this report research utilises AI(ML) and Network Defence (Intrusion detection systems). I am able to do this by studying courses such as ‘network defence’ and ‘secure software development’ which teach me basics and more of cyber-security and tools that aid strength the protection of networks and others such as ‘ knowledge based system’ and ‘ contemporary concept in computer science ‘ which further expand my existing knowledge on Ai and types of Ai and what it can be used for , for example stimulating environments of warehouse factories using robots to deliver and sort out boxes or more relevant being able to sort through massive amounts of data in a database . With these courses being learnt I intend to integrate the capabilities of Ai into tools used in network defence.

Second reason was I want to take on challenges. this report aims at finding the solution to whether we can reduce the number of false negatives and positives. There are reports of experts also researching this and finding certain solutions However, even if already solved I intended to do the following: 1) come up with my own experiment and own results that are preferably better than most out there but in the case of failing, 2) improve existing ones out there by evaluating their methods and testing different variables to just gain a deeper understanding.

# Background Research and Domain Analysis

## 2.1 Background Research

### 2.1.1 Cyber security

Cyber security is how individuals and organisations reduce the risk of cyber-attacks and can be defined as a measure of protecting computer systems, networks and information from any authorised access, modification or destruction. (Thakur, 2015). Furthermore, cyber security ensures confidentiality, integrity and availability of information assets being processed stored and communicated (Stallings et al., 2008). Confidentiality means preserving authorised restrictions on information and a loss of this would be unauthorised disclosure of information. Integrity means Guarding against improper information modification or destruction of information and a loss of this would be unauthorised modification or destruction of information. Finally, Availability ensures timely and reliable access to and use of information and a loss of this would be disruptions of access to or the use of information or systems. Cyber security consists of these three knows as the CIA triad. Cyber security exists for these reasons and in addition is used to protected against Threats, vulnerabilities and attacks (TVA). Threats are the potential for violation of security, such as there is a capability, action or even that could breach security and cause harm. Summarised down a threat is a possible danger that might exploit a vulnerability, and an example of a threat could be the opportunity to gain unauthorised access into systems such as bank. Threats can come in many forms such as malware which comes in many types such as worms, ransomware and viruses. Phishing and Social Engineering which is an attempt to gain access to an infrastructure by claiming to be a legitimate source, this is a threat that has been persistent even before the availability of the internet and is still rising and Finally APT which stand for advanced persistent threats and are sophisticated cyber-attacks in which an intruder establish a undetected presence in a network to steal sensitive information (Lenaerts-Bergmans, 2023) . Vulnerability is something that can allow threats to be realised and this can be weakness in the system. An example of this could be lack of two factor authentication or weak password vulnerability and finally an Attackis an assault on the system, that derives from the intelligent threat, this will always occur by attacking the vulnerability in the system. CIA and TVA show the basics of cyber security and the three categories that cyber security must cover. Cyber security comes in many forms such as critical infrastructure, cloud security, network security and more. Cyber security is important because the internet is a fundamental part of modern life due to its stronghold in online banking and shopping (National Cyber Security Centre, 2020). As infrastructure grows massively day by day so does the possibilities of cybercrimes hence the desperate need for extensive development in cyber security. Threats will continue to evolve due the growing population of those well knowledge and diverse in networks and despite this there is a shortage of skilled professionals globally to help secure organisations properly and this makes it difficult for cyber security to stay ahead of attackers (Sayegh, 2025).Hence based on the threats that security network and others face there is the need for development of security tools.

Cyber security first began in the 1970s when a researcher called Bob Thomas created a computer programme called Creeper that would move across ARPAnet network leaving a breadcrumb trail. (Davies, 2021). The ARPAnet was the first public-packet switched computer network used for academic and research. Creeper was alleged to be one the first computer worms alongside Reaper, the very first anti-virus software and self-replicating program and Morris worm the first public attack in 1988. Creeper is the first instance of a threat on computer systems whereas reaper is the first security tool developed to counter threats. Furthermore, fast forward to 1990s in which the internet became available to the public network security threats increased exponentially due to increases in organised crime and attacks and as a result firewalls and anti-viruses had to be programmed on a mass scale to protect network security and the public and throughout the 1990s to 2010s heavily funded professional cyber-attacks occurred and governments had to increase cyber security resilience and punishment. One example was in 2011 Microsoft awarded a money prize for whoever could offer a technical improvement in windows to stop malware (Sergio, 2023). Now in the 21st century cybersecurity market is large with common threats increasing day by day and new malware and threats being introduced.

Cyber security had to evolve over time, and this meant inventing new and better firewalls and anti-virus and one important security tool that came out of these were intrusion detection systems (Stallings et al., 2008). The first ever intrusion detection system is alleged to be created in an academic paper written in 1986 (Pirc, 2017) and still then the fundamentals haven been used in IDS and IPS developed today. Prior to IDS, firewalls were the most common security tools used. It wasn’t until 2010 that security companies stepped up the competition in advanced combination of Intrusion Detection and Prevention system

Despite the growing problems of threat, the future of cyber security looks bright with the introduction of Ai and more specifically machine learning.

### 2.1.2 Machine learning

AI which stands for artificial intelligence was initially designed to simulate the function of a single neuron with is simplest models starting as a simple input-output functions. One of the first known published work on AI was by McCulloch and Pitts in 1943 and their overall proposal was using a model that used Boolean inputs , processes them in a specific manner and if the processed value exceeds a certain threshold the neuron will be considered to have been activated (Forghani, 2020). Whereas the earliest substantial work in the field of artificial intelligence was done in the mid-century by a British computer pioneer called Alan Mathison Turing (Copeland, 2024). Alan pictured an idea of a computing machine consisting of a limitless memory and a scanner that moves through the memory – symbol by symbol – reading what it finds and having the capability to write further symbols. The actions of the scanner were dictated by a program of instructions that was stored in the memory in symbols. This later became known as the Turing machine and now all modern computers are essentially Turing machines (Copeland, 2024). Alongside this one of Turing original idea was to train a network of artificial neurons to perform specific task. In 1957 the Perceptron was created by Frank Rosenblatt , the perceptron was one of the first algorithms to use neural network which is widely used in machine learning today (Koch, 2022) its design was to improve the accuracy of computer predictions by learning from data and adjusting its parameters until an optimal solution had been reached .One of the first successful AI programs was developed in 1951 at university of Oxford in which a program was able to complete a game of checkers at reasonable speed another development which is further down the line in history is AlphaGo which was a program developed by DeepMind and made the headline after defeating a world champion Go player in a five-game match (winning four out of five) (Granter, 2017). There are reportedly many more programs and algorithms that were created such as Nearest neighbour algorithm in 1967 that identify patterns within large datasets and backpropagation in 1974 which was an upgrade designed to help neural networks and many more. There are arguments on their success but those mentioned demonstrated an idea on how the idea of AI and more specifically Machine leaning came to take place. These were the cornerstones to what we now know and see as AI and its capabilities to read stuff and further develop on it.

Machine learning is divided into three categories: Supervised, unsupervised, semi-supervised. (TimesPro, 2024).

#### 2.1.2.1 Supervised machine learning.

Supervised is the task of machine learning to learn a function that maps an input to an output based on sample input-output pairs, furthermore it uses labelled training data and a collection of training sets to infer a function and recognise patterns. This is more used when certain goals are identified to be accomplished from certain inputs. The most common are Support vector machines and neural networks and its applications are “classification” which can separate data for example in spam detection and “regression” which fits the data for example predicting patterns. The advantage of using this type of machine learning is a high accuracy in predictions due to the labelled data and clear results are produced but a disadvantage is it requires a large amount of labelled data which tends to be expensive and time-consuming (TimesPro, 2024).

In the paper ‘methods of IDS using deep neural network’ there are many types of tested techniques on using different Machine learning algorithm with IDS for example using ML algorithms such as SVM and decision tree and k-means. Another technique used was studying systems that can detect attacks using hybrid ML by combining K-means and SVM. But the main technique used, especially in a report by Jin Kim and others, was DNN (Kim et al., 2017).

In their paper they primarily specialised in presenting an AI-based detection systems using deep neural network to develop a fast and effective intrusion detection system. Deep neural network is just a class of machine learning algorithms like the artificial neural network, it aims to mimic the information processing of the brain. The dataset utilised was the KDD cup 99 developed by the defence advanced research projects agency aka DARPA and it contained both normal and abnormal connection on the network.

The first step in which they took was the obvious, data pre-processing. Data preprocessing which is the next step after data collection (collecting the data sets) includes processing and transformation of data by filtering unwanted and unreliable data from the required and targeted data needed and/or just rearranging data so that the model can understand. They had one record in the KDD cup 99 dataset that had a mixture of numeric values and symbolic values, and they converted them (Kim et al., 2017).

It is worth notifying that they did two things to possibly enhance the model’s performance the first being using the ReLU function as the activation function of hidden layers. ReLU function operator by outputting the input directly if positive. The point of this was to overcome the vanishing gradient problem allowing models to learn faster and perform better. The second one they used the adaptive moment optimizer. Another thing worth notifying is the environment in which they tested it in. DNN requires many calculations and hence it was essential for them to utilise something with the capability of processing many pieces of data simultaneously which is exactly what they did by using an GPU, this resulted in less time being spent on teaching the model and more time implementing it. The software TensorFlow was used to support the construction of the model (Kim, 2017).

After the pre-processing of the data, they then went onto the training and testing sets. KDD cup 99 has around 4.9 million records in its data set. Jin Kim idea was to use 10 percent of it for the training set while the testing set would use 100 percent of the dataset. Within that 10 percent training set they further went on to split it into two data: normal and attack data. The training set overall was made to include 10% normal data increasing to 90% while simultaneously the attack data was at 90% attack data decreasing to 10% respectively. For example, 20% normal and 80% attack data or 70%-30% (Kim, 2017).

The final thing touched on by Jin Kim and the others was their evaluation measure. They, as many reports, were testing to see the accuracy, detection rate and false alarms to evaluate the performance of the model. Before explaining the formula, they used Its useful to remember the abbreviations and what they mean. There is TP which means true positive, and that actual attack are detected and classified as attacks and there is TN which means true negative, and that normal data is detected as normal. Then the obvious FP and FN which have been previously talked about. False positives and false negatives are when the data are misclassified too be normal data that are classified as attacks and vice versa and is what many projects including my own aim at reducing.

Formulas used:

To evaluate the accuracy:

For detection rate

Finally for false alarms .

In this second paper ‘A machine learning approach for improving performance of network IDS by Azizan (Azizan et al., 2021) and others they discuss the methods they have used which is both similar but different, the main similarities is one of their machine learning methods used is SVM with the other machine learning methods used being Random forest and Decision tree two commonly known and used machine learning algorithms They had a knowledge discovery method that was used to perform research and the database they used was the KDD 99 dataset. However, the main difference is the dataset they used to access the performance of the three classification algorithms which was the intrusion detection evaluation dataset CIC-IDS201 with 85 attributes and includes attacks such as Dos, DDoS, brute force -similar to the type of attacks this project aims at reducing - and more.

Their evaluation methods were based of the confusion matrix which is used to define the performance of classification algorithm. like the evaluation metrics used in the first paper. The confusion matrix is needed to choose a decision threshold to label instances as positive or negatives. In which if the probability assigned by the ration exceeds the limit it would be declared positive and if it was less than the threshold It would be declared negative. They also used similar formulas to those in paper one:

Precision = which measures the ratio of attacks flow to the characteristic flow,

Recall = which is a ratio of correctly identified attacks over the overall predicted flows

Accuracy = this is the most used metric to judge a model and is not a clear indicator of performance and shows the percentage of true detection.

In the paper they later go on to show the steps they took in processing the data of the ML based model created from the dataset which follows as:

1. They Performed data extraction from the collection of CIC-IDS2017 dataset.

2. Consolidate conflicting information from multiple sources into a specific resource.

3. They would select the data based on the applicable data to the test.

4. Convert the data required through mining strategy into an appropriate structure of 10-folds cross validation.

5. Extract patterns potentially helpful for modelling designs in the training phase.

6. Perform pattern assessment to the abnormal traffic to enhance the recognition of the model design based on the given evaluation measures of the testing phase.

7. Represent the results of the two phases as findings.

Final thing to notice is their experiments was performed using the AZURE Machine Learning Tool using the validation method for 10-fold training and testing (Azizan et al., 2021).

Finally, in the paper ‘Network IDS using Neural network’ focuses on an intrusion detection system based on neural networks.

Within this They listed some of the techniques they use in the study such as, expert systems, signature analysis, data mining and of course neural networks. Within this paper they used a feedforward neural network based on a backpropagation training algorithm. Furthermore, an input layer, hidden layer and output layer is implemented. The dataset they used was obtained from the DARPA depository that was taken by the TCPDUMP program. Within this the attacks were broken down int of our categories Dos, remote to user, user to root and probe (Shun and Malki, 2008).

Furthermore, they mentioned the original parameters included in the dataset and how they pre-processed it down. The original parameters consisted of

* SESSION ID – which was used to uniquely identify a sessions connection
* Start date – the date in which they started
* Start time – and the time of the day the session started
* Duration – the length of the session
* Service – and the protocol used by the session
* Source port – the port used by incoming service
* Destination port – port targeted by the service
* Source Ip – internet address of source object
* Destination port – internet address of destination object

The main parameters they used was date, time, duration, protocol, source port and destination port (Shun and Malki, 2008) simply removing the address from their study.

Upon doing their experiments in doing four separate test splits into normal traffic, unknown attack traffic and known attack traffic they were able to receive results that using a neural network model correctly identified all normal and known traffic set and that 76% of unknown attacks were classified correctly.

#### 2.1.2.2 Unsupervised machine learning.

As shown supervised machine learning has its own advantages, but so does Unsupervised machine learning. Unsupervised machine learning analyses unlabelled datasets without the need for human intervention and can furthermore find patterns and structures within the given data without predefined labels. One common example of this is k-means clustering and this is applied in anomaly detection as in cyber security these types of models are trained to identify unusual patterns. Advantages of using this learning method is there is no need for labelled data making it cost-effective and less time consuming, and its capabilities to discover hidden patterns. Disadvantages of using this is often requires existing knowledge to make sense of the results its results can be less precise and harder to interpret unlike supervised. Supervised and unsupervised counter each other advantages and disadvantages (TimesPro, 2024).

