Network Intrusion Detection using Supervised and Deep Learning Machine Learning Algorithms

MSc Academic Research Project
Cybersecurity

Ayodele Oluwagbayi Jolayemi
Student ID: x21139288

School of Computing
National College of Ireland

Supervisor: Rejwanul Haque

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ayodele Oluwagbayi Jolayemi……………………………………………………………………
Student ID:	x21139288……………………………………………………………………………………..…….
Programme:	Cybersecurity……………………………………… **Year:** 2024…………………….
Module:	Msc Academic Research Project………………………………………………………….……
Supervisor:	Rejwanul Haque…………………………………………….………………………….………
Submission Due Date:	31/Jan/2024………………………………………………………………………………..……
Project Title:	Network Intrusion Detection using Supervised and Deep Learning Machine Learning Algorithms……….……….……….…..……………………….………
Word Count:	10137………………………… **Page Count :** 33……………………………….…….

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Ayodele Oluwagbayi Jolayemi………………………………………………………

Date: 31/Jan/2024……………………………………………….…………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Table of Contents

Table of Figures

List of Tables

Network Intrusion Detection using Supervised and Deep Learning Machine Learning Algorithms

Ayodele Oluwagbayi Jolayemi
Student ID: x21139288

Abstract

The expansion of computer networks has exacerbated worries about network security and resulted in a number of infiltration attempts. The confidentiality, integrity, and availability of data and systems are compromised by these attempts. The necessity to handle the growing risks of cyberattacks is highlighted by statistics showing an increase in the frequency of malware assaults and denial of service situations. The increase in network traffic, the complexity of Network Intrusion Detection Systems (NIDS), and the variety of protocols and data transferred via contemporary networks are the three main issues that worsen network security. The existing traditional approaches are unable to detect new types of attacks, thereby necessitating the need for more robust solutions. The goal of this research is to increase the effectiveness of machine learning and deep learning models, which include some of the most applied classification approaches, namely decision trees (DT), logistic regression (LR), naïve bayes (NB), convolutional neural networks (CNN) and recurrent neural networks (RNN). Additionally, I examined the models' performance in binary classification as well as the effects of feature significance selection and hyperparameter adjustment on the CICIDS 2017 and UNSW NB15 benchmark datasets. Based on the findings of the experiments, the optimized decision tree is the best model for a network intrusion detection system with accuracy, F1-score, and AUC score of 99.27%, 99.26%, and 99.27% respectively on the CICIDS 2017 dataset. On the UNSW NB15 dataset, the scores were 99.28% across all metrics. It outperforms other machine learning and deep learning classification techniques and underlines the superiority to traditional IDS.

Key terms: intrusion detection, machine learning, attack traffic, anomaly, network

1 Introduction

Computer security is becoming vital due to the widespread use of information technology in many areas of daily life. With society depending more and more on computerised systems, critical dangers like dynamic assaults and zero-day vulnerabilities have grown to be significant problems. Despite significant research efforts in the security arena, these dangers persist and provide challenges for the security environment. Specifically, the growth of computer networks has exacerbated worries about internet security within the context of today's networking environment and cutting-edge computing powers (Debar, Dacier and Wespi, 1999). When Internet Protocols (IPs) were initially formed, security was not given priority, therefore network operators have had to cope with a variety of intrusion attempts from both criminal individuals and large-scale botnets (Ahmad *et al.*, 2021). These penetration attempts pose a major risk to the confidentiality, integrity, and availability of data and systems (Papalexakis, Beutel and Steenkiste, 2012). Some numbers from Symantec's Internet Security Threat Report illustrate the scope of the problem. In 2010, there were over 3 billion malware attacks, indicating how common malicious software is that targets many platforms and devices. Furthermore, denial of service (DoS) attacks have become far more common by 2013. The Symantec Internet Security Threat Report (2014) states that these concerning statistics highlight the critical need to address and reduce the increasing risks associated with cyberattacks. Additionally, the rapid growth in popularity of the Internet of Things (IoT), the increasing levels of connectivity, and the extensive usage of cloud-based services may all be major contributors to this rapid increase.

This research highlights three critical flaws that significantly exacerbate the present network security issue. The initial restriction is the result of the unexpectedly high volume of network data traffic, which is expected to keep growing in the same direction. To manage and handle these exponential amounts properly, techniques for ever-faster, more efficient, and more effective data analysis must be used (Ahmim, Derdour and Ferrag, 2018). The second critical issue focuses on the requirement for more complicated and intensive monitoring in order to enhance the overall efficacy and accuracy of Network Intrusion Detection Systems (NIDS). To do this, NIDS analysis has to go beyond high-level and generalised observations, requiring a more comprehensive and contextually aware approach. For example, it becomes imperative to be able to correlate changes in behaviour with specific elements inside a network, such as distinct users, different versions of the operating system, or certain protocols (Mehmood *et al.*, 2017). The issue of network security is also rooted in the wide variety of protocols and data that are sent via contemporary networks. This multiplicity presents a difficult challenge with varying degrees of complexity and difficulty when attempting to discern between normal and abnormal conduct. It is more difficult to build an accurate baseline due to the complexity of protocols and the diversity of data, which increases the danger of exploitation or zero-day attacks

1.1 Motivation

One of the main issues with network security is the creation of a robust and efficient Network Intrusion Detection System (NIDS). Even though NIDS technology has come a long way, most solutions still rely on less effective signature-based methods rather than anomaly detection methods (Shone *et al.*, 2018). This preference for signature-based techniques is caused by a number of issues, such as the high false error rate and associated costs, the challenge of obtaining reliable training data, and the dynamic nature of system activity (Zhao *et al.*, 2016). Conversely, relying solely on signature-based techniques will lead to inaccurate and inefficient intrusion detection. Given the ongoing advancements in contemporary networks, the current situation highlights the need to construct an anomaly detection system that is widely accepted and capable of overcoming these limitations (Dong and Wang, 2016). One of the main reasons why signature-based techniques fail is because they rely too much on pre-established patterns of recognised assaults. This approach may easily overlook novel and zero-day assaults that don't match the current signatures. Furthermore, signature databases must be updated often to be up to date with new threats, increasing maintenance costs and needs (Sazzadul Hoque, 2012). Contrarily, anomaly detection techniques focus on identifying deviations from normal network behaviour. Establishing a baseline of usual network activity might help identify any anomalous conduct as a possible intrusion. But there are obstacles unique to implementing effective anomaly detection. The first is that reliable training data that accurately represents typical network activity may be hard to come by. The behavioural dynamics of system components can vary, and networks are dynamic in nature (Hou *et al.*, 2016). Another concern is the durability of the training data. Historical network data may become out of date and falsely depict the network's current state. To ensure reliable detection, models for anomaly detection must be updated and retrained on a regular basis.

1.2 Research Question

One of the key issues with establishing network intrusion detection systems is reducing the amount of false positives while still identifying a large proportion of legitimate network traffic. Because different machine learning algorithms examine data in different ways, selecting the best answer may be difficult. The suggested research project will answer the question of "Which machine learning technique is more effective for intrusion detection?".

1.3 Research Objectives

The research objectives of this study include a thorough examination of machine learning approaches within the context of Network Intrusion Detection Systems (NIDS). The goal of this research is to look at how machine learning and deep learning may enhance model development and data representation. This is accomplished by designing and implementing a detection system using deep learning and machine learning algorithms, as well as evaluating the effectiveness of various machine learning techniques such as Decision Tree (DT), Logistic Regression (LR), Naive Bayes (NB), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) in binary classification tasks using the CICIDS2017 and UNSW NB15 datasets as the benchmark. In addition, the research compares deep learning and machine learning algorithms utilising accuracy, F1-score, and area under curve (AUC). The impact of feature selection, hyperparameter tweaking on the chosen machine, and deep learning model parameters on classification accuracy are also thoroughly examined. The aforementioned study aims to offer valuable insights into binary classification cases, model parameter optimization, and a more profound understanding of the effectiveness of machine learning techniques for network intrusion detection systems.

1.4 Summary

Network security challenges persist despite research efforts. Intrusion attempts to compromise data and system integrity. Rising malware and denial of service attacks highlight the need for cybersecurity measures. Key challenges include increased network traffic, NIDS complexity, and diverse data transmission. This research aims to enhance Network Intrusion Detection using optimized machine learning and deep learning models. The models are LR, DT, NB,

CNN, and RNN and were evaluated using the accuracy, F1-score, and AUC on the CICIDS 2017 and UNSW NB15 datasets.
Subsequent sections of this research are structured as follows: In Chapter 2, an examination of intrusion detection systems is presented, encompassing an overview of their fundamental concepts, the various types of intrusion detection systems, the technical methodologies employed in network intrusion detection systems, and a critical review of pertinent literature. Chapter 3 delineates the research methodology applied in this study, elucidating the chosen research approach, data collection techniques, the employed models, and the system design. Chapter 4 provides an intricate account of the implementation. Chapter 5 engages in a comprehensive discourse concerning the research results and the encountered limitations. Finally, Chapter 6 concludes the research and suggests recommendations for further exploration within this domain.

2 Literature Review

A computer vision technique, machine learning, comprises the automatic training of machines to categorise and recognise various sorts of data, such as photos, movies, objects, sceneries, and more. Machine learning is built on particular algorithms that analyse, learn from, and make judgements from raw data. When it comes to analysing raw data, most machine learning approaches have limitations (Liu and Lang, 2019). These approaches need a thorough understanding and competence in feature selection, as well as rigorous engineering. A newer set of techniques, on the other hand, allows a system to be given a dataset and automatically derive the required representations for decision-making, such as detection or classification. SVM (Al-Qatf *et al.*, 2018; Marir *et al.*, 2018; Wu *et al.*, 2020), genetic algorithm (Tao, Sun and Sun, 2018; Zhang, Li and Wang, 2019; Elhefnawy, Abounaser and Badr, 2020) and random forest (Jiong Zhang, Zulkernine and Haque, 2008; Farnaaz and Jabbar, 2016; Waskle, Parashar and Singh, 2020) are some of the popular machine learning models used in intrusion detection systems. A more sophisticated method that enables computers to independently extract, evaluate, and understand important information from unprocessed data is called deep learning, a subset of machine learning. The outcomes of deep learning outperform those of traditional machine learning techniques. A multi-layered, nonlinear model is used by deep neural networks to help the system understand complex correlations between input and output data. The capacity of deep learning to automatically extract characteristics from unprocessed data, interpret it, and make judgements based on this knowledge gives it an advantage over classical machine learning (Chauhan and Singh, 2018). Recent studies have applied numerous deep learning algorithms such as, KNN (Liao and Vemuri, 2002; Li *et al.*, 2014; Wazirali, 2020), and LSTM (Althubiti, Jones and Roy, 2018; Hossain *et al.*, 2020; Ts and Shrinivasacharya, 2021) with the goal of curbing network attacks.

2.1 Intrusion Detection System

Intrusion can be characterized as any unauthorized actions resulting in harm to an information system. This encompasses any form of attack that potentially jeopardizes the confidentiality, integrity, or availability of information, thereby classifying it as an intrusion. For instance, actions that render computer services unresponsive to legitimate users are deemed as intrusions (Liao *et al.*, 2013; Khraisat *et al.*, 2019). An intrusion detection system (IDS) is a security tool that continually analyses host and network traffic for any odd activity that might signal a breach of security policy or imperil the system's availability, confidentiality, and integrity. When it detects malicious activity, an intrusion detection system alerts network or host administrators (Denning, 1987; Vasilomanolakis *et al.*, 2015). IDS are classed based on how they are implemented or the detection techniques they use.

2.2 Deployment-based IDS

Based on their deployment strategy, IDS may be divided into two categories: network-based IDS (NIDS) and host-based IDS (HIDS). At the individual information host level, HIDS monitors all activities, looks for security policy violations, and flags any questionable behavior (Mukkamala, Janoski and Sung, 2002; Verwoerd and Hunt, 2002). One significant drawback of HIDS, however, is that it can be deployed across several hosts that need intrusion protection. This adds to the processing expenses of each node and lowers the overall performance of the IDS (Zhang, Lee and Huang, 2003). Conversely, Network Intrusion Detection Systems (NIDS) are employed at the network level to safeguard all linked devices and the network against potential attacks. NIDS continuously scans and records network traffic to look for vulnerabilities or security breaches.

2.3 Detection Method

Two types of intrusion detection systems (IDS) are available for network intrusion detection: anomaly detection-based intrusion detection (AIDS) and signature-based intrusion detection (SIDS). SIDS, often referred to as knowledge-based or misuse intrusion detection, builds attack pattern signatures (Ahmim, Derdour and Ferrag, 2018; Axelsson, no date). After then, these fingerprints are kept in a database. To identify assaults, they are then matched to data patterns. SIDS has the advantage of being able to recognise known attacks since specific signatures exist (Kabiri and Ghorbani, 2005). But because they don't have established signature patterns, it is unable to identify new or creative assaults (Uddin *et al.*, 2013). It might also take a lot of time to maintain a sizable signature database and compare it to data packets.

AIDS or behavior-related IDS, on the other hand, is predicated on establishing a precise profile for usual activities. Novel risks can be identified by classifying any deviations from this profile as anomalies or aberrant behaviour (Neri, 2000; Ma, 2020). One significant feature of AIDS is the ability to alter the typical activity profile for various networks and applications (Zhang, Shen and Sang, 2007; Guo *et al.*, 2016). Despite this, managing a high False Alarm Rate (FAR) is the main issue AIDS confronts since it can be challenging to accurately determine the border between normal and abnormal profiles for intrusion detection (Chandola, Banerjee and Kumar, 2009). The planned research effort will primarily focus on AIDS.

One popular tactic for locating and thwarting cyberattacks is the use of signature-based detection algorithms. These approaches rely on preset signatures to identify known assaults (Lee and Stolfo, 2000; Manganaris *et al.*, 2000; Bloedorn *et al.*, 2001). But they have limitations in terms of recognising new attack types in the absence of a corresponding signature. Keeping up with the constantly evolving threat landscape is challenging because each time a new assault is discovered, the signature database has to be manually updated. Because of this, intrusion detection techniques based on data mining are spreading more and more. Misuse detection and anomaly detection are the two main functions of data mining-based intrusion detection systems (Depren *et al.*, 2005; Varal and Wagh, 2018). Attacks are identified by classifying events in a dataset as either "normal" or "attack," and then using this labelled data to train a learning system. As long as the input data is properly labelled, these methods may automatically adjust and retrain intrusion detection models on a variety of input data, including new sorts of attacks. The categorization of network intrusions utilising several popular data mining techniques, uncommon class prediction models, association rules, and cost-sensitive modelling have been the main topics of intrusion detection research. Misuse detection models, which are generated autonomously as opposed to signature-based systems, can be more sophisticated and precise than manually created signatures. To get over this restriction, researchers are looking on techniques that combine anomaly and misuse detection. By incorporating the benefits of both approaches, these hybrid models aim to improve the detection of known as well as unknown assaults (Aljamal *et al.*, 2019). Finding deviations from typical patterns using statistical analysis or machine learning techniques is the aim of anomaly detection. This approach has the potential to detect novel and yet undiscovered threats by identifying distinct patterns or behaviours. However, high false-positive rates and the challenge of distinguishing between normal and abnormal behaviour can also impede anomaly identification.

2.4 Related Work

(Belouch, El Hadaj and Idhammad, 2018) used Apache Spark to assess a number of classification methods (SVM, Decision Tree, Naive Bayes, and Random Forest) on the massive UNSW-NB15 dataset, which includes all 42 attributes. About 257,000 records were included in their analysis, a considerable increase above the quantity of records used in this investigation. 30% and 70% of the training and testing datasets, respectively, were divided. With an amazing accuracy rate of 97%, the definitive findings demonstrated that Random Forest was the best algorithm. The accuracy of the NB and DT models was 74.19% and 95.82%, respectively.

Utilising a random forest classifier, however, (Tama and Rhee, 2017) conducted a comparative evaluation of IDSs with an emphasis on two critical performance metrics: accuracy and false alarm rate. NSL-KDD, UNSW-NB15, and GPRS are three distinct IDS datasets that were examined with a 10-fold cross-validation method. The study also looked at the effectiveness of the Decision Tree, NB-Tree, and Multilayer Perceptron (MLP) classifiers. Lastly, their results proved the effectiveness of the proposed model, which employed cross-validation and the Random Forest classifier with well-calibrated parameter values.

Using machine learning approaches, the system developed by (Koroniotis *et al.*, 2018) was centred on the forensics of IoT botnet operations. After identifying 10 key characteristics from the UNSW-NB15 dataset using the information gain approach, they classified the dataset using four machine learning techniques: association rule mining (ARM), näive Bayes (NB), artificial neural network (ANN), and decision tree. According to the simulation findings, the decision tree approach (C4.5) performed best at differentiating between botnet traffic and regular network traffic, with an accuracy of 93.23% compared to 72.73% for NB. At 63.97% accuracy, the deep learning ANN model had the lowest performance.

(Azizjon, Jumabek and Kim, 2020) presented a deep learning strategy for constructing effective and adaptable IDS utilizing a one-dimensional Convolutional Neural Network architecture with 32 convolution filters in this study. They used max pooling for downsampling and regularization, with pooling size 2 and stride size 1. The network culminates with two completely linked classification layers. The results of their experiment showed that the 1D-CNN with three layers beat the other variations with one layer, two layers, the LSTM layer, random, and SVM models, with accuracy and F1-scores of 91.2% and 91.59%, respectively.

2.5 Ethics

The problems brought up by a network intrusion detection project containing attack traffic must be resolved in order to guarantee ethical research practises. Two important considerations are the data's source and any potential harm from using the data to perform an actual assault. Attack traffic utilisation may inadvertently reveal system weaknesses and vulnerabilities if it is not properly controlled, which might lead to breaches and compromises. This calls into question what constitutes proper disclosure as well as the consequences of making such material public. There may be privacy problems for individuals or organisations whose data is included in the dataset without their consent. Upholding ethical integrity requires striking a compromise between doing pertinent research and defending people's right to privacy about their personal information. To protect the dataset from any misuse or unauthorised access, I will take the required steps to anonymize and preserve it. When creating detection models based on these datasets, the potential impact on real-world networks should be considered, since doing so may result in inadvertent false positives or interfere with genuine network activity.

2.6 Summary

In summary, past research has repeatedly demonstrated that machine learning approaches outperform traditional methods such as rule-based systems, signature-based detection, or anomaly detection alone in efficiently tackling network intrusion detection. Furthermore, several research either did not use strong and varied datasets or concentrated on a narrow range of intrusion occurrences, thereby losing out on the whole spectrum of network threats. Notably, efficient network intrusion detection demands a thorough grasp of the shifting strategies used by hostile actors to fool and breach network security.

3 Research Methodology

This study uses the CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology, which is widely recognised for achieving the objectives of data mining and machine learning research. It provides an organised framework that ensures the effectiveness and efficiency of data-driven initiatives. The CRISP-DM process consists of six crucial stages: Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment.

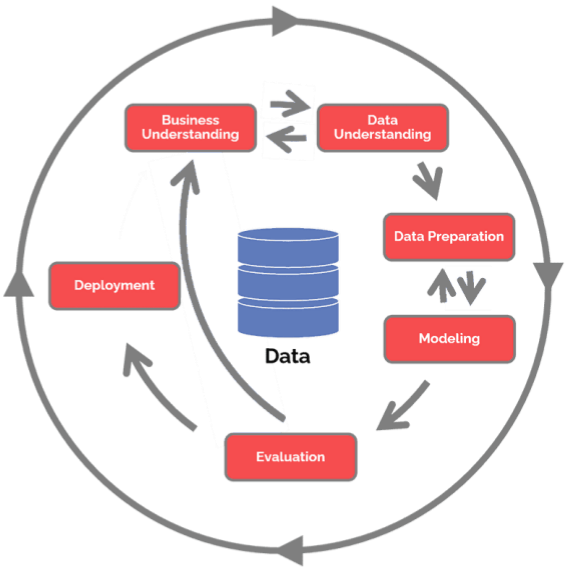

Figure 1. CRISP-DM Methodology (Huber et al., 2019)

3.1 Business Understanding

From a business standpoint, the value of a network intrusion detection system (NIDS) that employs machine learning in safeguarding a company's network infrastructure and sensitive data from online threats and assaults is best appreciated. As businesses rely more on digital operations and networked technology, the risk of potential attacks and security breaches grows. A machine learning-powered NIDS detects and mitigates aberrant behaviour, improving overall cybersecurity posture. By utilising complex algorithms and pattern recognition techniques, the system may learn from earlier data and respond to new threats, boosting its accuracy and efficacy over time.