Alongside there being many techniques for supervised models there’s also a few for unsupervised, for example In This paper ‘Network intrusion detection for cyber security using unsupervised deep learning’ by Alom (Alom and Taha, 2017) they mention machine learning models such as CNN that is shown to have better performance and Extreme Learning Machine (ELM) another model that is good with higher regularization performance at a much faster speed. It’s worth notifying they mention other studies that used a proposed Semi-supervised ELM, and an un-supervised ELM used in different domains, and this was their first attempt at implementing the Un-supervised ELM.

AE and RBM was used in this study to extract the features and iterative k-means clustering was used for final detection attacks, they also used the KDD-cup 99 dataset and he accuracy of the proposed approaches was compared against other machine learning techniques of k-means. They propose that an unsupervised deep learning based is highly beneficial especially when it comes to detecting new types of attacks with a minimum accuracy of 92%. They utilised AE and RBM for proper data encoding with dimensionality reduction. An auto-encoder is a fundamental deep learning approach with the aim of transforming input data into outputs with the minimum possible amount of distortion. An Auto-encoder is used for unsupervised feature learning with efficient coding. With the main objective to learn and represent data.

In another paper called ‘Deep learning for unsupervised insider threat detection’ they are experimenting using an unsupervised deep learning model approach to detect insiders’ behaviour patterns and they planned to do this by using Deep neural network and Recurrent neural networks which are trained to recognize activity that was characteristic of a user (Tuor et al., 2017). The effectiveness of the model was based on the synthetic CERT insider threat v6.2 dataset.

Furthermore, as this was an unsupervised approach they were able to utilise one of its advantages in models can be trained with unlabelled data they just split the dataset into 85% for training and 15% testing and after doing some tuning, they did several mini experiments to compare to each other also using TensorFlow. They would compare their models against typical anomaly detection methods such as Support vector machines and Isolation Forest and were able to conclude that the models they used, DNN and RNN, outperformed the most used ones such as SVM and isolation forest.

#### 2.1.2.3 Semi-Supervised machine learning.

Finally Semi-supervised as suggested in its name is the combination of the techniques used in both supervised and unsupervised, it utilises methods that blend those of supervised and unsupervised. it combines a small amount of labelled data with a large amount of unlabelled data. Its advantages are that it contains the strength of both supervised and unsupervised learning and can significantly improve performance with less labelled data than what purely supervised learning could do, however its disadvantages are its more complex due to balancing labelled and unlabelled data and requires careful tuning to achieve optimal results (TimesPro, 2024).

Researched in one paper called ‘Cyber security attack detection model using semi-supervised’ displayed that their aim was to leverage the power of semi-supervised learning, auto-encoders a subset of neural network and Probability Bayesian Networks (Kolawole Ayodeji et al., 2025).

The use of the auto-encoders was because semi-supervised auto-encoder can leverage both labelled and unlabelled which addresses the limitation of data dependency and robustness seen in MC-VAE (different type of auto encoder), the idea of this was by using unlabelled data the model would be able to capture a broader range of normal and potentially dangerous behaviours. These auto-encoders are also used to help keep the model’s complexity and simplicity balanced.

In this study four experiments had been done using the following: NUSW-NB15\_GT as the database. MC-VAE, PBN, MC-VAE-PBN and SSL-PBN as the models and amongst the four SSL-PBN had received the highest amount across all their evaluation metrics which were accuracy, recall, precision and F1 score which implies the semi-supervised approach was the strongest.

Furthermore, in the paper ‘Ransomware detection with semi-supervised learning’ by Saberi (Noorbehbahani and Saberi, 2020) in which they are detecting ransomware attacks semi-supervised learning was utilised to cope with the large training data they had that supervised learning can’t handle and would be time-consuming and expensive. To conduct the experiment Weka and collective package was applied.

Their proposed steps in carrying out this study were:

1. data set preparation - selecting the ransomware dataset from the CICandMAL2017 alongside 10 ransomware datasets and combining the 10 ransomware family datasets to create the DM dataset this step also concluded of data preprocessing.
2. Examining five features – (CFS) an algorithm that puts together evaluation formula with appropriate correlation measure, (One Rule) simple classification algorithm that generates rules, Gain Ratio, Relief and Chi-squared all algorithms with individual uses and advantages
3. Implementation of four semi-supervised learning methods

* CHOPPER – applies a classifier that labels test data after training
* YATSI – another classifier that labels unlabelled data additionally with the application of the kNN using actual training set and pre-labelled data, the test instances were classified
* Collective IBK – employs the ibk algorithm to find the best k in the training set.
* Wrapper based – classifiers are trained on labelled data then the predictions of the classifier are applied to generate additional labelled data to retrain the classified and results are evaluated using k-fold cross validation and accuracy measure.

Their results were that the Wrapper Rf classification was the best model when learning with random forest as a base classifier as it outperformed other methods. And, chi squared, and One Rule were effective feature selection methods in semi-supervised ransomware detection.

We can see the significance of machine learning and its different capabilities when training. Hence why when introduced to cyber security there are many possibilities for development of Cyber Security

### 2.1.3 Integration of Machine Learning into cyber security

Following on from before Ever since the creation of the internet human lives have become dependent on it, more so the key infrastructure that have been built upon or with it. Key infrastructures or critical infrastructures are systems, networks or public works that a government considers essential to its functioning and safety of its citizens, examples include electrical grids or public services such as hospitals. Resulting from this is the dire need to protect critical infrastructure to protect humans’ wellbeing and this is done through Cyber Security. There have been many cases of such key infrastructures being attacked by malicious actors. Two prime examples are Stuxnet a worm that infiltrated an Iranian nuclear plant facility and the Ukraine power grid attack in 2022.

Stuxnet was a worm that targeted SCADA systems and used advantage of multiple zero-day vulnerabilities to infect computers. It stayed undetected spreading through the operating system and furthermore was able to take control of programmable logic controllers which were small computers used to regulate the power in the devices (Baezner, 2017).The nuclear facility was air-gapped and had no connection to the internet but the worm was spread by the vector of a removable USB-drive , regardless of this Stuxnet had not been discovered initially and was able to bypass the monitor operator , people who monitor and operates systems essentially .Another example was Ukraine power grid being attacked by an APT called ‘sandworm’ which utilised LOTL techniques to trip substation circuit breakers which eventually led to unplanned power outage (CANDAN, 2024). This later left four regions temporarily without electricity and disrupted access across several areas. The important fact in this was the initial intrusion began as early as June, so until the disruptive events in October the threat actor had been undetected on the network for 2-3 months. Both examples show the need of enhanced network security and why IDS should be deployed and utilised as its beneficial and we can only imagine and estimate how much less damage would have been caused had they utilised efficient Intrusion detection systems or if they already did how much more efficient would it have been.

As already mentioned in the introduction IDS uses multiple ways to detect them being signature based and anomaly based. Traditional intrusion detection methods have served as a strong foundation for decades but with how rapid new threats are emerging such as generated AI attacks it diminishes their effectiveness in detecting the attacks. Signature based detection relies on a database of known threats which means without constant updates it is not as effective and is basically made redundant in the face of a new malware attack. Anomaly based detection are prone to false outlies leading to several alerts (Markevych, 2023). Overall, there are issues with using traditional methods such as scalability issues. Using a Hybrid AI-based IDS compared to other techniques such as signature or anomaly based has major advantages such as Flexibility, Adaptability, Pattern recognition and Fast computing (sachdeva, 2010). Hence why this project as many others aim to use integrate Ai with Cyber security and use Ai to train and make IDS more efficient.

A simple example on the effectiveness of integrating Ai into network defence is an Ai-based cybersecurity platform, developed by mathematicians in the university of Cambridge in 2013, was able to leverage machine learning to detect and respond to threats in real time. It would operate by learning the normal behaviour of users, device and networks in an organization and once it established a baseline of normal activity it would detect deviations that could suggest cyber threats, for example if a device started downloading large volumes of data outside of typical working hours the Ai would flag it as suspicious, this became an AI that learnt on the job as it mimicked an immune system of humans. By learning a sense of ‘self’ it was able to discover subtle, previously unseen patterns and emerging threats (Garza, 2020) which is what a lot of IDS do now. Darktrace was able to effectively prevent numerous cyber-attacks across a range of industries including healthcare and energy, both critical infrastructures. One feat of accomplishment to be noted is its intervention in a healthcare organisation where it detected and responded to a ransomware attack before it could encrypt critical data and as a result it minimized the damage, saving the organization from significant financial and reputational loss. This is the expected and wanted result when integrating AI into cyber security. Hence why it’s becoming significant to combine the capabilities of Ai with Security tools.

## 2.2 Domain Analysis

Carrying on from earlier there are many cyber threats going on around the world and the cybercrime examples mentioned previously show the pressing need for enhance IDS especially with an increasing reliance on digital infrastructures. Therefore, increasing the accuracy and effectiveness of Intrusion detection systems is important to boost the defence of cyber security against old and new malware. As touched upon the main problem that this report aims at solving is increasing the accuracy of IDS furthermore reducing the amounts of false positives and false negatives. They are a problem for the few reasons such as a high rate is dangerous and can leave a team overwhelmed and unable to distinguish between real and fake alerts and they can be more of a risk to cyber security, however after researched its shown that there not the only factors that increase risk, there is also the rate at which these alarms are made from detection of malware attacks. Even if an IDS was able to detect at a high success rate if it was slow in doing so and left no time for those who are expert responders to react especially with little context it would be just as damaging. The effectiveness of IDS and solution to the problem all come down to these factors (corelight, n.d.).

My original knowledge of AI was quite limited especially considering Machine learning algorithms. On doing a group project I was able to learn of three types of machine learning algorithms, those being: random forest, decision-tree and XGM boost. These were the only main machine learning models that I knew of and how it worked and furthermore the methods used in this were splitting data into training and testing with training being 30% and testing 70%, but upon researching furthermore and seeing how many different machine learning algorithms there are out there and many methods in which people train them in has increased my knowledge exponentially. Furthermore, my knowledge in cyber security was similar but deeper due to my interest in it. My ideas of how we can make IDS better with AI is like those in papers but I’m able to gain an understanding on what I should be aiming for. The relevance of AI models in this project is essential as they will help with the analysis and processing of data more efficient and increase pattern detection.

Based on the papers mentioned there should lay a visible picture of how we can use Ai in cybersecurity. Before discussing any gaps it’s important to note a term in machine learning algorithms ‘no free lunch’ which states that there is not one machine learning algorithm that is better than the other , the theorem states that if any algorithm (A) outperforms another algorithm (B) in a desired task then algorithm (B) will outperform algorithm (A) in many other task (Yang, 2020). While this report may focus on finding the best algorithm to use and dataset it is theory and opinion based.

In the first (supervised) paper, their use of the ML algorithm DNN and KDD cup 99 dataset was shown to be effective in having a high accuracy and detection rate. Based on their results its shown that within using the Machine learning algorithm DNN and the training set being split in such way they came to a result that for the highest accuracy an attack packet rate would be 50%,70% and 80%, for the highest detection rate it is 80% and 90% and for the lowest false alarm it was attack packet rate of 10% and 20% (Kim, 2017). With this result it could be assumed when specifically trying to train and model the IDS to detect attacks that the idea attack packet rate should be tested at 80% with the remaining 20% being normal as this resulted in the highest accuracy and detection rate. If I was using the same algorithm based on the intended project, I too would use this, only thing that I would consider is the attack packet rate with the lowest false alarm as that also takes part into increasing accuracy. The uncredible aspect of the paper is the dataset they are using which will be explained later (Kim et al., 2017).

As for the second paper unlike the other experiment using the tool Sensor Flow, they used a different tool to develop their model called Azure Machine learning tool while using a validation method for 10-fold training and testing and the algorithms they had was SVM, random forest and decision trees. Already there is a noticeable different in the machine learning algorithms used between paper 1 and paper 3 but also the tool in which they used. After their experiments concluded they had got the results that out of the three algorithms SVM had the highest value of accuracy scored which was 98.18% (over 10 test) with RF following with a score of 96.76% and then finally DJ with a score of 96.50%. they also concluded that with a split of 90:10 they had an accuracy of 100% which is likely impossible with datasets today. In normal circumstances I would have trusted this experiment to aid in the development of this project, but main disadvantage of this paper is once again the dataset used. However, the conclusion for the best split to get the highest accuracy was 90:10 which is also like that in the first paper. The same was done for the attribute precision results and recall rate with SVM being highest for precision result and RF being the highest for recall rate. After receiving their results, they were able to conclude that SVM had the best overall results and hence should be used to detect an intrusion in IDS. I agree for the most part that SVM is a good model to use due to its advantages as being a supervised model and more advantages such as its resistance to overfitting however its disadvantage of being expensive and time-consuming does play a factor into the development but not that much of an importance. Furthermore, due to the dataset used I must critique it and wonder if the same experiment was to be done on a different dataset what would the results be then (Azizan et al., 2021).

In the last supervised paper, their use of training the model is different as they are using deep neural networks which is more suitable for large datasets, and they use the technique backpropagation, which is fast, simple and easy to program. it has minimum parameters to tune apart from numbers of input and is a viable method to use due to its flexibility in the fact you don’t require prior knowledge. Also, unlike the other papers they utilised different datasets which would have varying different data but still overall compared to the first two papers it can be said that this experiment holds more valid results when comparing to how network systems are nowadays (Shun and Malki, 2008).

As for studies utilising unsupervised learning compared to the studies using supervised learning, they utilise different features and methods. For example, in the first paper they use auto-encoders something that’s not mentioned or touched upon in studies with supervised learning. Furthermore, they use a different machine learning algorithm unlike those in other studies ELM which is a supervised machine learning but they with the aid of other features were able to implement an Unsupervised ELM. One thing to be duly noticed is in their experiment they classified the five different types of attacks as a single attack category. This made an interesting idea as the idea of this would seem to spend less time training against one type of attack and instead focus on the differentiation of attack behaviour as one. Only wonder in this is would it prove the same similar result if the attacks were not categorised as one but tested individually despite it being time-consuming. Another thing is the authors of this study mention that the advantages of feature extraction with un-supervised deep learning approach is not only faster calculation but better representation of feature learning (Alom and Taha, 2017).

Furthermore, their use of evaluation by evaluating machine learning algorithms against each other was like those in supervised except the fact that they compared the algorithm model US-ELM to three other them being: k-means, Auto-encoder + k-means and RBM + k-means. With this they were able to conclude that the deep learning approach of RBM and AE with k-means clustering showed the highest accuracy of 92.12% and 91.86% respectively. Comparing these results to those seen in papers studied these are considerably low considering other studies with other methods have been able to reach an accuracy of 99% or above (Alom and Taha, 2017).

In the second paper by (Tuor et al., 2017) they use the model DNN and RNN which as mentioned are trained to recognize activity that is characteristics of a user on a network, in other words it can mimic and make decisions that a user would make after learning their pattern, I find this to be a suitable mode to use due to the design of the neural networks being created to act like a humans brain in how it processes data. Like the first paper they compared the model against those of supervised model and once again showed that for their desired problem and solutions that an unsupervised learning approach was better. In one of their experiments, they concluded that with a minimum daily budget on recall that DNN-diag and LSTM-diag and isolation model forest obtained 100% recall its possible its due to the difference in the study in how they are detecting insider behaviour, but this paper doesn’t seem to mention the accuracy in which it was able to detect and to what effectiveness.

Finaly for first paper that based around a semi-supervised one thing that’s good is they mention the parameters in each model they used and describe the use for them such as the Num epochs which defined how many times the entire dataset was iterated over during training. As already mentioned, the highest achieving model was the PBN model that was semi-supervised compared to the other models did increasingly well as it had a 98% accuracy compared to the other models having 91%-96%. The author furthermore mentioned how that f1 score received from the SSL-PBM indicates that the model effectively balanced precision and recall which is highly crucial for minimizing both false positives and false negatives in cyber-attack detection. This is important as to increase accuracy you need to decrease the false positive and negatives as this paper aims to Solve (Kolawole Ayodeji et al., 2025).

Following on to the second paper there is a lot of learning methods introduced and their method of testing the data was different , similar to those in papers where they used neural networks and would do a back propagation they too followed this method with the k-fold cross validation they mentioned that since they wanted to use a limited number of labelled data the folds were to be swapped hence why for example in a 10-fold cross validation , the classifier would be trained on 10% data while being tested against 90% data instead of the usual 90% training and 10% testing (Noorbehbahani and Saberi, 2020).

You may have noticed in the a few of the papers mentioned in my report they utilised the KDD dup 99 data set in some shape of form when training their model which at the time was widely used in security research. While it may have been good at the time of previous experiments there lies issues within using this data set in the current modern day. It is bad nowadays because of a few reasons such as 1) it’s too old 2) statistical analysis have found it to be inadequate and 3) the main reason why it’s not smart to implement and use this dataset to train models is because it was created by the 1999 USA air force base which is not a true representative of businesses and/or networks in 2024 (Stallings et al., 2008) Hence why both papers are evaluated harshly on their use of datasets. So, following into this report I will not be using the KDD cup 99 dataset, but instead more reliable and representative datasets mentioned in the requirement stage. Following on it was beneficial to see two different experiments with different techniques and models the only negative was both papers used a now invalid dataset. The challenges that I may have to solve beforehand is of course finding a good dataset that can represents a modern-day network system.

Besides the few similarities and differences between the papers there wasn’t much that wasn’t covered however the small gaps that should be mentioned is the first paper seemingly only did one test with each split training dataset whereas the third paper did 10 this may have been because of the evaluation method they were using which is the 10 fold training but having multiple test to get the average accuracy was more valid than the first paper of juts doing one test. Following from this, the project will address the possibility of running multiple tests for the split training dataset to reduce random errors furthermore increasing the accuracy of results ensuring the accuracy received can be counted as a valid and valuable.

# Requirement Analysis and Methodology

## 3.1 Requirement Analysis

As previously mentioned, the intended aim of this project is increasing the accuracy of IDS by however means necessary based on literature review there has been a deeper understanding on sub sections to progress the aid of increasing of accuracy.

Based on the literature review conducted there are many more implications and requirements that will be needed to be considered.

### 3.1.1 Objectives

Overall, my objectives for this study are

* Minimise the possibility of false alarms ensuring a more reliable outcome and increase in accuracy - I expect that upon training the algorithm it will not be time consuming (may depend on the algorithms chosen) but I expect it to have a high accuracy
* Train the machine learning to adapt to new data and emerging threats. - what I require of the model is for it to be able to work efficiently in any IDE showing worthy results. To do this the need for a user-friendly interface IDE will be beneficial.
* Get a high accuracy from the models trained
* Shown the benefits of integration ML with cybersecurity
* Finally produce clear results allowing for it to be evaluated.
* After training of model, I will have integrated it into an open-source IDS

### 3.1.2 Model requirements.

Depending on the nature of my problem this report tries to solve I will need to choose a model suitable. Therefore, a classification approach is needed.

Furthermore, depending on the model’s complexity, it will require powerful CPUs or GPUs to have the highest processing speed possible when training against a dataset especially if large. Furthermore, memory and storage will also be required to allow for the model to train to its highest potential while retaining a lot of data needed. Lack of these will severely disacknowledge the capability machine learning has to process and train.

Similarly depending on the model’s requirement and chosen programming language, Frameworks and libraries will be required to help simulate an essential programming environment for the model. It will be best to use known frameworks such as Pytorch and TensorFlow that are popular for deep learning and are advanced in advantages such as highly flexible or other frameworks such as sci-kit Learn which works well with python and is the top framework for data analysis. Additionally having good libraries like Pandas will also help with data manipulation and analysis. Having these is libraries such as these or more is necessary for the completion of this project

### 3.1.3 Data requirements

The next is the data requirements. Data requirements include studying data to extract useful information for intended purpose, in the case of our project this just simply entitles that the data we use must be validated and reliable including its data types. For example, a lot of papers researched seem to mention the KDD cup 99 dataset and as already implied while it may have been good at the time it is now irrelevant due to its difference in modern day networks. Furthermore, other papers researched seem to have needed to pre-process the dataset, which is expected, and convert data types to the same category such as changing real(float) into integers (sachdeva, 2010). As to improve from this my dataset will be different as it will be mathematically vigorous and will be more of a representative of modern-day network as we are aiming on enhancing our modern-day network security for future generations.

Furthermore, the dataset will require preprocessing as there is expected to be many outliers and missing values and if these are the remain depending on the model algorithm, I use it can make the model be oversensitive to outliers hence making the results inaccurate.

### 3.1.4 Evaluation requirements

Evaluation metrics. This typically concludes criteria’s such as effectiveness and its efficiency and impact etc. upon reading one of the papers ‘method of IDS using deep neural network ‘ by (Kim, 2017) I noticed that the aim of their project was similar to mine except theirs was more deeper in the sense they were not only testing the accuracy but also the rate of false alarms and the rate of detection attacks. This made me rethink my initial aim of the project which original was just focusing on accuracy and now must consider how I may be able to measure the recall to further evaluate how well the model I created is. Reason being is while similar it helps look at our model from a different angle such as the model’s rate of detecting attacks and how accurate is it when it does detect it. Implementing this into the project should make our model more viable and easier to evaluate on whether it has succeeded or not.

To do this this project aims at the possibility of using the following evaluation scores:

**Confusion matrix** which is a table that shows how well a classification-model performs by comparing predictive values to actual values and is used to evaluate accuracy and effectiveness of a model’s predictions. The table will contain the values for true positive and negative and asl false positive and negative in which this project aims to reduce.

**F1 Score** which is used to get the best precision and recall at the same time.

**AREA under the curve** which can further back up the confusion matrix and visualise between the sensitivity and false positive rate**.**

### 3.1.5 Functional and technical requirements

Functional requirements essentially mean what would my model need to have done to succeed in its intended purpose and in this case of using AI and ML with IDS, the artefact will need to have successfully detected abnormalities in a network with a minimum of 90% (more preferably 99%) accuracy and will have needed to have shown a decrease in the number of false alarms. For the technical side these are issues that will need to be addressed alongside the training of the model such as the performance of the model and the reliability and the main thing if that throughout the duration of the training period that the model is secure and not authorised by malware/malicious actor so that the model is not purposely fed mis-accurate or wrong data. In the case of this happening, it would make the whole training invalid and could lead the model to learning the wrong things.

### 3.1.6 Non-Functional

Non-functional requirements that I will need are to focus on ways I can improve evaluation measures. Even after using evaluation such as comparing accuracy with other models, it still won’t be sufficient, hence like in other papers this project will need the following to be considered an effective study.

* Other metric formulas such as detection rate and precision.
* F1 score
* ROC-AUC Curve
* K-fold cross evaluation

### 3.1.7 Challenges and Considerations.

Potential challenges that would arise from this and what solutions would be required. This suggests obstacles and/or difficulties that would arise. For example, one obstacle could be the initial start-up of teaching the model and finding the right dataset and choosing which machine learning algorithm is the best to use. In the case of any challenges like this the model would require extra focus and time spent on. Another could be figuring out how to integrate the trained model into the IDS.

A consideration I also need to keep in mind is ethical concerns as my dataset doesn’t rely on human data there isn’t too much to be worried about but there is still the concern of staying with the guidelines as the data I use was collected from somewhere and was only possible due to human activity.

Overall, there are many requirements that this project will have but they will be dealt with throughout and extra care will have been taken to possibly prevent before it arises.

## 3.2 Methodology

This section contains a mini step by step on the plans I took and why and how. For this experiment I intend to use a free open source and friendly user interface software IDE, there are many options out there such as TensorFlow or Jupiter, but I plan to use ‘spyder’ a lightweight IDE beneficial for data preprocessing but also training the model, that is in ‘Anaconda navigator’. Anaconda navigator is an interface that allows me to interact with many Integrated development environment like spyder.

### 3.2.1 Data Collection

First step is Data collection, data collection involves me gathering and measuring the usefulness of data, in other words I will use a dataset. As for the dataset that I will be using I plan to use a databased found online with already collected data as this will be less time consuming for me. Furthermore, when picking this dataset, I will also have to check the structure of the data as some datasets I have seen have readily data to be trained and tested on whereas some do not. However, even though the dataset may be open to public and free and already collected doesn’t mean that it will be perfect for the experiments I plan to do. At first, I looked at papers already mentioned to see how they found their datasets but there was limited or no information at all related to the discovery of location of their datasets; after looking through the internet and learning materials on hand, I was able to find a selection of datasets that varied. One was the obvious KDD cup data set, but the others were newer datasets with interesting attack types.

Finally, after getting the dataset, I needed to do some preprocessing which means the following were done:

### 3.2.2 Explanatory Data analysis

The first step in data preprocessing is knowing what is relevant and what is not needed. Data analysis is the process of examining data to make in-formed decisions in our case the data has already been collected and slightly prepared, but I will still need to do a diagnostic analysis check to see if there would be any data that could cause unnecessary errors or noisy distractions. But this step mainly would consist of reading over the data to know what I would like to keep and what to get rid of. To do this, I had simply read whatever information was provided such as documents containing descriptions of features or stats and furthermore, I had a look in papers to see what approach they took.

One main thing that helped was reading past papers and talking to experts in this field on the type of data that is commonly used and discarded as noisy, this was an important factor in the development of the artifact.

### 3.2.3 Feature selection

The next thing I did after the data analysis was feature selection. This is the process of isolating the most relevant features, non-redundant and most consistent. The goal of doing this was so I can identify the minimum number of features that I should keep that will be able to provide the highest predictions upon training with model. Further benefits of this will be an improved accuracy, faster computation time due to less data being trained and a reduction in overfitting which is eliminating unnecessary features so future models won’t be distracted by noisy data.

My first action was to Refer to the literature review mentioned in a paper by Shun (Shun and Malki, 2008) in which they talk about cutting down on the features presented in the dataset. For their case they only removed a few columns that they thought unnecessary as they knew it could be noisy data.

As their study is very similar to mine there is undoubtedly the exception that I will also be using if not the same features as them but very close, after all it depends on the dataset I plan to use.

After further studying the data set, I did remove a selection of features that I deemed unneeded. The way I went about this was looking at the description of the features and what their purpose was for and if I felt they added no contribution to the project I would remove them.

### 3.2.4 Cleaning and preprocessing

Final thing in the stage of the preprocessing is just the cleaning selection of the dataset. This will typically come at after the feature selection as the only columns that remain will be what was desired. For me Cleaning will consisted of things such as

1. looking for empty values and either choosing to replace them with Null or completely get rid of the column if the empty values persist too much
2. Deeper looking into the data and ensuring there’s no extraordinary data present or just invalid data such as random symbols being present where an integer was meant to be. One thing I will have to consider is due to the type of data I am using there will be random changes in the data values which could represent attacks on the network or other valuable information, so I had to be careful
3. Looking for duplicate values, searching for any duplicate values in which I find unnecessary. This is another thing that will be proceeded with caution as its certain that some rows will have duplicate data for example source Ip and destination will typically be the same as these will represent were data is being sent to and where it’s intended to be received. Second example can be source bytes which represent the amount of data being sent on a network and this can vary but also remain the same.

Other words those are the main things that I had in mind to do when it comes to preprocessing my dataset. Furthermore, there is one optional technique that I may implement depending on the dataset and that is to just make sure there Is two categories that the model is looking for, normal and attack network. As shown in a lot of papers, one example being from Jin (Kim et al., 2017) they train their model to look for the two categories attack and normal. Originally, they had a dataset of multiple types of attacks but for the purpose of training they categorised it all into one name. Doing this could allow for a lower amount of computational time.

### 3.2.5 Machine learning

After all of that, the preprocessing of the dataset should have been completed and now ready to be trained. Now began the research of machine algorithms models to train with the dataset. I had the desire to use three machine algorithms and upon researching there was lots of options to choose from based of advantages and disadvantages of models. However, in due time I originally concluded that I was going to use the following models.

* Random forest
* SVM
* Neural network

My idea for using these models were mainly based of papers researched in which a lot of them typically used these types I trained all three algorithms with the same dataset for the process of using them to further evaluate my results.

Random forest and SVM are simple in its process however when it came to neural networks there was a lot more steps, I must do for this. When it comes to neural networks, they are more complex as they represent the human brain and can think like one and see patterns, the way they do this is by their structure which consists of three layers of which I could manipulate:

* Input layer – receives data
* Hidden layer – process the data
* Output layer – outputs the data

Manipulating the hidden layer, I was able to get different results. Furthermore, they use a linear regression model. Data is passed between nodes in a feed forward.

However, one thing to mention is the need of having to change one of the machine learning algorithm models, I changed my original plan of using SVM to decision tree instead as the dataset I was using was large and this severely limits the capabilities of SVM hence why I changed it to random forest.