3.2 Data Understanding

The CICIDS 2017 and UNSW-NB15 datasets together provide a large and diversified collection of network traffic data for cybersecurity research and analysis. The UNSW-NB15 dataset combines real-world network activities with synthetic attacks, captured using the tcpdump tool and totaling 100 GB. The attack types are Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms., resulting in 49 features with class labels and were obtained from UNSW Canberra’s Cyber Lab. The CICIDS2017 dataset comprises recent common attacks and benign data, closely resembling real-world network traffic. The dataset attack types include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS, and takes up 51.1 GB of space. The dataset was collected from the Canadian Institute for Cybersecurity.

3.3 Data Processing

The first phase of the analyses was the data preprocessing and data cleaning. In this phase, the column names were cleaned to remove white spaces and special characters using the clean_dataframe_column_names function.

Next, the pandas object data types for the datasets were enforced for the source and destination IP address columns.

Next, the dependent variable column was created for the CICIDS2017 dataset as a binary class distribution. The column helped to label the traffic types into attack traffic labeled as 1 and normal traffic labeled as 0. The column labeled “label” was dropped from the column list. For the UNSW NB15 dataset, the columns labeled “label” and “attackcat” were renamed “traffictype” and “trafficcategory” respectively. Then, a check was done for missing values on both datasets. This entailed using the numpy library to check for infinite values in the dataset and convert them to NaN (Not a Number) value. All NaN or missing values were dropped from the dataset. For the UNSW NB15 dataset, the columns "ctflwhttpmthd" ,"isftplogin", "trafficcategory" had 1,348,145, 1,429,879 and 2,218,764 missing records respectively, hence, these columns were dropped from the dataset.

3.3.1 Exploratory data analysis

In exploring the features in the datasets, Seaborn and Matplotlib were used to create pie charts to visualize the class distribution in the dataset. The CICDS 2107 dataset contains 2,830,743 records and 79 columns. 2,273,087 was the number of normal traffic records and the attack traffic was 557,656 records. The UNSW NB15 dataset has 2,540,047 records and 49 columns. The number of normal traffic and attack traffic records were 2,220,001 and 500,389 respectively. Figures X and Y below show the class distribution of the dependent variable for CICIDS 2017 and UNSW NB15 datasets.

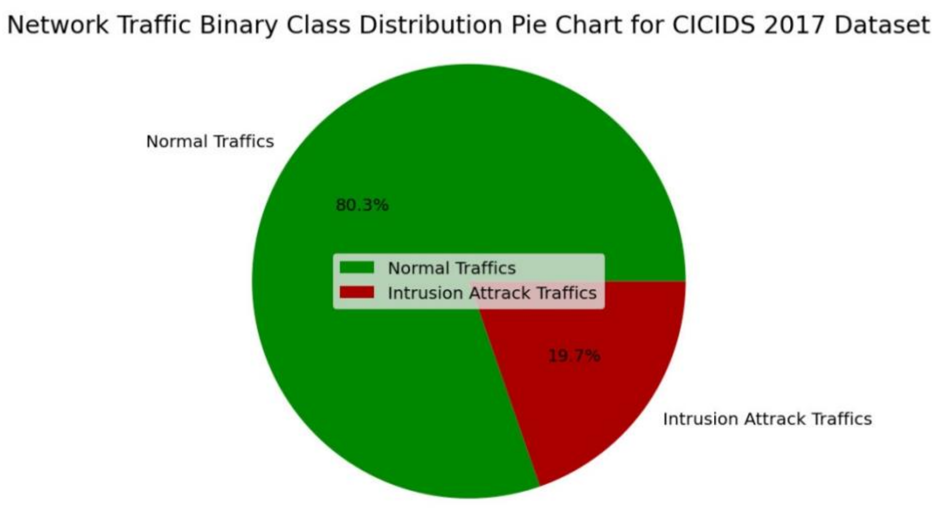

Figure 2. CICIDS 2017 Dataset Class Distribution

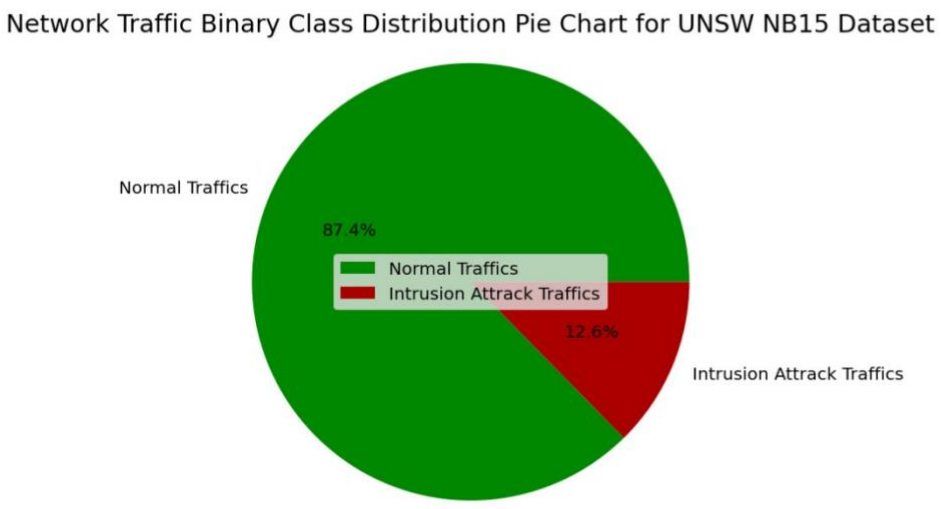

Figure 3. UNSW NB15 Dataset Class Distribution

For the numerical data types in the datasets, the described method in pandas dataframe was used to view the descriptive statistics summary for the target columns. For the object data type columns, pandas series value_counts method was used to count the frequency of each category in the targeted column.

3.3.2 Multicollinearity analysis

It refers to the presence of high correlation or interdependence between two or more independent variables (predictors) in a regression model. Multicollinearity can complicate the interpretation of regression results and can lead to unstable coefficient estimates. To address this issue, the perform_multicollinearity_analysis was created to calculate the correlation between the independent variables and retrieve column names from the dataset whose correlation estimate is greater than or equal to 0.7 which demonstrates high correlation between the target variables. The retrieved columns were then removed from the datasets. Figures X and Y below show the correlation heatmap after the multicollinearity analysis. They both demonstrate that no 2 variables have a correlation estimate above 0.7. Hence, multicollinearity between the variables have been eliminated.