### 3.2.6 Evaluation measures

When it came to the evaluation of efficiency of the model, I planned to use many evaluation formulas such as

* Accuracy - which is a measure that shows how well a model performed by correctly identifying outcomes
* Precision - A measure used to show the prediction of positive instances made by the model, can measure false positives
* Recall – A measures that shows the correct identification of all positive instances, can measure false negatives
* F1 score – combines the recall and precision score of the models to give a truer accuracy of the model’s efficiency
* Receiver characteristics operating - Area under the curve – represent the probability that if a model is randomly given a chosen positive and negative example it will rank the positive higher.

Using all that mentioned above I planned to further show the true effectives of my model. Furthermore, I used evaluation techniques such as:

* K-fold cross evaluation – I will divide my dataset into several k-folds and use to assess the model’s ability in the case of new data being available.
* Leave one out evaluation – a technique that is computationally expensive but ends in reliable results, only appropriate if I have a small dataset. This involves using each individual data point as a test set once while using the remaining data points to train the model

Reason I plan to use a lot of techniques such as those is because having accuracy alone is not a good way to evaluate the model, it could say it has a high accuracy while meanwhile severely lacking in the other areas.

I was able to do all the above to the best of this study after using informational websites on how to code and display the results. Furthermore, I was able to use other evaluation techniques such as a cross-matrix evaluation which helped depict the True positives against the False positives and further analysis false negatives and true negatives.

### 3.2.7 Hypothesis:

Going into this study I had an assumption that one model would significantly outperform the others. My assumption was amongst all trivial machine learning models that the neural networks, despite the type of neural network used, will show a significant result compared to the other machine learning algorithm, reason being is because of its ability to mimic a human brain and its capabilities of pattern detection. And based of research found in papers I could have this strong assumption that I was right. But I was not.

# Design of Artefact

The report should have one or more chapters describing the design and development of the project solution/artefact. These may include formulation of scientific questions and the answers to them, appropriate theoretical background, technical problems considered, methods used to solve them (methodologies and tools employed e.g. case tools), discussion of issues arising in specifying, designing, and implementing the system (e.g. requirements analysis, user interface, system architecture, algorithms, major data structure, etc.) and evaluation of results (e.g. complexity, efficiency, user-friendliness, reliability, etc.).

This section contains a detail step by step of the process taken. (I will show each action taken in processing the data, code snippets of what I did to make the model train Using spider to help preprocess the data.)

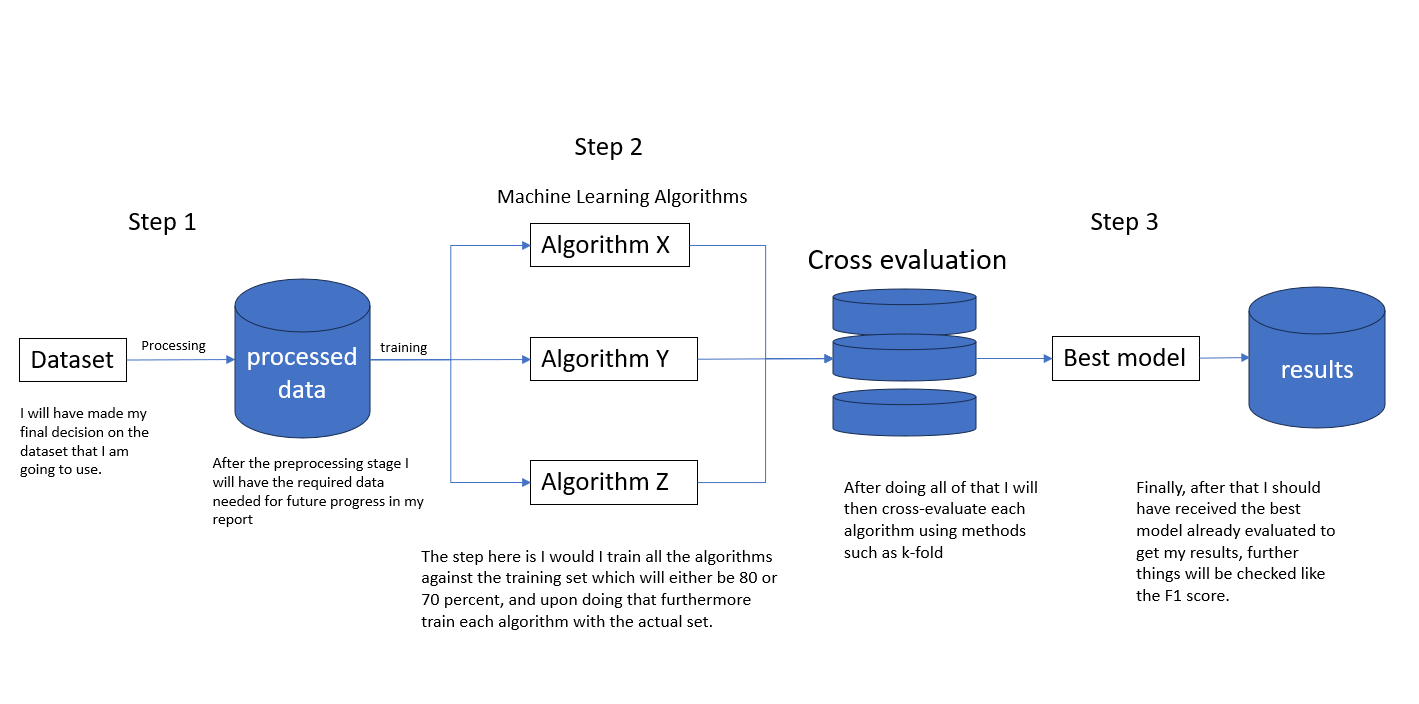
This section of the report contains a detailed step by step of the proposed plan that will be taken to meet projects aim and objectives and furthermore the decisions made to carry in each step. Below is a diagram created to help visualise.

Figure 1: flowchart of proposed plan

flowchart of proposed plan

### 4.1 Finding the Dataset

The first step in this project is choosing a suitable dataset. a dataset is the collection of related data that’s stored together for analysis. In the context of this project a dataset that contains data on network systems such as source Ip and destination or network flow traffic and types of attacks was needed.

Upon researching for related datasets, I came to three decisions on databases to use:

#### 4.1.1 NSL-KDD dataset

This dataset was updated 6 years ago by (zaib, 2019). Upon looking at this a few things had been noticed such as the language used in this was python which isn’t an issue at all, it just entailed that I had to think of my preferred coding type which is probably python or java. Secondly, they use multiple algorithms which are SVM, decision tree and random forest and briefly mentioned neural network. And finally, the supposed dataset they use is KDD cup which already mentioned is not suitable as its old and not a representative of today’s networks. Based on the dataset alone this no longer became a valuable decision, and this left us with the decision of the other two decisions.

#### 4.1.2 UNSW-NB15

This Dataset was created by the IXIA Perfect storm tool for generating a hybrid of real modern activities and synthetic contemporary attack behaviours. Contains 9 types of attacks a few in which this project would aim to reduce.

The total number of records is 2 million and 540,044 which are stored in four CSV files. Seemingly a training set and testing set has been created and within the training containing 175,341 records and the testing contains 82,332 from different attacks and normal. However, it was apparent that if the training and testing had irrelevant features it would not be beneficial.

#### 4.1.3 TON\_Iot

This dataset is said to be used for validation and testing various cybersecurity applications-based Ai such as malware detection and the obvious IDS.

It contains various attacking techniques such as Dos and DDOs and ransomware. Preventing these would be valuable considering the development of IOT and how lots of data is being backed up IOT cloud. Preventing these will be highly beneficial and due to this became a high decision of this project.

Overall compared to the three decisions with two being tough it was decided that the dataset to proceed with would be the TON\_Iot due to its reasons such as not only did it contain attacks that became preferable to try and detect and stop early but it was one of the newest and popular datasets yet compared to the KDD.

TON\_IOT dataset

The TON\_IOT dataset by (Moustafa, 2021) is a dataset used for evaluating the fidelity and efficient of different cyber security applications.

The dataset contains five directories:

1. Raw datasets – within this are four folders for Iot datasets, network data sets, Linux datasets and windows data sets, and furthermore within each of them CSV files are found
2. Processed datasets – four datasets compiled and processed in the form of CSV files.
3. Train/test datasets – this folder contains samples of four datasets in a CSV format that were specifically selected for evaluating the fidelity and efficiency of new cyber security app-based AI and machine learning algorithms.
4. Description of stats- folder includes the description features of the four processed datasets and statistics.
5. SecurityEvent/GroundTruth datasets – includes events of hacking happening in the datasets and timestamp it occurred.

Upon looking through each directory for this study I decided best to use the train/test dataset already provided, reason being is this contains all records of all attacks compared to the many processed datasets that only contain individual attacks. Despite being readily made there is still quite a bit of data preprocessing that is required.

It must be noted that for use of this dataset the following papers must be cited:

1. “ [A new distributed architecture for evaluating AI-based security systems at the edge: Network TON\_Iot datasets](https://www.sciencedirect.com/science/article/pii/S2210670721002808?casa_token=IlzSGltIsZkAAAAA:K-oImykBBkUVp-WRfhXB1TsUtZXRjZlrSSon-YStTEN8EyKOO5apRzZtb25yphNsm0PCWffQIjw).” (Moustafa, 2021b)
2. " [TON\_Iot-The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion datasets](https://ieeexplore.ieee.org/document/9444348/authors#authors)." (Booij et al., 2022)
3. "[TON\_Iot telemetry dataset: a new generation dataset of IoT and Ilot for data-driven Intrusion Detection Systems](https://ieeexplore.ieee.org/document/9189760).” (Alsaedi et al., 2020)
4. “[Federated TON\_Iot Windows Datasets for Evaluating AI-Based Security Applications](https://ieeexplore.ieee.org/document/9343133)." (Moustafa et al., 2020)
5. "[Data Analytics-Enabled Intrusion Detection: Evaluations of TON\_Iot Linux Datasets](https://ieeexplore.ieee.org/document/9343084)." (Moustafa, Ahmed and Sherif Sayed Ahmed, 2020)
6. "[New Generations of Internet of Things Datasets for Cybersecurity Applications based Machine Learning: TON\_Iot Datasets](https://conference.eresearch.edu.au/wp-content/uploads/2019/08/2019_eResearch_59_New-Generations-of-Internet-of-Things-Datasets-for-Cybersecurity.pdf)." (Moustafa, 2019)
7. "[A systemic IoT-Fog-Cloud architecture for big-data analytics and cyber security systems: a review of fog computing.](https://arxiv.org/abs/1906.01055)" (Moustafa, 2019a)
8. "[IoTBoT-IDS: A Novel Statistical Learning-enabled Botnet Detection Framework for Protecting Networks of Smart Cities](https://www.sciencedirect.com/science/article/abs/pii/S2210670721003255)." (Ashraf et al., 2021)

### 4.2 Preprocessing the data

The next action to take is the processed of removing any unneeded or unnecessary features for example in the dataset there are mini datasets containing data on household appliances connected to Iot such as fridges or garages and in the case of this project we focus on increasing the accuracy of IDS not on implementing it in real life application like those mentioned hence things like that will need to be removed to ensure no confusion in training the model and no outliers.

This step consists of making sure the dataset is as clean as I require so the machine learning algorithm model can learn to the best of its ability with no disruption of outliers or noisy data. This project uses the Anaconda Navigator which is a desktop graphical user interface thar allows us to launch other applications. Within this we also use Spyder, a free and open-source environment for python, to help preprocess our data.

I also imported the following:

Pandas

It’s important to mention that the dataset is one and so when it comes to training the model I will be splitting the dataset into 80% and 20%, 80% to train and 20% to test.

The table below shows the few things I plan to check:

|  |  |  |
| --- | --- | --- |
| Action to be taken: | Steps to be done: | Expected output: |
| Removing unnecessary data | I will be removing what I deem to be unnecessary for this study such as unnecessary columns | This should remove many of the noisy data that the model doesn’t need to learn furthermore increasing accuracy and precision |
| Cleaning the data | Will be looking for any duplicates or missing values and delete them. and in the case of seemingly unusual and different values compared to others I may get rid of them too | Removes the possibilities of outliers |
| Optional step: data types conversion | Converting all the 9 different attack names into one category so the model will be looking for attack and normal behaviour. | Hopefully ensures a higher accuracy and recall when training the model to distinguish the behaviour between normal and attack |

Table 1. Stages of Data Preprocessing.

#### 4.2.1 Removal of unnecessary data

The first task in this was deleting the columns I felt were unnecessary for my required project.

To do this I used the following code:

Figure 2: code showing how I separated my features

Using this code enabled me to drop all the unnecessary columns I didn’t want while keeping the ones I did by specifically listing the names, this was way quicker than typing out all the features to drop

As you will see in the figure 3 shown below, I kept the most common attributes of a network log.

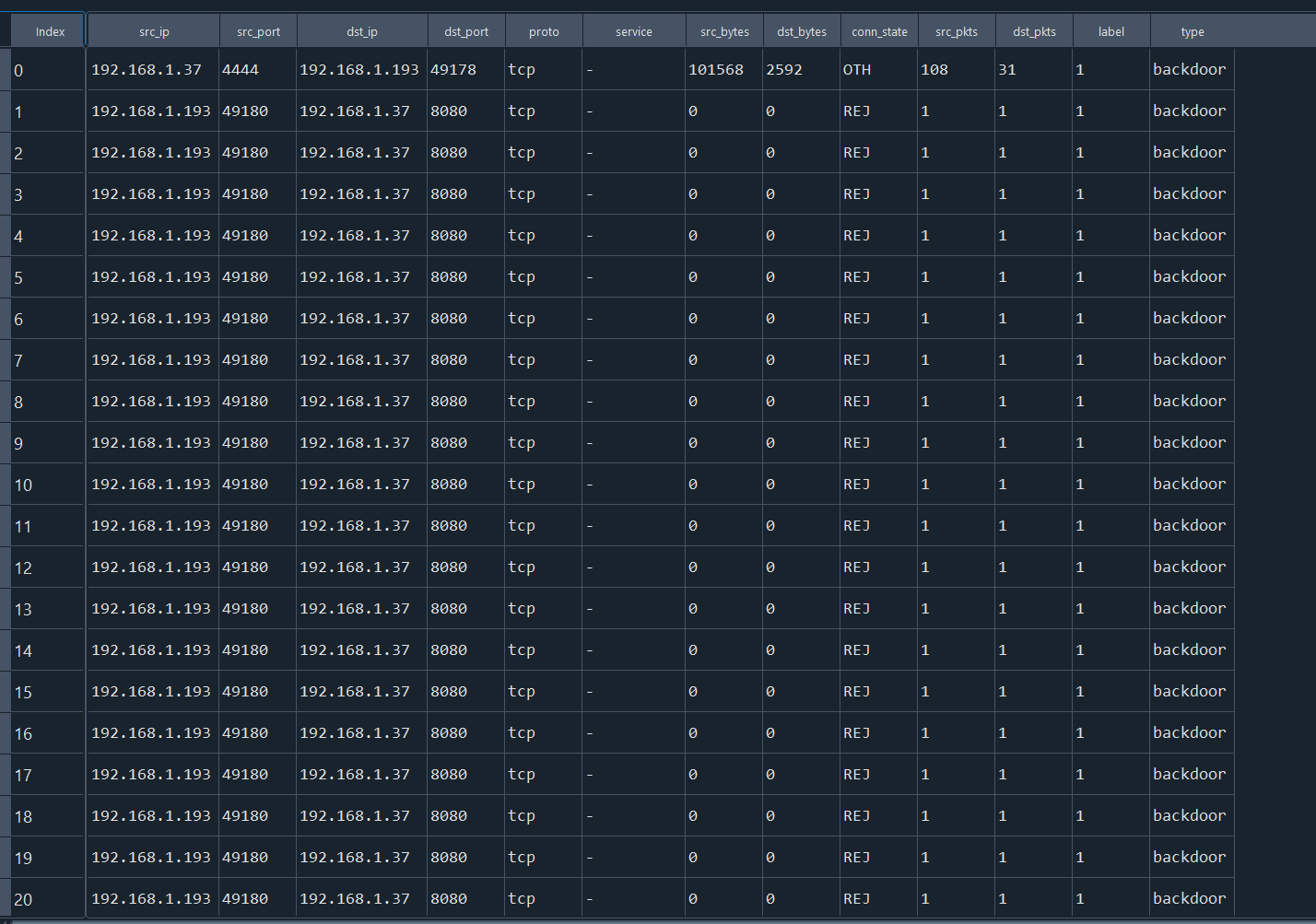


Figure 3 : image of dataset after features Selecetion

The features kept were source Ip and port, destination Ip and port, protocol, service, source bytes, destination bytes, conn\_state, source packets, destination packets, label and type. Leaving only 13 columns left out of the original 44.

#### 4.2.2 Cleaning the data.

For the next part this consisted of finding all empty or duplicate values, I had to consider when finding the empty values if I should replace with the variable null or just drop the row (column if too many) however I figured it was best to just replace all the values in the database that was empty with a variable to represent Null as shown below.



Figure 4: code showing removal of empty values

Furthermore, when it came to duplicate it was exactly as predicted that there would be many, and this is due to the type of dataset I am using so this required no further action on deleting duplicates as it would have a highly drastic change on the results of our experiment

#### 4.2.3 data/type conversion

This was an optional step to me as it wasn’t entirely required for the development of the model, however I came to the assumption that not only would it look neater, but it would be easier to train on the model to just give it two categories to look for. Therefore, I converted all the name attacks into one category called ‘Attack’ as shown below.

newdf = newdf. replace(['backdoor','ddos','dos','injection','mitm','password','ransomware','scanning','xss'], "Attack")

Doing all this I kept in mind the single responsibility principle when coding which just ensures that even if the dataset was to be changed minor or no code would have to be changed at all.

### 4.3 Machine learning algorithms

After processing the data, I will now need an algorithm (for our project three) to train against the processed dataset. Just like with the dataset there were many decisions that had to be made especially now considering we know the type of dataset we plan to use. It became beneficial to pick algorithms with benefits based of the dataset used.

#### 4.3.1 Neural network

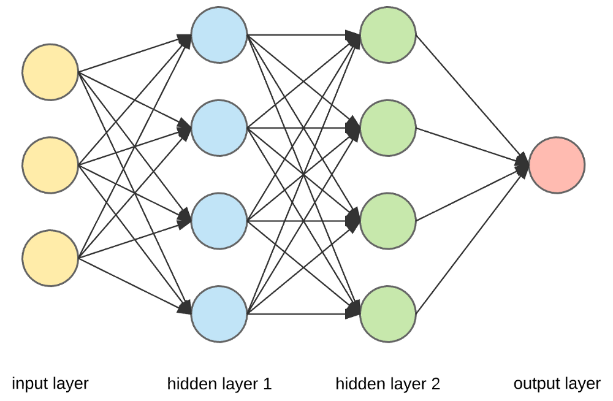
Neural network Created and inspired in 1943 by Warren McCulloch based on the structure and function of a human brain and can be used for both classification and regression.

Figure 5: Structure of neural network (Taken from google images )

An Advantage of using this as one of the algorithms is that it can learn complex patterns and is flexible and furthermore is highly suitable for large datasets. I chose this model due to its advantages on being able to perform well on large datasets and its ability for prediction.

Neural network structure is more complex due to having three main layers, an input, output and hidden layer

#### 4.3.2 Random Forest

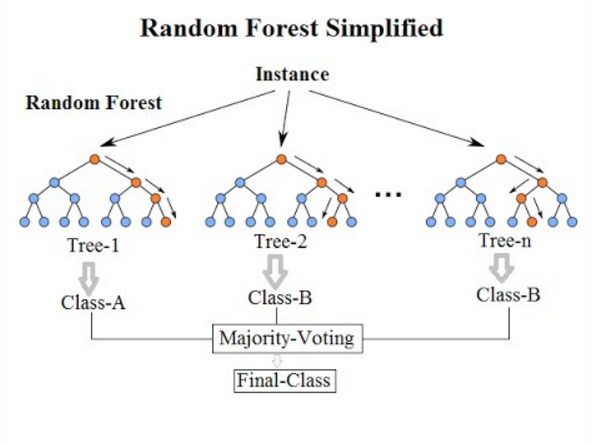
Random forest is a collection of trees, decision trees. once again used for classification but also regression and this helps ensure the accuracy of predictions and decision making. This algorithm is useful when making decisions about how to carry out a project or plan.

Figure 6: Structure of Random Forest (Taken from google images)

An advantage is it works well with datasets that have missing values and reduces risk of overfitting. But one worry with this is that its slow when predicting and harder to interpret and requires more memory.

#### 4.3.3 Decision tree

Primarily used for classification and predictions based on one central piece of information. They help visualising predictive models by showing potential outcomes and pathways that come from making an initial decision.

The advantage is low computational cost compared to random forest hence faster training and prediction and can handle variety of input types.

You may notice that this has been changed from SVM to decision tree, the reason being is as already mentioned I now know what dataset I’m using, and the dataset is a large one, which is a disadvantage when it comes to SVM, hence why I have chosen to substitute it out for this.

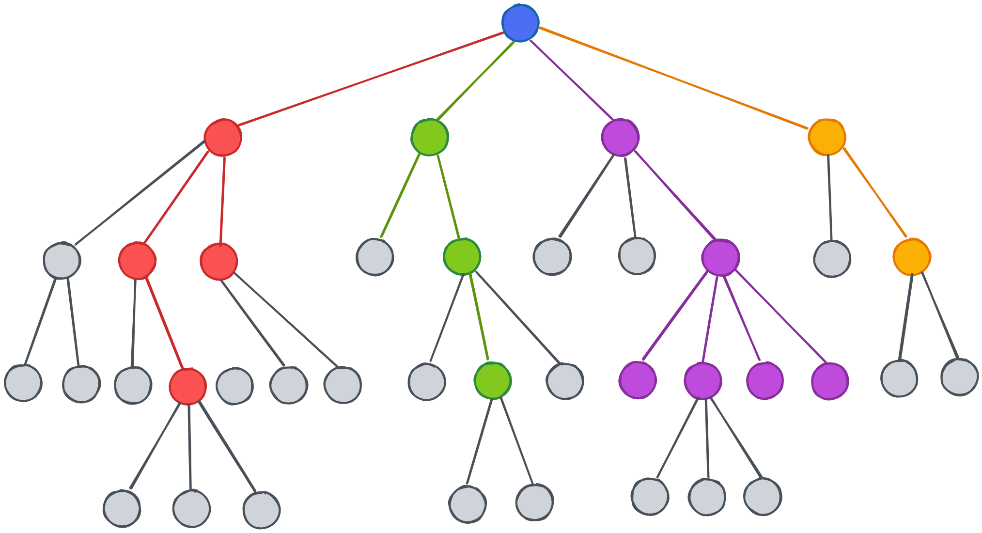


Figure 7: Structure of Decision Tree (Taken from google images)

Decision tree work by following a path of decisions as shown in the structure to the right. The idea is it will follow the rules but also the best outcome until it can reach a decision

# Development of Artefact

This sector contains documentation of the testing of the machine learning algorithms with the processed dataset. The intended purpose of using multiple algorithms is of course to see if AI can be used to efficiently increase accuracy and furthermore which type of machine learning is more suited for this problem.

The overall aim for this is to use a combination of attack and normal traffic data to test against the machine learning algorithms, the dataset as already mentioned will be split int 80/20 in which 80% is sued for training and the remaining for test.

To have trained our models I had to do the following

* Installing and opening Anaconda Navigator
* Downloading the IDE Spyder.

### 5.1 Decision tree:

After preprocessing the data set it now came time to train the models. The first being decision tree.

##### 5.1.1 Importing of libraries

Importing the following libraries is necessary for the development of this model.

from sklearn.tree import DecisionTreeClassifier

import pandas as pd

from sklearn.tree import plot\_tree

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix,ConfusionMatrixDisplay

from sklearn.metrics import classification\_report

importing these libraries allowed for us to do things like:

* importing the model, we want to train
* allowing us to select our train/test Split
* allowing us to have evaluation measures such as accuracy score and f1 score.

##### 5.1.2 Development of model

This sector contains the code I used to develop the decision tree algorithm model. My first step was to define the features in my datasets that I want to represent by X and the target variable represent by y.

X = newdf[['src\_ip', "src\_port", "dst\_ip", "dst\_port", "proto", "service",

"src\_bytes","dst\_bytes", "conn\_state", "src\_pkts", "dst\_pkts", "label"]]

y = newdf ["type"]

In my study I set my target variable to be the ‘type’. This consisted of two different things , whether it was a normal traffic data or an attack traffic data and as shown above I used all the other variables as the features. Of course, if the result seemed unreliable, I would wonder if I was left exposed to overfitting.

My next step was splitting the dataset into the desired training and testing and furthermore deciding the maximum depth ( how large I want the tree to grow ).

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size = 0.2,random\_state=42)

tree\_clf = DecisionTreeClassifier(max\_depth=2)

tree\_clf.fit(X\_train, y\_train)

The figure above shows the code I used to split the dataset into 80% training and 20% testing. You may notice that there is an additional code in which there is a random state = 42 , this is to help control the shuffling around. Also, the max depth that I use is 2 but of course I can change around if I am not happy with my results.

The next thing I did was optional but still beneficial, I planned to plot the decision tree to show how it works.

column\_names = newdf.columns

class\_unique\_values = ['0','1']

plot\_tree(tree\_clf,feature\_names=column\_names[[0,1,2,3,4,5,6,7,8,9,10,11]].tolist(),class\_names=class\_unique\_values, filled=True, rounded=True)

plt.show()

I once again defined the column names and unique values and in the plot tree code I designed how it should plot it 0:11 are the x features that I wants to show, and the class unique values is the target variable

Final step in this was making my prediction.

y\_pred = tree\_clf.predict(X\_test)

Using this simple code, I could make a prediction upon my test

Upon trying to run my first test I came across my first problem, an error displaying the following: “could not convert string to float: '192.168.1.31'”. This was a problem that I had to overcome to progress further. The reason it was a problem was because the decision tree classifier doesn’t accept categorical values. Upon trying many solutions such as using one hot encoder and others I was still unable to come to a fix this being the Ip address could not be formatted in the required format. To overcome this, I took a step back.

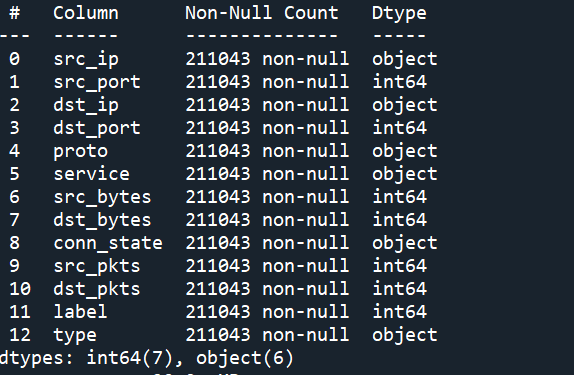
Going all the way back the data preprocessing I had to make some changes due to the error being the Ide was unable to convert strings to float.

Figure 8: image of features and their data types

As shown in the diagram to the right here was all my original features I chose during the preprocessing phase and as you can see 7 of them were int64 and 6 are objects in which the problem exists.

To fix this decided to take a slow approach and work my way backwords. My first step was to get rid of all the columns that were objects excluding the target variable.

##### 5.1.2.1 First experiment

Upon doing so the error had been gone and I was able to plot the model on a graph and furthermore I was able to get receive some results.

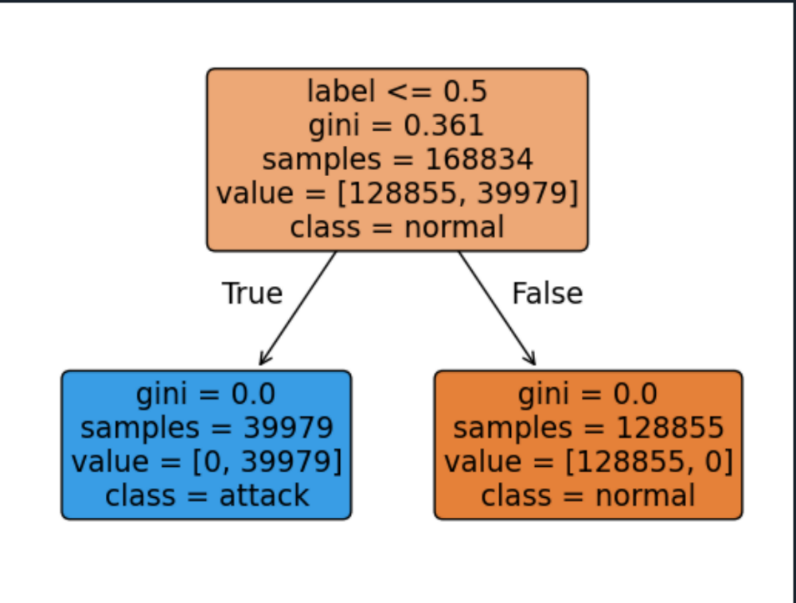


Figure 9: image of decision tree

Accuracy = 1.0, precision= 1.0 and recall =1.0. these results are obviously not real due to it being perfect. My assumption is I removed too many features needed.