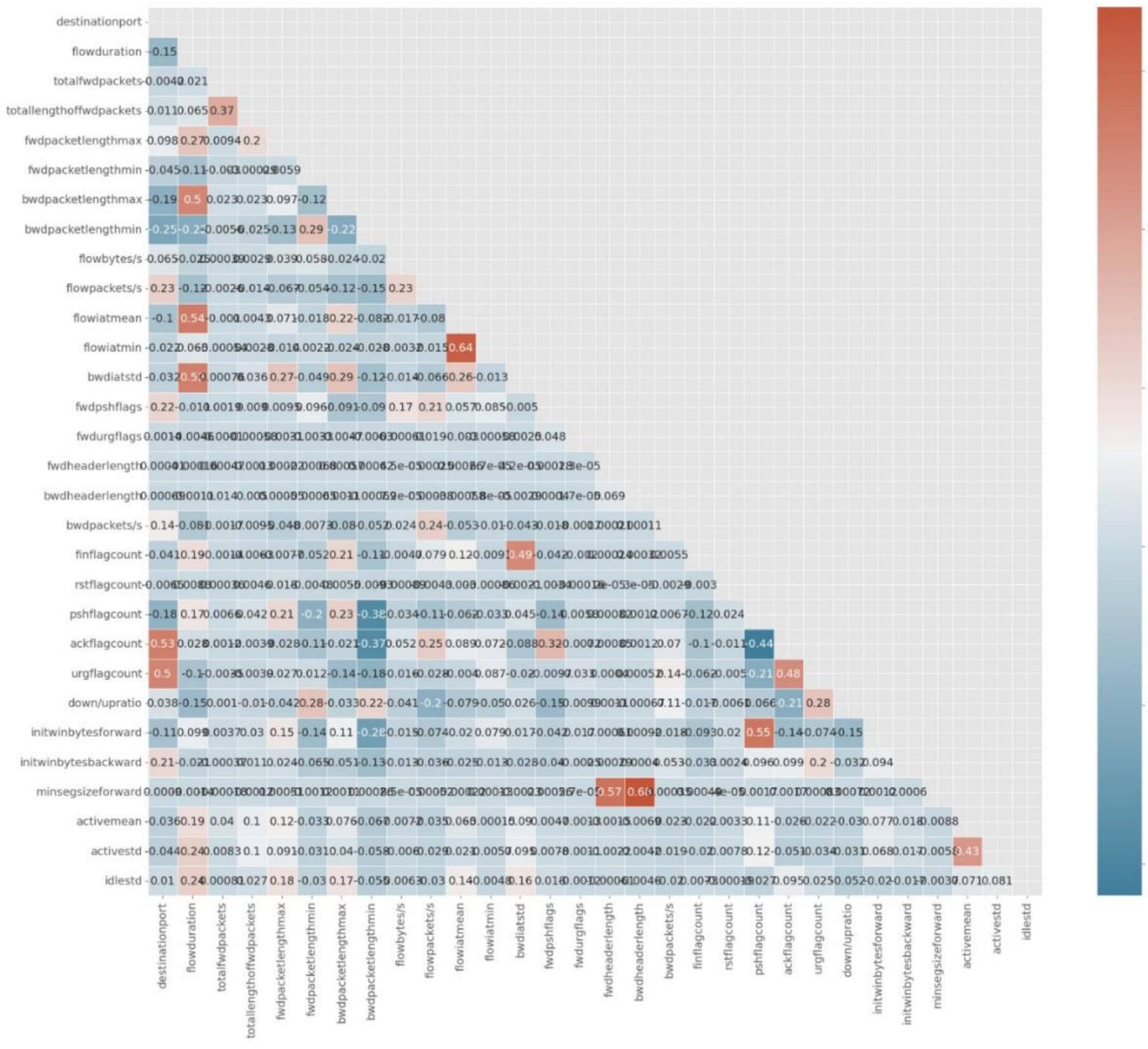

Figure 4. CICIDS 2017 Dataset Correlation Heatmap

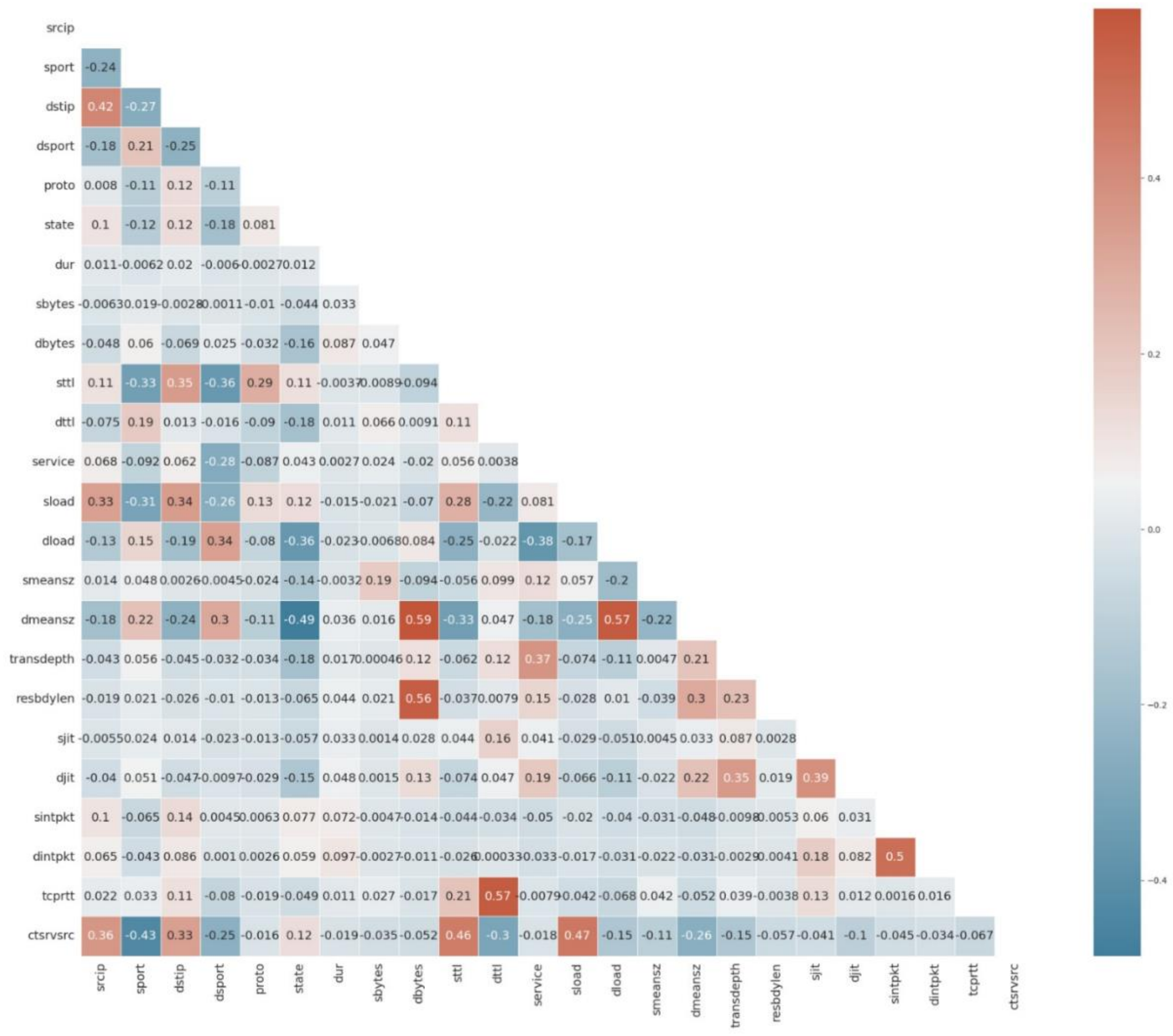
Figure 5. UNSW NB15 Correlation Heatmap

3.4 Modeling

I looked into five machine learning models: decision trees (DT), naïve bayes (NB), logistic regression (LR), convolutional neural networks (CNN), and recurrent neural networks (RNN) in order to develop effective and efficient IDS. Several machine learning techniques are used at this stage, such as recurrent neural networks, logistic regression, naive bayes, decision trees, and convolutional neural networks. The preprocessed dataset is used to train the models. This dataset includes labelled samples where the labels indicate whether the data relates to potentially invasive behaviour or normal behaviour. The features in the dataset reflect different network traffic properties. Throughout the process, the model's parameters are adjusted to maximise accuracy and performance. To make sure the models can correctly distinguish between legitimate and malicious network activity, they are tested using a distinct validation dataset after training. The models' resilience and effectiveness are then strengthened on fresh data through the use of the cross-validation and hyperparameter modification approaches. Below is a description of the models.

3.4.1 Logistic Regression

A statistical approach known as logistic regression is critical in binary classification jobs for network traffic data. The model in this technique aims to estimate the parameters of a logistic function, which connects input information to the risk of network traffic being classified as an attack. The logistic regression model is trained by using labeled data to derive these parameters with the help of feature selection, which aims to minimize a cost function and improve accuracy, frequently employing the cross-entropy loss, which quantifies the difference between predicted probabilities and actual labels (Rani *et al.*, 2023). Once properly trained, the model may classify fresh, unlabeled data by generating predicted probabilities based on input characteristics and estimated parameters.

3.4.2 Naïve Bayes

The Nave Bayes classifier operates under the strong premise of independence, which means that the likelihood of one characteristic does not impact the likelihood of another. The Nave Bayes classifier makes personalized assumptions for each of n attributes when dealing with a group of n attributes. Surprisingly, the Nave Bayes classifier frequently produces accurate results. The study offered in this paper digs into the conditions and causes behind the Nave Bayes classifier's performance. It implies that classification mistakes may be attributable to three factors: noise in training data, bias, and variance. To reduce training data noise, high-quality training data must be chosen. Furthermore, the machine learning system should categorize the training data (Amor, Benferhat and Elouedi, 2004).

3.4.3 Decision Tree

Using Decision Trees (DT) for network intrusion detection requires little user involvement during the dataset training phase. DT uses the network traffic data to perform feature selection and variable analysis on an efficient basis throughout this training phase. Decision trees reduce the uncertainty usually involved with decision-making by providing clear values for issues, options, and the outcomes connected with each decision. Especially, decision trees perform better than other machine learning techniques because they thoroughly investigate every possible scenario and methodically analyze every alternative through to the end. This method is very helpful as it is easy to understand and makes it possible to evaluate different decision tree nodes in a clear manner (Amor, Benferhat and Elouedi, 2004).

3.4.4 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a type of neural network that uses convolution operations to extract meaningful feature representations from input. A CNN's core design consists of two main layers: a convolutional layer and a pooling layer. These layers are optimized to suit the purpose of identifying and categorizing network intrusions, potentially enabling the classification of different intrusion types. Notably, the pooling layer is critical in minimizing the number of parameters linking the convolutional layers, lowering computing burden, and increasing the effective receptive field of the subsequent convolutional layers (Chen *et al.*, 2020).

3.4.5 Recurrent neural networks

Recurrent neural networks (RNNs) are made up of input units, output units, and hidden units, with the latter being crucial to their operation. Information travels unidirectionally from the input units to the hidden units in the RNN model, with information synthesis from the previous temporal hidden unit to the present temporal hidden unit. These hidden units may be thought of as the network's memory, storing information from beginning to finish. When the RNN is unfolded, it is clear that it contains deep learning concepts. RNNs provide an effective method for supervised classification learning (Yin *et al.*, 2017).