##### 5.1.2.2 Second experiment

Upon trying to slowly re-enter the columns previously selected I started receiving the same error and this was an obvious problem that I had to solve. My solution to this was using a label encoder as shown in the code below.

newdf[['src\_ip','dst\_ip','proto','service','conn\_state']] = newdf[['src\_ip','dst\_ip','proto','service','conn\_state']].apply(LabelEncoder().fit\_transform)

Using this I was able to overcome the error I had and when I did the second run I received the following results: Accuracy = 1.0, precision= 1.0 and recall =1.0. this was weird as it was like the first experiment in which I removed more features

##### 5.1.2.3 Third experiment

At this point I had to ask myself whether there was a possible data leak and if the model was somehow memorising the data set instead of learning from it.

First thing I did was no longer combine all the nine attacks into one category and instead I left I the way it was. Upon doing this I was able to get an accuracy score of 42%, a precision score of 91% and a recall of 42% and a f1 score of 35 %. As seemingly low as some of the scores were, this seemed more a right step in the future and suggested that I needed to tune some parameters. I noticed within this experiment it seems to only give values for some of the attacks, which can imply it wasn’t reading or training on some of the attack.

The solution to this was increasing my maximum depth of the decision tree classifier. Doing this allowed more complex training of the model. Shown in the table below is the different results received when increasing the maximum depth.

Table 2. results after increasing maximum depth

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Max depth | Accuracy (nearest decimal point) | Precision  (nearest decimal point) | Recall  (nearest decimal point) | F1 score  (nearest decimal point) |
| 2 (original) | 42% | 91% | 42% | 35% |
| 4 | 66% | 86% | 66% | 59% |
| 5 | 78% | 87% | 78% | 78% |
| 6 | 85% | 91% | 85% | 85% |
| 7 | 94% | 94% | 94% | 94% |
| 8 | 95% | 95% | 95% | 95% |
| 9 | 96% | 96% | 96% | 96% |
| 10 | 97% | 97% | 97% | 97% |
| 12 | 98% | 98% | 98% | 98% |
| 15 | 99% | 99% | 99% | 99% |
| 20 | 99% | 99% | 99% | 99 |

As shown in the table above I carried out multiple mini experiment in which I increases the maximum depth by one or two and as shown in the results there was a difference. Having a maximum depth of 15 and 20 still yielded the same results but for the sake of computational cost it is required to have the lowest maximum depth possible. One important thing to mention is the possibility of overfitting when increasing the maximum depth. The result that sticks out the most is when the maximum depth is at a 6 at this was the last result to be high but not closely high to the next as shown with 7 and 8 and so one. Hence recommended best pick is a maximum depth of 12-15.

Finally with the code below I was able to determine whether I was underfitting or overfitting but comparing the training accuracy and the testing accuracy

y\_train\_pred = tree\_clf.predict(X\_train)

train\_accuracy = accuracy\_score(y\_train, y\_train\_pred)

print("Accuracy on the training set:", train\_accuracy)

With this the training and testing accuracy was respectively 0.9824679863060758 (98%) and 0.9809282380534957 (98%)

##### 5.1.3 Further hyper parameter tuning

Using more parameters allow for better results and more reliable and before accepting the results for what it was, I wanted to make sure the model had everything it needed. To do this I experiment by adding hyper parameter called criterion, min\_sample\_leaf and min\_samples\_split.

By now adding more parameters to train the model and doing trail by error would be time costly. Using the code below I was able to get the best parameters.

param\_grid = {'criterion':["gini","entropy"],'min\_samples\_leaf': [2, 10,50],'min\_samples\_split': [100, 2000, 30000], 'max\_depth':[12,13,14,15]}

tree\_clf = DecisionTreeClassifier()

grid\_search = GridSearchCV(tree\_clf, param\_grid, cv=5)

grid\_search.fit(X\_train, y\_train)

print("Best hyperparameters: ", grid\_search.best\_params\_)

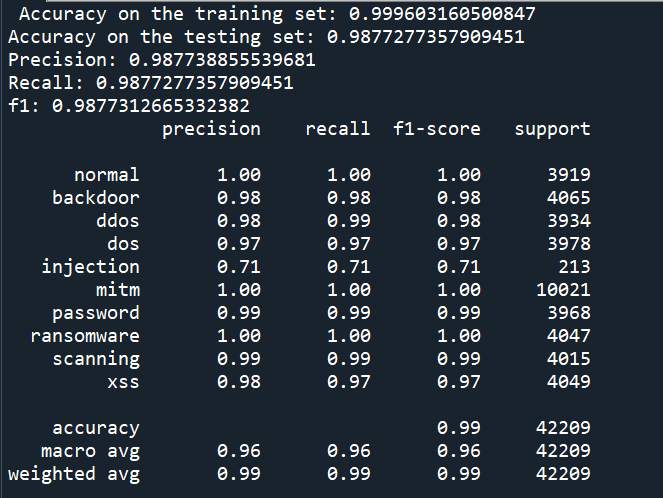
With this I was able to receive the following results shown below

Figure 10 : results for decision tree

I concluded that these results were valid due to the training and testing being very close and that I had been able to program my first model to successful detect the attack at a high percentage. (maybe crop put the attack percentages to reduce confusion)

### 5.2 Random Forest:

Random forest is very similar to decision tree as it combines multiple decision trees. The code to set up the dataset and the target variables remained the same. However, the small changes that occurred in the development of this model was:

* Importing the random forest classifier shown below

from sklearn.ensemble import RandomForestClassifier

* Creating the classifier

rf = RandomForestClassifier(n\_estimators = 100)

* Training the model

rf.fit(X\_train, y\_train)

Upon running the result, I as able to get an accuracy of 0.995 which is essentially a 100 percent accuracy. Once again, I had to consider the possibility of overfitting this required doing some hyper parameter tuning.

The hyper parameters that I chose to tune was 1) n\_estimators which is the number of decision trees in the forest, increasing this typically improves the performance of the model but a negative is the additional computational cost and 2) max\_depth which is the maximum depth of each decision tree, setting too high or low can lead to overfitting or underfitting.

Following the steps below I was able to randomly search through a range of set parameters of n\_estimators and max\_depth and define which was the best.

Step 1. Import the function random search cv

from sklearn.model\_selection import RandomizedSearchCV

Step 2. Set the parameters

param\_dist = {'n\_estimators': randint(50,500),'max\_depth': randint(1,20)}

Step 3. Train many models with iterations

rand\_search = RandomizedSearchCV(rf,param\_distributions = param\_dist,n\_iter=5, cv=5)

Step 4. Save the best model and display

best\_rf = rand\_search.best\_estimator\_

print('Best hyperparameters:', rand\_search.best\_params\_)

Following these steps, I was able to receive the output on the best parameters as shown in the snippet below:



Figure 11:test1

I noticed after running the test again that the parameters seemed to change, and this is due to the type of model we are using ‘random’ forest. Hence, I planned to get an average based of multiple test run: results are shown below

Test2:



Figure 12:test2

Test3:

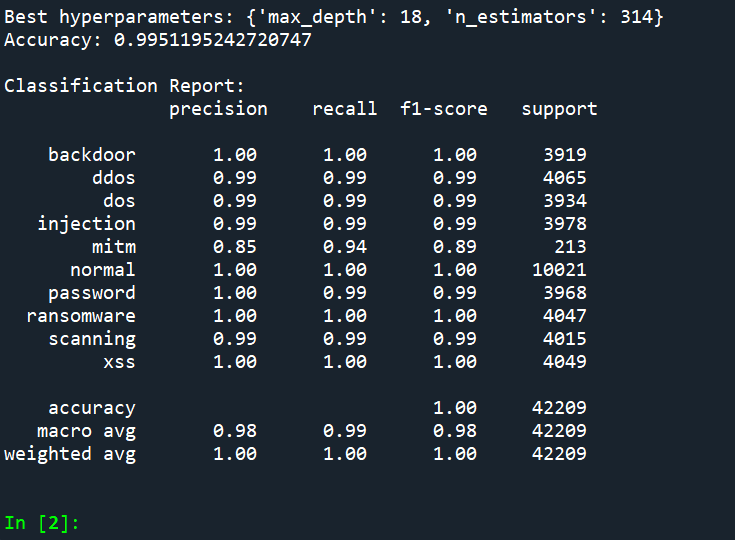


Figure 13:test3

Test4:



Figure 14:test4

Test5:



Figure 15:test5

Using the test above I calculated that the bet parameters which was the average of them all was: a max\_depth of 16 and n\_estimators of 381

##### 5.2.1 Further hyper parameter tuning

I wanted to do the same for decision tree in which I add additional parameters that I felt necessary to improve the models speed. I decided to add the following:

* Min\_sample\_leaf – checks that if a split results in a child with fewer samples the split will be avoided
* Max\_features – determines the number of features to consider when splitting a node, influencing the complexity of the forest tree
* Min\_samples\_split – minimum number of samples required to split an internal node

Using the same code to search for the best estimator I got the following as I had expected. Once again due to it being random forest the variables are likely to change but the accuracy stays consistence.

Figure 16: best parameters

param\_dist = {'min\_samples\_leaf': [1,50],'max\_features': ['sqrt','log2',None],'min\_samples\_split': [2,100,500]}

One final thing I did before accepting the results for what it was once again display the accuracy on both training and testing to see If there was any underfitting or overfitting, but I also added a final thing called oob\_score that is a performance metric calculated on data points not used to train a specific decision tree. By declaring oob is equal to true when creating the random forest classifier, I was able to conclude that the following were the results from using random forest machine.

Out-of-bag score: 0.995480768091735 = 100%

Accuracy on training = 0.9999526161792056 = 100%

Accuracy on testing = 0.9900021322466772 = 99%

### 5.3 Neural network

Neural network unlike random forest and decision tree is more complex. Neural network is commonly used for classifying images, hence there’s not much work shown being done on network datasets

##### 5.3.1 Importing of libraries

Importing the following libraries was needed

import os

os.environ['TF\_ENABLE\_ONEDNN\_OPTS'] = "0"

os.environ['TF\_CPP\_MIN\_LOG\_LEVEL'] = '1

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score,precision\_score, recall\_score,f1\_score ,confusion\_matrix,ConfusionMatrixDisplay

from sklearn.metrics import classification\_report

from sklearn.preprocessing import LabelEncoder

The changes in this compared to the other models were importing TensorFlow used to train neural networks and importing os, this was to reduce the notice messages it displayed to remove confusions.

##### 5.3.2 Development of model 1

The code below shows my initial development of the training a neural network

tf.random.set\_seed(42)

model\_1 = tf.keras.Sequential([tf.keras.layers.Dense(1)])

model\_1.compile(

loss=tf.keras.losses.BinaryCrossentropy(),

optimizer=tf.keras.optimizers.SGD(),

metrics=['accuracy']

)

# Train the model using the training data

model\_1.fit(X\_train, y\_train, epochs=5)

The’ tf.random.set\_seed(42)’ ensures that the results of operations involve randomness ( such as weights and shuffling of data) , upon running the same code with the same sees the results will stay the same. The next line of code creates a single dense layer with one neuron.

Finally, ‘the model\_1. compile’ consist of the main body of this model training. This is where the binary classification problems are solved, and the model is trained, and accuracy is displayed.

# Train the model using the training data

model\_1.fit(X\_train, y\_train, epochs=5)

Finally, the code above shows the code in how I trained the model, I used epochs which signify that the model has gone through the training set completely in each loop.

Upon trying the first run I received the error:

Figure 17: error on neural network

This occurs when the model is trying to read data that is not numerical but instead categorical. the only categorical values I had in the dataset was the type which had the features of the attack names (converted 9 attacks into just one attack).

My solution to this was convert the categorical column ‘type’ to numerical and furthermore i made the column ‘ Label’ the target variable instead. After doing so I was able to get the following results show in Figure 15.

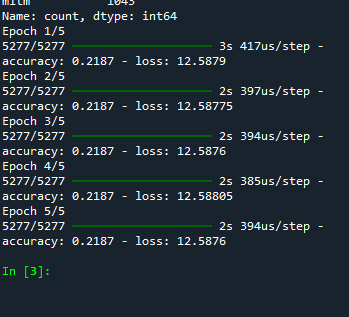


Figure 18: first run of neural network

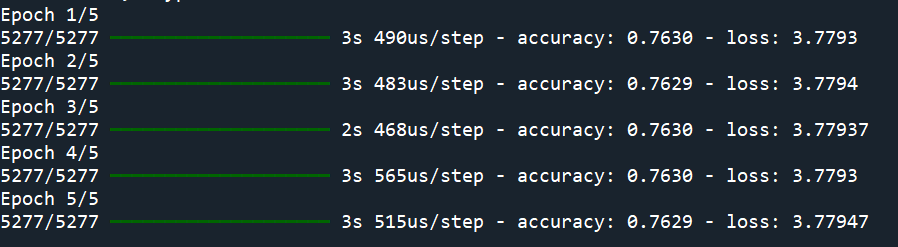
An accuracy around 21% with a loss of 13 (the loss tracks the margin of error made). Before I made any changes to the code such as parameter tuning wanted to run the code again to ensure it produced the same accuracy and to my surprise it provided a completely different accuracy score shown in figure 16.

Figure 19: second run on neural network

An accuracy of 76% with only a margin error of 4, this is closer to what I attempt to make my model output, however there was the error such as why the accuracy was changing drastically. My solution was to further update the code and make it stronger.

I remembered that the reason this was happening was because the model was shuffling the data and having random weighs so the accuracy would differ occasionally. My solution to this was adding an initialiser gotten from the TensorFlow website as shown below:

initializer = tf.keras.initializers.GlorotUniform(seed=42)

model\_1 = tf.keras.Sequential([tf.keras.layers.Dense(1, kernel\_initializer=initializer)])

the idea of this is that the data should no longer be shuffled, and I would receive the same accuracy. My justification for this is that as good as the fluctuation of accuracy was, for the validity of this research I must start my way from the most accurate accuracy and work my way up. After running this code multiple times across the multiple runs the accuracy remained the same at 23%.