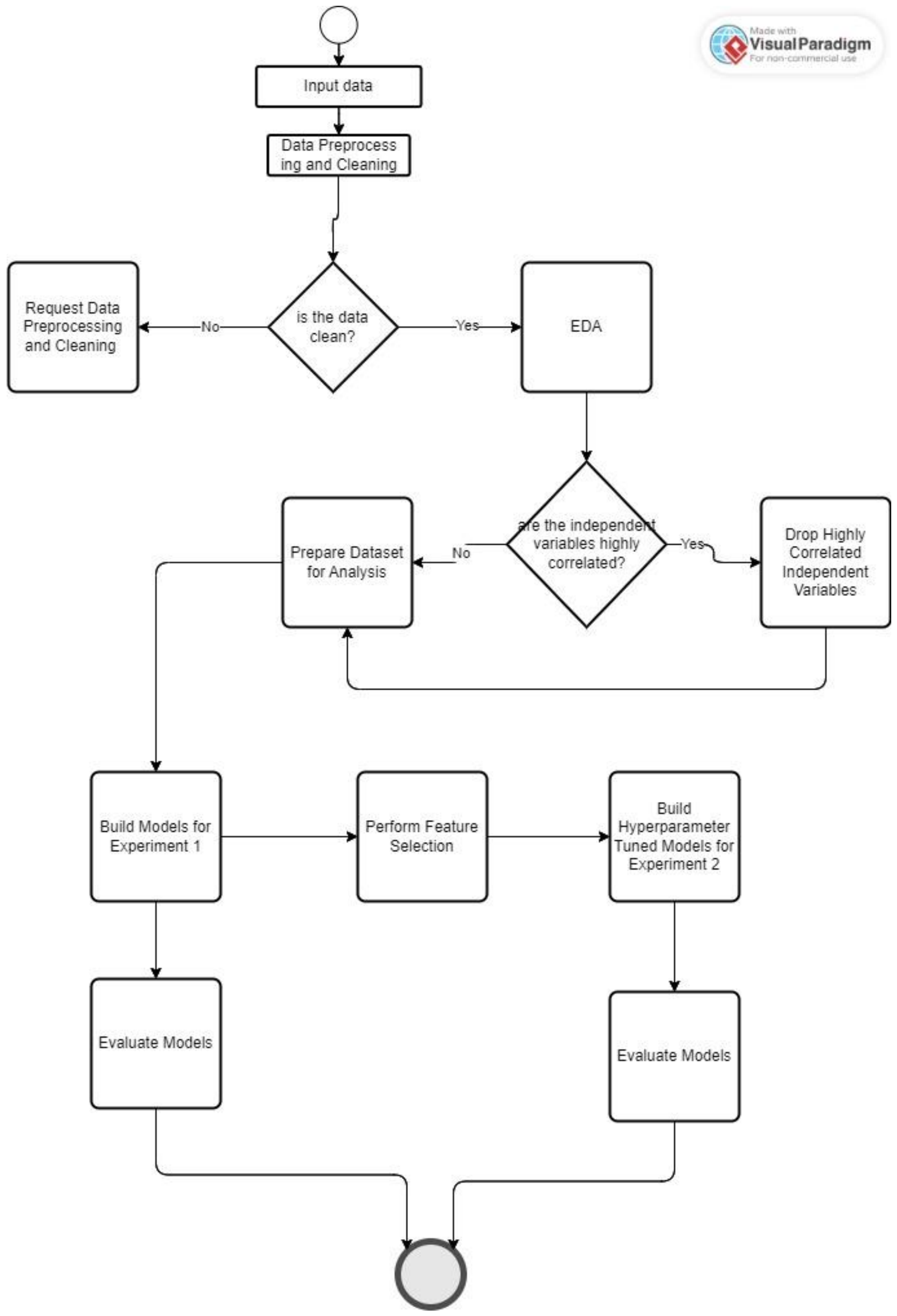
Made with Visual Paradigm For non-commercial use
Input data
Data Preprocessing and Cleaning
is the data clean?
No
Yes
Request Data Preprocessing and Cleaning
EDA
are the independent variables highly correlated?
No
Yes
Prepare Dataset for Analysis
Drop Highly Correlated Independent Variables
Build Models for Experiment 1
Perform Feature Selection
Build Hyperparameter Tuned Models for Experiment 2
Evaluate Models
Evaluate Models

Figure 6. Flow Diagram of System Design

3.5 Evaluation

Users may often access massive datasets containing normal and attack traffic types to evaluate network intrusion detection systems. The proportion of correctly classified network data is used as an extra statistic for evaluating the classification algorithm's performance. It is the proportion of network data packets that are correctly classified.

Accuracy: The accuracy of a machine learning model is expressed as a percentage of all correctly predicted outcomes. The accuracy of machine learning models is calculated by dividing the number of successfully categorised examples by the total number of instances in the dataset.

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

F1-score: An indicator of a model's overall accuracy is the F1-score. It is the harmonic mean of accuracy and recall, two more crucial performance indicators for the model. The degree to which optimistic predictions prove to be true positives is known as precision. Recall is the proportion of outstanding results that were accurately predicted to be good outcomes. F1-score may be written as

$$F1 - Score = \frac{2 * \text{Precision} * \text{Recall}}{\text{Recall} + \text{Precision}}$$

Area Under Curve (AUC): The true positive rate (TPR) against false positive rate (FPR) shown at different categorization thresholds is known as the Receiver Operating Characteristic (ROC) curve, and the AUC is the area under the curve. The fraction of true positive instances that are accurately projected as positive is known as the TPR, whereas the fraction of true negative cases that are wrongly predicted as positive is known as the FPR.

$$AUC = \int (\text{TP} / \text{TP} + \text{FN})\, \text{d}(\text{TN} / \text{TN} + \text{FP})$$

Where:

True Positive, or TP, is a measure of how many instances were accurately anticipated to be positive.

The number of occurrences accurately anticipated as negative is indicated by the symbol TN (True Negative).

The number of occurrences that are erroneously projected as positive is known as FP (False Positive).

The number of occurrences that are erroneously predicted as negative is known as FN (False Negative).

3.6 Deployment

To guarantee its smooth integration into an active network environment, several important considerations need to be considered at this stage. It's time to apply the model in real-time operations after training and confirming it using historical data. In order to ensure that the model can manage incoming network traffic efficiently and react to changing network traffic and potential threats in a timely manner, this technique requires scalability and latency optimisation. To maintain the model accurate over time and adjust to new assault patterns, constant updates and ongoing monitoring are needed. A network intrusion detection system that is successful in its deployment will be able to quickly identify and address security threats while protecting the confidentiality and integrity of the network. This phase, however, is outside the purview of this study.

3.7 Summary

To summarize, the methodology described in this chapter provides a solid framework for detecting network intrusions using machine learning techniques. I created a dependable method for discriminating between malicious and normal network activity by integrating several preprocessing processes and feature selection. My solution was flexible and successful, employing a variety of machine learning approaches such as logistic regression, naive bayes, decision trees, convolutional neural networks, and recurrent neural networks. The combination of critical feature selection and model optimisation has allowed for accurate differentiation of harmful and non-malicious network behavior. The study's findings validate my strategy to solve the issues provided by network intrusion while adhering to legal and ethical requirements.

4 Implementation

In this chapter, I discussed the design and implementation of the models for the experiments in this research. Two experiments were performed in this study, the first experiment was aimed at creating a baseline implementation for each of the models to be analyzed in this study.

The first experiment initializes each model with a set of parameters that help to define a minimalist implementation of the models.

The second experiment is designed to create optimal implementation for each of the models using feature importance as a way to select the most relevant features in the dataset, and hyperparameter tuning to select the optimal parameters for the models.

4.1 Logistic Regression

In Experiment 1, I utilized the "run_logistic_regression_analysis" function to implement Logistic Regression (LR) analysis. The function accepted six parameters: "train_x," "train_y," "test_x," "test_y," "dataset," and "experiment." These parameters defined the training and testing datasets and provided labels for dataset and experiment identification. I imported the LogisticRegression module from Scikit-Learn and instantiated it with key parameters to create the LR model. The model was trained with the training dataset and used for predictions. Model performance analysis was conducted using the "show_model_analysis_summary" function, comparing predictions with the actual "test_y" dataset.

In Experiment 2, I aimed to optimize the LR model's performance through feature importance analysis and hyperparameter tuning. The "perform_feature_selection_using_feature_importance" function was used for feature selection, employing the ExtraTreeClassifier module to build an ensemble model. This model identified the most relevant features in the target dataset, resulting in a list of approximately two-thirds of the most important independent variables. For hyperparameter tuning, the "run_hyper_parameters_tuned_logistic_regression_analysis" function was employed. It retained the same parameters as the base model but included selected columns from the feature selection process in the "train_x" parameter. The hyperparameters considered for training the LR model included:

- "penalty" with values: "l1," "l2," and "elasticnet."
- "C" with values set using "np.logspace(-4, 4, 20)."
- "solver" with values: "lbfgs," "newton-cg," "liblinear," "sag," and "saga."
- "max_iter" with values: 2500 and 5000.

The hyperparameter-tuned model was created using GridSearchCV, utilizing the LogisticRegression module, a parameter grid, and a cross-validation variable. This model underwent training with the training dataset and was used for generating predictions. Model performance evaluation was conducted through the "show_model_analysis_summary" function, comparing predictions with the actual "test_y" dataset.

4.2 Naïve Bayes

In Experiment 1, the base model employed the Gaussian Naive Bayes (NB) analysis using the "run_naive_bayes_analysis" function. This function utilized default parameters: "train_x," "train_y," "test_x," "test_y," "dataset," and "experiment." These parameters defined the training and testing datasets and provided labels for dataset and experiment identification. The GaussianNB module from Scikit-Learn was imported and instantiated with one parameter to create the Naive Bayes model. The model was then trained with the training dataset and used to make predictions. Model performance analysis was conducted using the "show_model_analysis_summary" function, comparing predictions to the actual "test_y" dataset.

Experiment 2 aimed to optimize the Naive Bayes model's performance through feature importance analysis and hyperparameter tuning. Feature selection was carried out using the "perform_feature_selection_using_feature_importance" function. An ensemble model was built using the ExtraTreeClassifier module to select the most relevant features from the target dataset, resulting in a list of approximately two-thirds of the most important independent variables. For hyperparameter tuning, the "run_hyper_parameters_tuned_naive_bayes_analysis" function was used, retaining the same parameters as the base model. The key hyperparameter considered for tuning was "var_smoothing," with values set using "np.logspace(0, -9, num=100)." The hyperparameter-tuned model was created through GridSearchCV, using the GaussianNB module, a parameter grid, and a cross-validation variable. This model underwent training with the training dataset and was utilized to generate predictions. Model performance evaluation was conducted through the "show_model_analysis_summary" function, comparing predictions with the actual "test_y" dataset.

4.3 Decision Tree

In Experiment 1, the base model involved implementing a Decision Tree (DT) analysis using the "run_decision_tree_analysis" function. This function accepted six parameters: "train_x," "train_y," "test_x," "test_y," "dataset," and "experiment." These parameters described the training and testing datasets and provided labels for dataset and experiment identification. The Decision Tree model was constructed using the DecisionTreeClassifier module from Scikit-Learn. It was initialized with three key parameters: "max_depth" set to 10, "criterion" set to "entropy," and "random_state" set to 42. After model construction, training was performed using the training dataset, and the resulting model was used for predictions. Model performance was assessed through the "show_model_analysis_summary" function, which compared predictions with the actual "test_y" dataset.