##### 5.3.3 Further development and hyper parameter tuning

###### Model 2

Now I wanted to increase the accuracy, so I did further development, firstly I increase the epochs but there was no change in the accuracy,

Next thing I tried was making another model that had more layers and epochs training time.

model\_2 = tf.keras.Sequential([ tf.keras.layers.Dense(1),tf.keras.layers.Dense(1)])

model\_2.compile(loss = tf.keras.losses.BinaryCrossentropy(),

optimizer = tf.keras.optimizers.SGD(),

metrics = ['accuracy']

)

model\_2.fit(X\_train, y\_train, epochs = 3 )

Doing this I was able to see a slight increase in the accuracy from 24% to 76% as shown in figure 17. I noticed increasing the epochs don’t seem to have much effect so from now on I kept it at 3 to minimise computational cost and make debugging quicker.

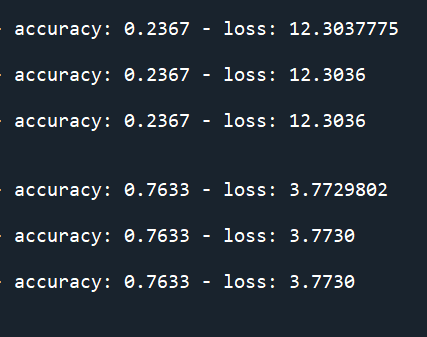
from this came the idea that had to increase the layers and neurons in my model to get a higher accuracy.

Figure 20: image of accuracy

###### Model 3

Like the second model I added more layers, furthermore with in these layers I added a different amount of neurons

model\_3 = tf.keras.Sequential([

tf.keras.layers.Dense(100,kernel\_initializer=initializer), # add 100 dense neurons

tf.keras.layers.Dense(10,kernel\_initializer=initializer), # add another layer with 10 neurons

tf.keras.layers.Dense(1,kernel\_initializer=initializer)])

model\_3.compile(loss = tf.keras.losses.BinaryCrossentropy(),

optimizer = tf.keras.optimizers.SGD(),

metrics = ['accuracy'])

model\_3.fit(X\_train, y\_train, epochs = 3 )

In the first layer I had 100 neurons and within the second I had only 10 neurons. Running this, I received the same accuracy as I did with model 2 that had less layers and neurons, this of course was another minor problem that I had to overcame in model 4.

###### Model 4

After seeing that adding layers and neurons wasn’t making a change at all I decide that I had to make changes in other Areas. The two changes I initially made were

1. Utilizing an activation function:

Shown in the code below I added two activation function into the hidden layers, and I reduce the neurons down to 4

model\_4 = tf.keras.Sequential([

tf.keras.layers.Dense(4,activation = 'relu', kernel\_initializer=initializer), # add 4 dense neurons

tf.keras.layers.Dense(4,activation = 'relu', kernel\_initializer=initializer), # add another layer with 4 neurons

tf.keras.layers.Dense(1,activation = 'sigmoid', kernel\_initializer=initializer)])

1. I changed the optimiser from SGD to Adam:

optimizer = tf.keras.optimizers.Adam(learning\_rate = 0.01)

1. Simple code to display the model loss with accuracy

loss, accuracy = model\_4.evaluate(X\_test, y\_test)

print(f' Model loss on the test set: {loss}')

print(f' Model accuracy on the test set: {100\*accuracy}')

Even after doing this the accuracy remained the same, however what worsened was the model loss on the set . it increased from 3 to 54 as shown in figure 18.



Figure 21: same accuracy, worse mode loss

Further parameter tuning

There were many things to change around but I shortly will brief it. First, I decreased the learning rate from 0.01 to 0.001 as it allows the model to learn slower



Figure 22: decreasing learning rate

As shown the accuracy decreased but the model loss on the set also lowered. Next thing I did was increase the epochs which is how many times it trains. I increased the epochs by 1 to an epoch of 4 and received the following result

* Model loss on the train set was 1.138930 and model accuracy on the train set was 96.767830
* Model loss on the test set was 1.266221 and model accuracy on the test set was 96.7944521

This suggest there was slight overfitting.

To combat this, I increased the learn rate a little more from 0.001 to 0.005. doing this I was able to counteract the over fitting as shown below

* Model accuracy on the train set was 77.95349359512329
* Model accuracy on the test set was 77.74171233177185

Furthermore, I backtracked to now increasing the epoch to see if I could increase the accuracy, however no matter how much I increased the epochs whether it was to 5, 10 or 25 the accuracy stayed the same. So, one thing I tried was changing the neurons in the layer this also made no changes until I changed the neurons which were originally respectively 100 and 10 to 3228 and 1264 to only get a slight increase of three percent. This was time costly and not the way I wanted to go.

###### Model 5: new structure

This model is a different approach with a different structure, using multi layered perception which is still a neural network I changed only the code a little and the dataset was able to remain the same.

Shown in the code snippet below is the different approach

mlp\_clf = MLPClassifier(hidden\_layer\_sizes=(100), max\_iter=1000,learning\_rate\_init=0.01,random\_state=42)

mlp\_clf.fit(X\_train,y\_train)

By doing this I was able to get an accuracy of 82% which was the highest that the model was able to produce. Upon tuning the parameters by adding an activation function and a solver the accuracy remained the same so a different thing I did was increase the hidden layer sizes and from this the accuracy was able to change drastically. After multiple attempts of changing the hidden layer size I concluded that using a hidden layer size of 150 was good receiving an accuracy of 97%. However, when displaying accuracy on training set it was lower than the testing set which suggest overfitting as shown below



Figure 23: image showing overfitting

My solution to this was having an early stopping function which prevents over fitting and would stop the training when the performance could start to degrade. Doing this the accuracy reduced by a notable amount to 86% but when displayed the accuracy on both training and testing, the training was higher which I could conclude that there was no more over fitting

mlp\_clf = MLPClassifier(hidden\_layer\_sizes=(149), max\_iter=1000,learning\_rate\_init=0.01,random\_state=42, early\_stopping=True,validation\_fraction=0.15,n\_iter\_no\_change=10)

mlp\_clf.fit(X\_train,y\_train)



Figure 24: image showing no more overfitting

# Testing and Evaluation of Artefact

The proposed plan of this study was to train three algorithms’ models and upon doing so evaluate which model was the best. To do this I plan to carry out cross evaluation and by this I will mention the noticeable features that came from training and developing each model on the dataset

### 6.1 Metric evaluation:

First is the accuracy of all three models shown in table 3:

Table 3:

|  |  |  |
| --- | --- | --- |
| Models: | Accuracy on training | Accuracy on testing |
| Decision tree | 0.99960 (100%) | 0.98772 (99%) |
| Random forest | 0.99995 (100%) | 0.99000 (100%) |
| Neural Network | 0.86435 (86%) | 0.86213 (86%) |

As displayed the accuracy received from both training and testing are very similar which shows no under or over fitting. Furthermore, it demonstrates their ability to have an accurate response after being trained to still get the same accuracy on the testing phase.

However, even so when it came to the neural network no matter how many hyper parameters tuning and building was done the accuracy would remain at around 78% until introducing a different structure shown in model 5 in which the accuracy was then 86% unlike decision tree with a 99% percent accuracy when testing and random forest with an 100%.

Next is precision and recall

|  |  |  |
| --- | --- | --- |
| Models: | Precision | Recall |
| Decision tree | 0.98773 | 0.98772 |
| Random forest | 0.99567 | 0.99564 |
| Neural Network | 0.87464 | 0.86213 |

Table 4:

Like the accuracy the precision and recall for decision tree and random forest are also high.

Final is f1 Score

Table 5:

|  |  |
| --- | --- |
| Models: | F1 score |
| Decision tree | 0.98773 |
| Random forest | 0.99565 |
| Neural Network | 0.84232 |

Calculated from precision score and recall score, the f1 score further shows a model accuracy. As we can see random forest once again take the lead with an f1 score of 1 which is the highest with decisions closely behind with a score of 0.9.

### 6.2 non-metrics evaluation

Within this mini section contains little info and notes that I felt necessary to mention.

Processing data

I used the same processing dataset for decision tree and random forest to ensure there was no unfairness or bias with the algorithm model. This helps clear out the possibility on data leakage. However, when I came to neural network, I had to change a bit around as mentioned. I converted the 9 attack types into one column called attack and instead of having type be a target variable I switched it with label. My justification for this was because the model was looking for numerical values and not able to read categorical hence why I switched it, but when it came to model 5, I was able to switch back to the way it originally was.

Speed

Speed is an important feature in training machine learning and when it came to the fastest model to train, decision tree stood at top. Decision tree was one of the models that we were able to tune parameters and still have a quick processing speed unlike neural network in which increasing certain parameters did not increase accuracy but still increased the time the model took to train. (However, when it came to using MLP classifier the training speed was very low and quick)

Complexity

The model with the most complex development was of course neural network due to its structure of input and output layer but furthermore having hidden layers that could be tweaked to include different activation functions and neurons. Decision tree and random forest doesn’t have a high complexity when training the model which can be good or bad.

### 6.3 Chosen model:

Based on the metric and non-metric evaluation of the three models I concluded that the best model out of the three was random forest with the highest metric evaluation across the board. I had the thought of justifying my pick by showing how the model completes my functional and non-functional requirements as shown below

Functional

For my functional I had two requirements: the model would have an accuracy of 90% or above and that it would be able to show a reduction in false positives and negatives.

As shown in the results before using random forest I was able to not only get an accuracy above 90% which was the minimum, but I was able to get an accuracy of 100% which was more than my preferable, hence the model was able to complete this requirement

As for reduction of false positives and negatives using the confusion matrix shown below in figure 20 which helps plot the model prediction against what its actual prediction was and using this, we can understand the trade-off between false positives and false negative.

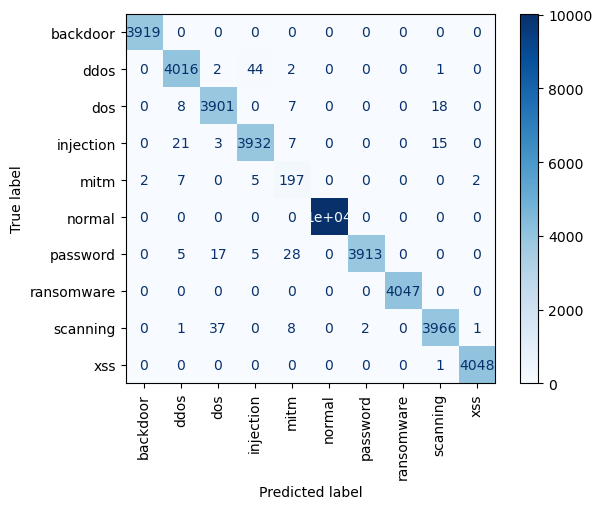


Figure 25: cross matrix

As shown for every predicted attack label when paired with its true label we can see how highly accurate it gets it right, of course occasionally it would make mistakes and classify one attack as another but as shown there are all very low instances which can suggest that the number of false negatives and positives is certainly low.

Non-functional requirements

k-fold cross evaluation

K-fold cross validating is a technique used to evaluate the performance of machine learning models and in the case of random forest this was an easy thing to implement. K fold stands for the number of splits the training data is split into Shown in the code below,

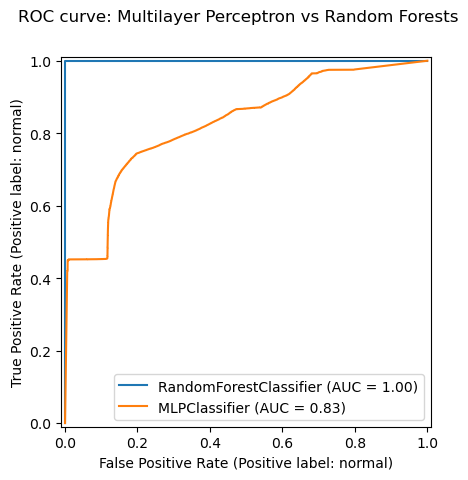
cv = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

# Perform k-fold cross-validation using cross\_val\_score

cv\_scores = cross\_val\_score(rf, X\_train, y\_train, cv=cv, scoring='accuracy')

Doing this I was able to receive the following validation scores to further evaluate my model

Figure 26: cross-validation

ROC-AUC Curve

The Roc curve is used to show the performance of binary classification. It plots the true positive rate against the false positive rate and the AUC is the area under the curve which sums up how well a model can produce scores to discriminate between positive or negative.

In figure 22 I utilised this evaluation technique by plotting the highest performing model against the lowest as shown. The random forest being the highest with an AUC score of 1.00 meaning a perfect model

Whereas MLP classifier has an AUC of 0.83, we can see from the graph that the MLP classifier has a small false positive rate.

Figure 27:ROC curve

# Project Evaluation

The idea of this project was to utilise the capabilities of Machine learning and train models on a network security dataset with the aim of increasing the accuracy of intrusion detection systems. To do this we would have the model shown that when trained it would reach a high accuracy and furthermore have high scores when it came to other evaluation techniques. When it came to the completion of the project there was things I was able to do well and things that I couldn’t achieve such as:

I was able to successfully train my models with two of them having an accuracy of 95% or above. After coding the structures of my models, I was able to train the models on the dataset to classify the difference between a normal behaviour and an attack behaviour. Upon doing that both my random forest and decision tree was able to get an accuracy of 99% with other evaluation metric scores getting around 95% or above.

Furthermore, using the confusion matrix and the ROC-AUC curve, I was able to demonstrate and display the low number of false positives and negatives which was minimised. From these I was able to complete two of my objectives. Other objectives I completed were showing the benefits of integrating Machine learning with cyber security to help develop more accurate Security tools. Training the machine learning to adapt to emerging threats. Due to the type of attacks, I trained my model on, I can successfully say that it is trained against emerging threats and can further be developed.

However, there was one Objective that I couldn’t complete and that was further integration of my model by adding it into an IDS. My attempt even after training the model was to deploy into an IDS software whether created or freely available, however I couldn’t do this due to the limited time I had.

I severely underestimated the time it would take me to complete this project as shown in the Gantt chart however even then I was able to follow through with my original plan of training my three models and cross evaluating them to help demonstrate the overall use of Ai

However, there are several things I would have done better such as:

* Train more algorithm models to further demonstrate the capabilities of Ai. My idea is I can train more models that I originally did with more time and more parameters
* Had another dataset on standby with similar features so that when I trained the models against the new one and got the accuracy I could see if it was able to do just as well when giving different datasets with similar features and data, furthermore if there was new attack types that it didn’t originally train against it could be a stepping stone to help train the model detect and learn new patterns
* Had better time management. If I did, I could have developed this project furthermore in many ways
* Develop the model into an open-source IDS

# Conclusion and Future Work

## Conclusion

Coming into this project I had the goal of showing that Ai can and will be used to help develop key infrastructure. I had many objectives which concluded finding a good dataset, training the algorithm models, further training of them to detect new attacks, evaluating the final model with k fold cross evaluation and more. These all seemed like achievable goals and for most part they were. for example, the dataset I used was one of the newest and most popular and furthermore the three models I had chosen to train were beneficial in their own way to the project. However, there were some goals that weren’t achieved such as integrating the model into IDS this being due to not having enough time.