For Experiment 2, feature importance and hyperparameter tuning were applied to optimize the Decision Tree model's performance. Feature importance analysis was conducted using the "perform_feature_selection_using_feature_importance" function. It utilized the ExtraTreeClassifier module to build an ensemble model and selected the most relevant features from the target dataset, yielding a list of approximately two-thirds of the most important independent variables. Hyperparameter tuning was carried out using the "run_hyper_parameters_tuned_decision_tree_analysis" function. It retained the same parameters as the base model but incorporated selected columns from the feature selection process in the "train_x" parameter. The hyperparameters considered for training the Decision Tree model included:

- "max_depth" with values: 2, 3, 5, 10, and 20.
- "min_samples_leaf" with values: 5, 10, 20, 50, and 100.
- "criterion" with values: "gini" and "entropy."

The hyperparameter-tuned model was created through GridSearchCV, utilizing the DecisionTreeClassifier module, a parameter grid, and a cross-validation variable. After training the model with the training dataset, predictions were generated and model performance was assessed using the "show_model_analysis_summary" function.

4.4 Convolutional Neural Network

In Experiment 1, the base model utilizes a Convolutional Neural Network (CNN) for analysis. The "run_convolution_neural_network_analysis" function accepts six parameters: "train_x," "train_y," "test_x," "test_y," "dataset," and "experiment." These parameters represent the training and testing datasets, as well as labels for dataset and experiment identification. The CNN model is constructed using TensorFlow Keras libraries. The "build_convolution_neural_network" function is instrumental, requiring a "dim" parameter, signifying the number of features in the independent variable group. The model architecture comprises Sequential, Convolution1D, MaxPooling1D, Flatten, Dense, and Dropout modules. The Sequential module serves as the framework for layer stacking. The initial layers consist of Convolution1D modules, followed by MaxPooling1D and Flatten modules. Subsequently, Dense and Dropout modules are introduced. The final two layers represent the output layer, defining the neural network structure. The model is compiled with "binary_crossentropy" as the loss function, "adam" as the optimizer, and "accuracy" as the metrics. Model instantiation is achieved using the KerasClassifier wrapper, specifying the "build_convolution_neural_network" function, "epochs" set to 10, and "batch_size" set to 128. Training is conducted on the training dataset, followed by prediction generation. Model evaluation is performed through the "show_model_analysis_summary" function.

In Experiment 2, I focused on feature importance and hyperparameter tuning to optimize the CNN model's performance. The "perform_feature_selection_using_feature_importance" function is used for feature selection, employing the ExtraTreeClassifier module to build an ensemble model. It fits the independent variables in the target dataset and utilizes the "feature_importance_" property to identify the most significant independent variables. For hyperparameter tuning, the "run_hyper_parameters_tuned_convolution_neural_network_analysis" function is invoked, employing the same parameters as the base model. The "train_x" parameter contains the selected columns from the feature selection process. Hyperparameters include "batch_size" (choices: 512, 256, 128), "epochs" (choices: 30, 20, 10), "units" (choices: 32, 16, 8), and "dim," representing the number of independent variables. The hyperparameter-tuned model is created using GridSearchCV, instantiated with a KerasClassifier wrapper, the "build_hyper_parameters_convolution_neural_network" function, a parameter grid ("param_grid"), and a cross-validation parameter ("cv"). The model is trained using the training dataset, and predictions are generated. Model performance is assessed through the "show_model_analysis_summary" function, comparing predictions to the "test_y" dataset, facilitating an evaluation summary of the implemented model.

4.5 Recurrent Neural Network

In Experiment 1, the base model employs a Recurrent Neural Network (RNN) for analysis. It takes six parameters: "train_x," "train_y," "test_x," "test_y," "dataset," and "experiment." These parameters correspond to the training and testing datasets, and labels for dataset and experiment identification. The RNN model is constructed using TensorFlow Keras libraries. A pivotal function, "build_recurrent_neural_network," is employed with a single parameter, "dim," representing the number of features in the independent variable group. The model architecture comprises layers like Sequential, LSTM, Dense, and Dropout. The Sequential module serves as the framework, with alternating LSTM and Dropout layers (6 in total) for feature learning, followed by two Dense layers for the output. The model is compiled with "binary_crossentropy" loss, "adam" optimizer, and "accuracy" metrics. A KerasClassifier wrapper is used for the model instantiation, specifying the "build_recurrent_neural_network" function, "epochs" set to 10, and "batch_size" set to 128. Model training is conducted on the training dataset, followed by prediction generation. Evaluation is performed using the "show_model_analysis_summary" function.

In Experiment 2, I focused on optimizing the RNN model through feature selection and hyperparameter tuning. Feature selection is conducted using the "perform_feature_selection_using_feature_importance" function, which employs the ExtraTreeClassifier ensemble model to select the most relevant features from the target dataset. This process yields a list of approximately two-thirds of the most significant independent variables. For hyperparameter tuning, I employed the "run_hyper_parameters_tuned_recurrent_neural_network_analysis" function. It uses the same parameters as the base model, with specific values for "batch_size" (choices: 512, 256, 128), "epochs" (choices: 30, 20, 10), "units" (choices: 32, 16, 8), and "dim," representing the number of independent variables. The hyperparameter-tuned model is established using GridSearchCV, instantiated with a KerasClassifier wrapper and a parameter grid containing the specified values. The "cv" variable represents cross-validation. After training the model with the training dataset, predictions are generated. Model performance is assessed using the "show_model_analysis_summary" function, which compares predictions to the "test_y" dataset, providing an evaluation summary of the model's performance.

5 Evaluation

The findings of the two tests on the CICIDS 2017 and UNSW-NB15 datasets give useful insights into the performance of alternative machine learning models for network intrusion detection while taking into account changes in dataset complexity and optimisation strategies.

5.1 Experiment 1

In the first experiment, which aimed to establish baseline implementations, Decision Trees (DT) emerged as the best-performing model in both datasets, with remarkable accuracy of 99.42% and 99.40%, F1-scores of 99.41% and 99.39%, and AUC scores of 99.42% and 99.40% for CICIDS 2017 and UNSW-NB15, respectively. This discovery is intriguing since DT is recognised for its ability to handle complicated decision boundaries, which may be especially important in network intrusion detection. It is important to note, however, that Naive Bayes (NB) fell substantially behind, with accuracy rates of 69.72% and 81.83% for the two datasets, emphasising its limits in dealing with the different nature of incursion behaviours. Tables 1 and 2 below shows the result of the experiment on the two datasets.

Table 1. Models Performance Results on CICIDS 2017 Dataset

Models	Accuracy	F1-score	AUC
LR	90.53%	90.78%	90.53%
NB	69.72%	75.89%	69.72%

DT	99.42%	99.41%	99.42%
CNN	95.98%	95.92%	95.98%
RNN	95.03%	95.07%	95.03%

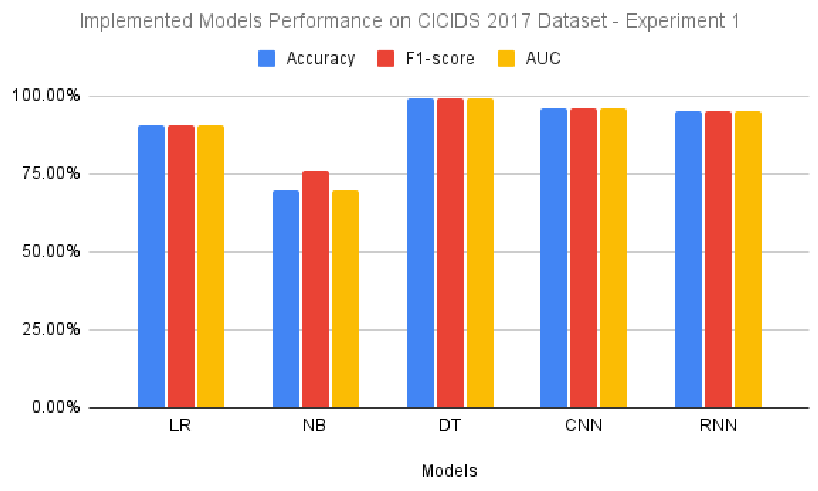

Figure 7. Model Comparison Bar Chart on CICIDS 2017 Dataset

Table 2. Models Performance Results on UNSW NB15 Dataset

Models	**Accuracy**	**F1-score**	**AUC**
LR	0.9220	0.9236	0.9220
NB	0.8183	0.8354	0.8183
DT	**0.9940**	**0.9939**	**0.9940**
CNN	0.9577	0.9579	0.9577
RNN	0.9565	0.9566	0.9565

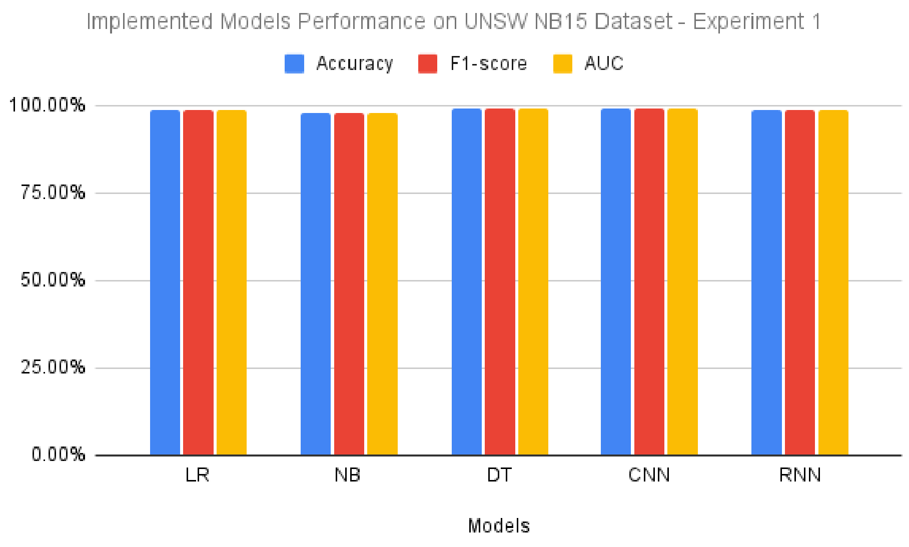

Figure 8. Model Comparison Bar Chart on CICIDS 2017 Dataset

5.2 Experiment 2

The second experiment, which focused on optimisation via feature importance and hyperparameter tuning, demonstrated a significant improvement across all models. Logistic Regression (LR) and Decision Trees (DT) excelled in this environment, with accuracy rates of 99.07% and 99.13% for CICIDS 2017 and UNSW-NB15, respectively. With accuracy rates of 99.20% and 99.18%, Convolutional Neural Network (CNN) displayed persistent brilliance, indicating its ability to excel in complicated intrusion detection scenarios. Despite somewhat poorer performance than some other models, Recurrent Neural Network (RNN) maintained excellent accuracy and F1-scores, with rates of 99.22% and 99.18%, showing its use in network intrusion detection.