However, this study had many achievements such as the obvious reaching an accuracy of 100 percent on one of the models trained but also being able to show a reduction of false positive and false negatives. Furthermore, I was able to make personal achievements such as a deeper understanding in machine learning and data preprocessing and stronger researching skills.

Of course, with everything I had my own difficulties when completing this study. The first being time management, I had taken most of my time researching and writing the study and not that much time on my models. The second is training the model neural network, this was the most complex as at the time I had not much knowledge on the general structure of neural networks since it being less common than random forest and decision tree. Furthermore, when it came to neural network the preprocessing data was a little bit different as it had to be adjusted so neural network can train with no problems.

Throughout the completion of this study, I had learnt many lessons such as:

* The importance of time management and not underestimating how long things will take
* Learning how to code to train the different models
* The use of training multiple models to cross evaluate and using further evaluation methods.
* The importance of tuning parameters when coding the model and the significance it can have especially on training time.

The overall purpose of this study was to see if I could increase the accuracy of an ids using AI and furthermore show a decrease in false negatives and positives. Despite any shortcomings I had in this project such as not being able to integrate my model with an open-source IDS or making my own I was still able to do exactly what the aim of this project was with my chosen model random forest. By doing this I was able to show the benefits with using Ai and its capabilities, hence I can conclude that we can use Ai to further improve the accuracy of detection system such as IDS.

## Future Work

Based on the results of this project I was able to successfully conclude Ai can help but even so here are many things I would like to do to further develop the future of cyber security with Ai

Designing a polymorphic virus

A polymorphic virus is a virus that can change its code when it produces offspring that are replication of the mother, the idea of this is that it can be used to bypass security tools that are designed to search for a limited thing. In the case of my model if it was to come up against such a thing there is a high chance it won’t be able to detect the virus at a consistent rate, hence why I see it beneficial to create a polymorphic virus to train against the model. This would consist of other side projects like collecting data left behind from the polymorphic virus to train to the IDS. This would be a severely complex thing to do, However I feel it would be beneficial for the development of cyber security

Further development in algorithms

The neural network was the model with the lowest accuracy and other metric sores by a little amount, so in future works I would like to research a lot more about neural networks and furthermore develop a more complex neural network to Train. If this was to not work out as expected yet again then I would implement other algorithms

Tuning parameters

Across training the three models I had done an 80/20 split in which 80% of the dataset was used for training and the remainder for testing, the remained the same for all the models to remove unfairness but in future experiments I feel it be beneficial to see the outcomes of changing parameters that wasn’t rarely touched such as doing a 70/30 split instead, increasing or decreasing the columns kept in the data preprocessing stage.

I believe that by accomplishing all of the above mentioned and much more that the domain of cyber security will be severely impacted in a good way and that with more development and time spend we will be able to come to a standpoint in where a Ai based IDS will be able to predict and detect all and every new attacks alongside the common ones , while this may be my personal view its important to see the capabilities of Ai when paired with cyber security.

# Reference list

Alom, M.Z. and Taha, T.M. (2017). Network Intrusion Detection for Cyber Security Using Unsupervised Deep Learning Approaches. *2017 IEEE National Aerospace and Electronics Conference (NAECON)*. [online] doi:https://doi.org/10.1109/naecon.2017.8268746.

Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A. and Anwar, A. (2020). TON\_IoT Telemetry Dataset: a New Generation Dataset of IoT and IIoT for Data-driven Intrusion Detection Systems. *IEEE Access*, [online] pp.1–1. doi:https://doi.org/10.1109/access.2020.3022862.

Ashraf, J., Keshk, M., Moustafa, N., Abdel-Basset, M., Khurshid, H., Bakhshi, A.D. and Mostafa, R.R. (2021). IoTBoT-IDS: a Novel Statistical learning-enabled Botnet Detection Framework for Protecting Networks of Smart Cities. *Sustainable Cities and Society*, [online] 72, p.103041. doi:https://doi.org/10.1016/j.scs.2021.103041.

Azizan, A.H., Mostafa, S.A., Mustapha, A., Foozy, C.F.M., Wahab, M.H.A., Mohammed, M.A. and Khalaf, B.A. (2021). A Machine Learning Approach for Improving the Performance of Network Intrusion Detection Systems. *Annals of Emerging Technologies in Computing*, [online] 5(5), pp.201–208. doi:https://doi.org/10.33166/aetic.2021.05.025.

Booij, T.M., Chiscop, I., Meeuwissen, E., Moustafa, N. and Hartog, F.T.H. den (2022). ToN\_IoT: the Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets. *IEEE Internet of Things Journal*, [online] 9(1), pp.485–496. doi:https://doi.org/10.1109/jiot.2021.3085194.

Canadian Institute for Cybersecurity (2017). *IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. [online] Www.unb.ca. Available at: https://www.unb.ca/cic/datasets/ids-2017.html.

Candan, B. (2023). *Top 5 critical infrastructure cyberattacks*. [online] www.anapaya.net. Available at: https://www.anapaya.net/blog/top-5-critical-infrastructure-cyberattacks.

Copeland, B.J. (2024). History of artificial intelligence (AI) - Connectionism | Britannica. In: *Encyclopædia Britannica*. [online] Available at: https://www.britannica.com/science/history-of-artificial-intelligence/Connectionism.

corelight (n.d.). *Reducing IDS False Positives | Corelight*. [online] corelight.com. Available at: https://corelight.com/resources/glossary/ids-false-positives.

Davies, V. (2021). *The history of cybersecurity*. [online] Cyber Magazine. Available at: https://cybermagazine.com/cyber-security/history-cybersecurity.

Foote, K.D. (2021). *A Brief History of Machine Learning - DATAVERSITY*. [online] DATAVERSITY. Available at: https://www.dataversity.net/a-brief-history-of-machine-learning/.

Granter, S.R., Beck, A.H. and Papke, D.J. (2017). AlphaGo, Deep Learning, and the Future of the Human Microscopist. *Archives of Pathology & Laboratory Medicine*, [online] 141(5), pp.619–621. doi:https://doi.org/10.5858/arpa.2016-0471-ed.

hari, sindhuja (n.d.). *Best Machine Learning Frameworks(ML) for Experts in 2022*. [online] Hackr.io. Available at: https://hackr.io/blog/machine-learning-frameworks.

Holdsworth, J. (2024). *What is Data Mining? | IBM*. [online] www.ibm.com. Available at: https://www.ibm.com/topics/data-mining.

Kim, J., shin, N., Yeon jo, S. and Hyun Kim, S. (2017). *Method of Intrusion Detection Using Deep Neural Network*. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/7881684.

Koch, R. (2022). *History of Machine Learning - a Journey through the Timeline*. [online] clickworker.com. Available at: https://www.clickworker.com/customer-blog/history-of-machine-learning/.

Kolawole Ayodeji, D., Isaac, S., Mathew Karma, S., Luqman, Y. and Aminu, J. (2025). *View of CYBER SECURITY ATTACK DETECTION MODEL USING SEMI-SUPERVISED LEARNING*. [online] Fudutsinma.edu.ng. Available at: https://fjs.fudutsinma.edu.ng/index.php/fjs/article/view/2343/1793 [Accessed 3 Feb. 2025].

Kumar, G., Kumar, K. and Sachdeva, M. (2010). The Use of Artificial Intelligence Based Techniques for Intrusion detection: a Review. *Artificial Intelligence Review*, 34(4), pp.369–387. doi:https://doi.org/10.1007/s10462-010-9179-5.

Lenaerts-Bergmans, B. (2023). *What Is an Advanced Persistent Threat (APT)? | CrowdStrike*. [online] Crowdstrike.com. Available at: https://www.crowdstrike.com/en-us/cybersecurity-101/threat-intelligence/advanced-persistent-threat-apt/.

Lutkevich, B. (2021). *What Is an Intrusion Detection System (IDS)? Definition from SearchSecurity*. [online] TechTarget. Available at: https://www.techtarget.com/searchsecurity/definition/intrusion-detection-system.

Moustafa, D.N. (2021a). *The TON\_IoT Datasets | UNSW Research*. [online] research.unsw.edu.au. Available at: https://research.unsw.edu.au/projects/toniot-datasets.

Moustafa, N. (2019a). *A Systemic IoT-Fog-Cloud Architecture for Big-Data Analytics and Cyber Security Systems: a Review of Fog Computing*. [online] arXiv.org. Available at: https://arxiv.org/abs/1906.01055 [Accessed 17 Feb. 2025].

Moustafa, N. (2019b). *New Generations of Internet of Things Datasets for Cybersecurity Applications Based Machine Learning: TON\_IoT Datasets EXECUTIVE SUMMARY*. [online] Available at: https://conference.eresearch.edu.au/wp-content/uploads/2019/08/2019\_eResearch\_59\_New-Generations-of-Internet-of-Things-Datasets-for-Cybersecurity.pdf.

Moustafa, N. (2021b). A New Distributed Architecture for Evaluating AI-based Security Systems at the Edge: Network TON\_IoT DatasetsNourMoustafaa1⁎TON\_IoT datasets: a Distributed Architecture for Evaluating Artificial Intelligence-based Security Systems in IoT Networks. *Sustainable Cities and Society*, [online] 72, p.102994. doi:https://doi.org/10.1016/j.scs.2021.102994.

Moustafa, N., Ahmed, M. and Sherif Sayed Ahmed (2020). Data Analytics-Enabled Intrusion Detection: Evaluations of ToN\_IoT Linux Datasets. [online] doi:https://doi.org/10.1109/trustcom50675.2020.00100.

Moustafa, N., Marwa Keshky, Essam Debiez and Janicke, H. (2020). Federated TON\_IoT Windows Datasets for Evaluating AI-Based Security Applications. [online] doi:https://doi.org/10.1109/trustcom50675.2020.00114.

National Cyber Security Centre (2020). *What Is Cyber security?* [online] ncsc.gov.uk. Available at: https://www.ncsc.gov.uk/section/about-ncsc/what-is-cyber-security.

Noorbehbahani, F. and Saberi, M. (2020). *Ransomware Detection with Semi-Supervised Learning*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICCKE50421.2020.9303689.

Okta (2023). *Defining Intrusion Detection Systems & How IDS Monitors Work | Okta*. [online] www.okta.com. Available at: https://www.okta.com/identity-101/intrusion-detection-system/.

Pirc, J. (2017). *The Evolution of Intrusion Detection Prevention Then Now and the Future*. [online] Secureworks.com. Available at: https://www.secureworks.com/blog/the-evolution-of-intrusion-detection-prevention.

Sarker, I.H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, [online] 2(3), pp.1–21. doi:https://doi.org/10.1007/s42979-021-00592-x.

Sas.com. (2024). *A guide to the types of machine learning algorithms*. [online] Available at: https://www.sas.com/en\_gb/insights/articles/analytics/machine-learning-algorithms.html# [Accessed 25 Nov. 2024].

Sayegh, E. (2025). *The Cybersecurity Crisis: Companies Can’t Fill Roles, Workers Shut Out*. [online] Forbes. Available at: https://www.forbes.com/sites/emilsayegh/2025/02/05/the-cybersecurity-crisis-companies-cant-fill-roles-workers-shut-out/.

Sergio (2023). *Four Cyber Security Milestones That Shaped the Future of Malware*. [online] Telefónica Tech. Available at: https://telefonicatech.com/en/blog/four-cyber-security-milestones-that-shaped-the-future-of-malware [Accessed 27 Mar. 2025].

Shun, J. and Malki, H.A. (2008). Network Intrusion Detection System Using Neural Networks. *2008 Fourth International Conference on Natural Computation*. [online] doi:https://doi.org/10.1109/icnc.2008.900.

Srivastava, T. (2019). *Evaluation Metrics Machine Learning*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/.

Stallings, W., Brown, L., Bauer, M., Howard, M., Columbus, B., New, I., San, Y., Upper, F., River, S., Cape, A., Dubai, T., Madrid, L., Munich, M., Montreal, P., Delhi, T., Sao, M., Sydney, P., Kong, H., Singapore, S. and Tokyo, T. (2008). *COMPUTER SECURITY PRINCIPLES AND PRACTICE Second Edition*. [online] Available at: https://unidel.edu.ng/focelibrary/books/Computer%20Security%20\_%20Principles%20-%20WILLIAM%20STALLINGS\_2089.pdf.

Stuxnet, R., Baezner, M. and Robin, P. (2017). Research Collection. *Stuxnet*. [online] doi:https://doi.org/10.3929/ethz-b-000200661.

Tableau (n.d.). *What Is the History of Artificial Intelligence (AI)?* [online] Tableau. Available at: https://www.tableau.com/data-insights/ai/history#definition.

TimesPro (2024). *All about Supervised Machine Learning Examples [2024 Outlook]*. [online] TimesPro. Available at: https://timespro.com/blog/what-is-supervised-machine-learning-advantages-and-examples.

Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N. and Robinson, S. (2017). *Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams*. [online] Available at: https://cdn.aaai.org/ocs/ws/ws0325/15126-68350-2-PB.pdf.

Wright, G. (2023). *What is critical infrastructure? - Definition from WhatIs.com*. [online] WhatIs.com. Available at: https://www.techtarget.com/whatis/definition/critical-infrastructure.

Yana Storchak (2024). *Top 10 Best-Known Cybersecurity Incidents and What to Learn from Them | Syteca*. [online] Syteca. Available at: https://www.syteca.com/en/blog/top-10-best-known-cybersecurity-incidents-and-what-to-learn-from-them.

Yang, X.-S., He, X.-S. and Fan, Q.-W. (2020). Mathematical Framework for Algorithm Analysis. *Elsevier eBooks*, pp.89–108. doi:https://doi.org/10.1016/b978-0-12-819714-1.00017-8.

Zaib, H. (n.d.). *NSL-KDD*. [online] www.kaggle.com. Available at: https://www.kaggle.com/datasets/hassan06/nslkdd.

# Appendix

## Project Specification



## Signed Monthly Reports and other documentation