Table 3. Models Performance Results on CICIDS 2017 Dataset

Models	**Accuracy**	**F1-score**	**AUC**
LR	99.07%	99.06%	99.07%
NB	97.78%	97.79%	97.78%
DT	**99.27%**	**99.26%**	**99.27%**
CNN	99.20%	99.19%	99.20%
RNN	99.22%	99.21%	99.22%

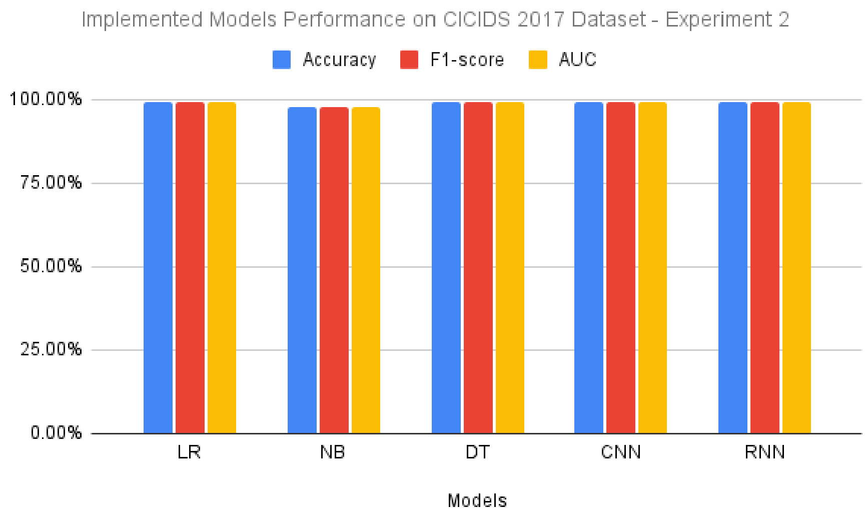

Figure 9. Model Comparison Bar Chart on CICIDS 2017 Dataset

Table 4. Models Performance Results on UNSW NB15 Dataset

Models	**Accuracy**	**F1-score**	**AUC**
LR	99.13%	99.13%	99.13%
NB	98.70%	98.69%	98.70%
DT	**99.28%**	**99.28%**	**99.28%**
CNN	99.18%	99.18%	99.18%
RNN	99.18%	99.18%	99.18%

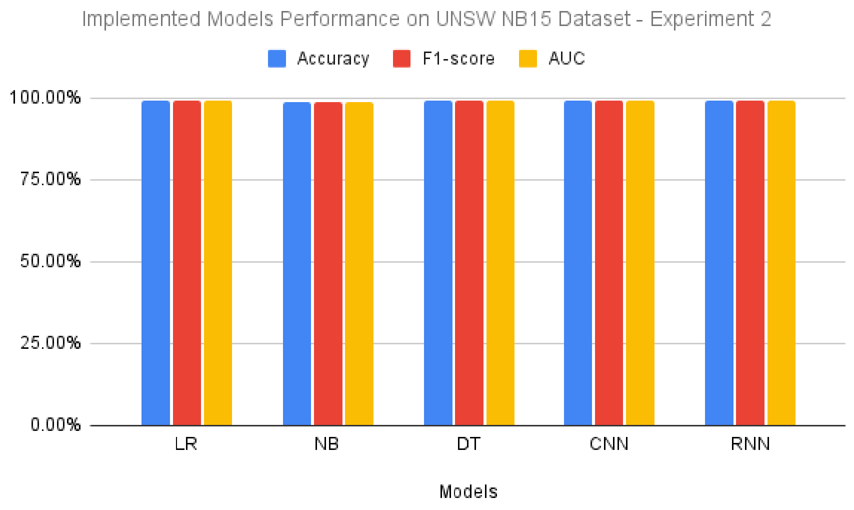

Figure 10. Model Comparison Bar Chart on CICIDS 2017 Dataset

5.3 Discussion

Overall, DT performed well across tests, datasets, and optimisation strategies, making it a trustworthy option. NB, on the other hand, remained the weakest model in all testing, showing its limitations for complicated intrusion detection tasks. In the second trial, the most improved models were LR, DT, and CNN, demonstrating the importance of optimisation in improving model performance. These findings highlight the need of rigorous model selection, dataset appropriateness, and optimisation strategies in creating successful intrusion detection systems. The findings show that when optimised, models like as LR, DT, CNN, and RNN have the ability to provide solid security solutions in real-time network monitoring, however, Naive Bayes should be used with caution in complicated intrusion detection scenarios.

When compared to other (Belouch, El Hadaj and Idhammad, 2018) and (Koroniotis *et al.*, 2018), I observed some similarities in the findings. The decision tree performed better than the other models except in the former study where the decision tree was the best model with a 95.82% accuracy. In the later study, the network intrusion detection model obtained 93.23% accuracy, which is less than the 99.28% obtained in this research. This difference could be as a result of the number of samples used.

5.4 Limitations

This research faced some challenges and hence, encountered some shortcomings. One major limitation of this study is the number of samples used in the training and testing datasets. Because of the limited computing capacity, a small portion of the datasets were used.

6 Conclusion and Future Work

Network security concerns have been exacerbated by the expansion of computer networks, leading to various intrusion attempts. These attempts jeopardize data and system confidentiality, integrity, and availability. Statistics show a rising prevalence of malware attacks and denial of service incidents, underscoring the need to address the growing dangers of cyberattacks. Three major challenges exacerbate network security: the surge in network traffic, the complexity of Network Intrusion Detection Systems (NIDS), diversity of protocols and data transmitted across modern networks. Developing an effective NIDS is crucial, with signature-based approaches proving inadequate for addressing evolving threats. This research aimed to compare various machine and deep learning techniques for Network Intrusion Detection, focusing on binary classification performance of 5 models. Two experiments were conducted. The first experiment was the implementation of the base models and the second experiment was the implementation of the optimal models using hyperparameter tuning. The outcome of the research showed that machine learning and deep learning models perform well in segregating the network traffic types into normal and attack on both the CICIDS 2017 and UNSW NB15 benchmark datasets. The results show that feature selection combined with hyperparameter tuning significantly improves performance in some models and slight improvements in others. The decision tree was the best performing model in both experiments on the CICIDS 2017 and UNSW NB15 datasets with an accuracy of 99.27% and 99.28%

respectively. These findings highlight the need of rigorous model selection, dataset appropriateness, and optimisation strategies in creating successful intrusion detection systems.

6.1 Future Work

In the future, an ensemble of feature selection techniques could be used on a larger dataset sample size and a mitigation system. Moreover, as the implemented solution relies only on a single feature selection technique and hyperparameter tuning and no mitigation, I believe it can be extended to other advanced optimization techniques and enable real-time mitigation for more robust commercial products.

References

Ahmad, Z. *et al.* (2021) ‘Network intrusion detection system: A systematic study of machine learning and deep learning approaches’, *Transactions on Emerging Telecommunications Technologies*, 32(1), p. e4150. Available at: https://doi.org/10.1002/ett.4150.

Ahmim, A., Derdour, M. and Ferrag, M.A. (2018) ‘An intrusion detection system based on combining probability predictions of a tree of classifiers’, *International Journal of Communication Systems*, 31(9), p. e3547. Available at: https://doi.org/10.1002/dac.3547.

Aljamal, I. *et al.* (2019) ‘Hybrid Intrusion Detection System Using Machine Learning Techniques in Cloud Computing Environments’, in *2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA)*. *2019 IEEE 17th International Conference on Software Engineering Research, Management and Applications (SERA)*, Honolulu, HI, USA: IEEE, pp. 84–89. Available at: https://doi.org/10.1109/SERA.2019.8886794.

Al-Qatf, M. *et al.* (2018) ‘Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection’, *IEEE Access*, 6, pp. 52843–52856. Available at: https://doi.org/10.1109/ACCESS.2018.2869577.

Althubiti, S.A., Jones, E.M. and Roy, K. (2018) ‘LSTM for Anomaly-Based Network Intrusion Detection’, in *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*. *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, Sydney, NSW: IEEE, pp. 1–3. Available at: https://doi.org/10.1109/ATNAC.2018.8615300.

Amor, N.B., Benferhat, S. and Elouedi, Z. (2004) ‘Naive Bayes vs decision trees in intrusion detection systems’, in *Proceedings of the 2004 ACM symposium on Applied computing*. *SAC04: The 2004 ACM Symposium on Applied Computing*, Nicosia Cyprus: ACM, pp. 420–424. Available at: https://doi.org/10.1145/967900.967989.

Axelsson, S. (no date) ‘Intrusion Detection Systems: A Survey and Taxonomy’.

Azizjon, M., Jumabek, A. and Kim, W. (2020) ‘1D CNN based network intrusion detection with normalization on imbalanced data’, in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Fukuoka, Japan: IEEE, pp. 218–224. Available at: https://doi.org/10.1109/ICAIIC48513.2020.9064976.

Belouch, M., El Hadaj, S. and Idhammad, M. (2018) ‘Performance evaluation of intrusion detection based on machine learning using Apache Spark’, *Procedia Computer Science*, 127, pp. 1–6. Available at: https://doi.org/10.1016/j.procs.2018.01.091.

Bloedorn, E. *et al.* (2001) ‘Data Mining for Network Intrusion Detection: How to Get Started’.

Chandola, V., Banerjee, A. and Kumar, V. (2009) ‘Anomaly detection: A survey’, *ACM Computing Surveys*, 41(3), pp. 1–58. Available at: https://doi.org/10.1145/1541880.1541882.

Chauhan, N.K. and Singh, K. (2018) ‘A Review on Conventional Machine Learning vs Deep Learning’, in *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*. *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, Greater Noida, Uttar Pradesh, India: IEEE, pp. 347–352. Available at: https://doi.org/10.1109/GUCON.2018.8675097.

Chen, L. *et al.* (2020) ‘A Novel Network Intrusion Detection System Based on CNN’, in *2020 Eighth International Conference on Advanced Cloud and Big Data (CBD)*. *2020 Eighth International Conference on Advanced Cloud and Big Data (CBD)*, Taiyuan, China: IEEE, pp. 243–247. Available at: https://doi.org/10.1109/CBD51900.2020.00051.

Debar, H., Dacier, M. and Wespi, A. (1999) ‘Towards a taxonomy of intrusion-detection systems’, *Computer Networks*, 31(8), pp. 805–822. Available at: https://doi.org/10.1016/S1389-1286(98)00017-6.

Denning, D.E. (1987) ‘An Intrusion-Detection Model’, *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING* [Preprint], (2).

Depren, O. *et al.* (2005) ‘An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks’, *Expert Systems with Applications*, 29(4), pp. 713–722. Available at: https://doi.org/10.1016/j.eswa.2005.05.002.

Dong, B. and Wang, X. (2016) ‘Comparison deep learning method to traditional methods using for network intrusion detection’, in *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*. *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, Beijing, China: IEEE, pp. 581–585. Available at: https://doi.org/10.1109/ICCSN.2016.7586590.

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

Farnaaz, N. and Jabbar, M.A. (2016) ‘Random Forest Modeling for Network Intrusion Detection System’, *Procedia Computer Science*, 89, pp. 213–217. Available at: https://doi.org/10.1016/j.procs.2016.06.047.

Guo, C. *et al.* (2016) ‘A two-level hybrid approach for intrusion detection’, *Neurocomputing*, 214, pp. 391–400. Available at: https://doi.org/10.1016/j.neucom.2016.06.021.

Hossain, M.D. *et al.* (2020) ‘LSTM-Based Intrusion Detection System for In-Vehicle Can Bus Communications’, *IEEE Access*, 8, pp. 185489–185502. Available at: https://doi.org/10.1109/ACCESS.2020.3029307.

Hou, S. *et al.* (2016) ‘Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs’, in *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*. *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*, Omaha, NE, USA: IEEE, pp. 104–111. Available at: https://doi.org/10.1109/WIW.2016.040.

Jiong Zhang, Zulkernine, M. and Haque, A. (2008) ‘Random-Forests-Based Network Intrusion Detection Systems’, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(5), pp. 649–659. Available at: https://doi.org/10.1109/TSMCC.2008.923876.

Kabiri, P. and Ghorbani, A.A. (2005) ‘Research on Intrusion Detection and Response: A Survey’.

Koroniotis, N. *et al.* (2018) ‘Towards Developing Network Forensic Mechanism for Botnet Activities in the IoT Based on Machine Learning Techniques’, in J. Hu et al. (eds) *Mobile Networks and Management*. Cham: Springer International Publishing (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), pp. 30–44. Available at: https://doi.org/10.1007/978-3-319-90775-8_3.

Lee, W. and Stolfo, S.J. (2000) *Data Mining Approaches for Intrusion Detection:* Fort Belvoir, VA: Defense Technical Information Center. Available at: https://doi.org/10.21236/ADA401496.

Li, W. *et al.* (2014) 'A New Intrusion Detection System Based on KNN Classification Algorithm in Wireless Sensor Network', *Journal of Electrical and Computer Engineering*, 2014, pp. 1–8. Available at: https://doi.org/10.1155/2014/240217.

Liao, Y. and Vemuri, V.R. (2002) 'Use of K-Nearest Neighbor classifier for intrusion detection', *Computers & Security*, 21(5), pp. 439–448. Available at: https://doi.org/10.1016/S0167-4048(02)00514-X.

Liu, H. and Lang, B. (2019) 'Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey', *Applied Sciences*, 9(20), p. 4396. Available at: https://doi.org/10.3390/app9204396.

Ma, W. (2020) 'Analysis of anomaly detection method for Internet of things based on deep learning', *Transactions on Emerging Telecommunications Technologies*, 31(12), p. e3893. Available at: https://doi.org/10.1002/ett.3893.

Manganaris, S. *et al.* (2000) 'A data mining analysis of RTID alarms', *Computer Networks*, 34(4), pp. 571–577. Available at: https://doi.org/10.1016/S1389-1286(00)00138-9.

Marir, N. *et al.* (2018) 'Distributed Abnormal Behavior Detection Approach Based on Deep Belief Network and Ensemble SVM Using Spark', *IEEE Access*, 6, pp. 59657–59671. Available at: https://doi.org/10.1109/ACCESS.2018.2875045.

Mehmood, Y. *et al.* (2017) 'Internet-of-Things-Based Smart Cities: Recent Advances and Challenges', *IEEE Communications Magazine*, 55(9), pp. 16–24. Available at: https://doi.org/10.1109/MCOM.2017.1600514.

Mukkamala, S., Janoski, G. and Sung, A. (2002) 'Intrusion detection using neural networks and support vector machines', in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290). 2002 International Joint Conference on Neural Networks (IJCNN)*, Honolulu, HI, USA: IEEE, pp. 1702–1707. Available at: https://doi.org/10.1109/IJCNN.2002.1007774.

Neri, F. (2000) 'Comparing local search with respect to genetic evolution to detect intrusions in computer networks', in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*. *2000 Congress on Evolutionary Computation*, La Jolla, CA, USA: IEEE, pp. 238–243. Available at: https://doi.org/10.1109/CEC.2000.870301.

Papalexakis, E.E., Beutel, A. and Steenkiste, P. (2012) 'Network Anomaly Detection Using Co-clustering', in *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. *2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, Istanbul: IEEE, pp. 403–410. Available at: https://doi.org/10.1109/ASONAM.2012.72.

Rani, D. *et al.* (2023) 'Design of an Intrusion Detection Model for IoT-Enabled Smart Home', *IEEE Access*, pp. 1–1. Available at: https://doi.org/10.1109/ACCESS.2023.3276863.

Sazzadul Hoque, M. (2012) 'An Implementation of Intrusion Detection System Using Genetic Algorithm', *International Journal of Network Security & Its Applications*, 4(2), pp. 109–120. Available at: https://doi.org/10.5121/ijnsa.2012.4208.

Shone, N. *et al.* (2018) 'A Deep Learning Approach to Network Intrusion Detection', *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1), pp. 41–50. Available at: https://doi.org/10.1109/TETCI.2017.2772792.

Tama, B.A. and Rhee, K.-H. (2017) 'Performance evaluation of intrusion detection system using classifier ensembles'.

Tao, P., Sun, Zhe and Sun, Zhixin (2018) ‘An Improved Intrusion Detection Algorithm Based on GA and SVM’, *IEEE Access*, 6, pp. 13624–13631. Available at: https://doi.org/10.1109/ACCESS.2018.2810198.

Ts, P. and Shrinivasacharya, P. (2021) ‘Evaluating neural networks using Bi-Directional LSTM for network IDS (intrusion detection systems) in cyber security’, *Global Transitions Proceedings*, 2(2), pp. 448–454. Available at: https://doi.org/10.1016/j.gltp.2021.08.017.

Uddin, M. *et al.* (2013) ‘Signature-based Multi-Layer Distributed Intrusion Detection System using Mobile Agents’.

Varal, A.S. and Wagh, S.K. (2018) ‘Misuse and Anomaly Intrusion Detection System using Ensemble Learning Model’, in *2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)*. *2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)*, Bhubaneswar, India: IEEE, pp. 1722–1727. Available at: https://doi.org/10.1109/ICRIEECE44171.2018.9009147.

Vasilomanolakis, E. *et al.* (2015) ‘Taxonomy and Survey of Collaborative Intrusion Detection’, *ACM Computing Surveys*, 47(4), pp. 1–33. Available at: https://doi.org/10.1145/2716260.

Verwoerd, T. and Hunt, R. (2002) ‘Intrusion detection techniques and approaches’, *Computer Communications*, 25(15), pp. 1356–1365. Available at: https://doi.org/10.1016/S0140-3664(02)00037-3.

Waskle, S., Parashar, L. and Singh, U. (2020) ‘Intrusion Detection System Using PCA with Random Forest Approach’, in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*. *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, Coimbatore, India: IEEE, pp. 803–808. Available at: https://doi.org/10.1109/ICESC48915.2020.9155656.

Wazirali, R. (2020) ‘An Improved Intrusion Detection System Based on KNN Hyperparameter Tuning and Cross-Validation’, *Arabian Journal for Science and Engineering*, 45(12), pp. 10859–10873. Available at: https://doi.org/10.1007/s13369-020-04907-7.

Wu, Y. *et al.* (2020) ‘Large-Scale and Robust Intrusion Detection Model Combining Improved Deep Belief Network With Feature-Weighted SVM’, *IEEE Access*, 8, pp. 98600–98611. Available at: https://doi.org/10.1109/ACCESS.2020.2994947.

Yin, C. *et al.* (2017) ‘A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks’, *IEEE Access*, 5, pp. 21954–21961. Available at: https://doi.org/10.1109/ACCESS.2017.2762418.

Zhang, Y., Lee, W. and Huang, Y.-A. (2003) ‘Intrusion Detection Techniques for Mobile Wireless Networks’.

Zhang, Y., Li, P. and Wang, X. (2019) ‘Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network’, *IEEE Access*, 7, pp. 31711–31722. Available at: https://doi.org/10.1109/ACCESS.2019.2903723.

Zhang, Z., Shen, H. and Sang, Y. (2007) ‘An Observation-Centric Analysis on the Modeling of Anomaly-based Intrusion Detection’.

Zhao, R. *et al.* (2016) ‘Deep Learning and Its Applications to Machine Health Monitoring: A Survey’. arXiv. Available at: http://arxiv.org/abs/1612.07640 (Accessed: 7 November 2023).