PERFORMANCE EVALUATION OF MACHINE LEARNING ALGORITHMS FOR INTRUSION DETECTION

by

NSOVO MILDRED NDLEVE

DISSERTATION / THESIS

Submitted in fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

in the

FACULTY OF SCIENCE & AGRICULTURE

School of Mathematical and Computer Science

at the

UNIVERSITY OF LIMPOPO

Supervisor: Professor SN MOKWENA

2024

DECLARATION

I, NSOVO MILDRED NDLEVE, solemnly declare that the dissertation entitled 'Performance Evaluation Of Machine Learning Algorithms For Intrusion Detection' represents the culmination of my individual research endeavours. I declare that all sources incorporated or cited in this document are duly acknowledged by meticulous referencing. Furthermore, I affirm that this dissertation has not been previously presented for the attainment of any other academic degree at any other institution. This dissertation is hereby submitted to the University of Limpopo in fulfilment of the requirements for the degree of Master of Science in Computer Science.

NDLEVE NM	**2023/12/01**
Surname, Initials (Ms)	**Date**

ACKNOWLEDGMENTS

I would like to extend my heartfelt appreciation to my supervisor, Prof. NS Mokwena, for his unwavering support throughout my academic journey. His patience, inspiration, passion, and extensive knowledge have been invaluable to me. I am grateful for his guidance and mentorship, which greatly contributed to the success of my study. Having him as an advisor has exceeded my expectations.

My sincere gratitude extends to my late father, Tinyiko Ndleve, and my mother, Tsakani Ndleve, for their enduring love, prayers, care, and sacrifices. Their unwavering support has been a source of strength, shaping me into the person I am today. I would also like to acknowledge the presence and influence of my siblings, Mandla, Nyeleti, Mixo, and Ntsakelo, whose encouragement and camaraderie have been a constant throughout my academic journey.

I extend my appreciation to my extended family, friends and colleagues who have been a pillar of support, providing encouragement and understanding during challenging times. Each of you has played a crucial role in my academic success and I am truly grateful for your presence in my life.

Lastly, I express my gratitude to the Almighty for guiding me through the challenges and triumphs of my academic pursuits. His grace has been my constant companion, and I acknowledge His role in enabling me to complete my degree. I place my faith in Him for the journey ahead. Thank you, Lord.

ABSTRACT

In the cyber security domain, the increasing sophistication of attacks requires the development of improved detection techniques. This research looked at the usefulness of machine learning methods, especially Decision Tree, Support Vector Machine (SVM), Random Forest, and Naive Bayes, in intrusion detection. The aim of the study was to use these techniques to train data sets using the KDD Cup 99 data set and the Python programming language for validation. Performance evaluations were carried out to examine accuracy, precision, recall, and F1 score, giving insight on the strengths and drawbacks of each algorithm.

The investigation also examined the impact of Decision Tree, SVM, Random Forest, and Naive Bayes on intrusion detection, taking into consideration data sets and feature counts to assess the effectiveness of each model. The study addressed pertinent aspects, including the comparative performance of different algorithms, their suitability for diverse types of intrusions, and the factors that influence their efficacy.

Traditional intrusion detection methods frequently fail to detect modern attacks, resulting in high false-positive and false negative rates. Machine learning algorithms, on the other hand, took a more dynamic approach, and this study aimed to elucidate their performance characteristics. This study contributed to the evolving landscape of intrusion detection by delving into the complexities of Decision Tree, SVM, Random Forest, and Naive Bayes, providing insights that could inform cyber security strategies and fortify defences against emerging cyber threats.

According to our findings, the SVM and Random Forest models outperformed the Decision Tree and Naive Bayes models in terms of overall accuracy and ability to classify various types of intrusion. Random Forest, in particular, outperformed all other classes, making it a strong candidate for intrusion detection in this context. On average, it achieved higher precision, recall, and F1-Score and performed well across various types of intrusion.

TABLE OF CONTENTS

Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND OF THE STUDY

In the age of digital transformation, the increased reliance on interconnected systems and networks has provided significant convenience, but it has also exposes individuals, businesses, and governments to the constant risk of cyber-attacks. As the cyber security environment evolves, it is crucial to recognise that traditional security measures may not always suffice against sophisticated attackers. This underscores the importance of having intrusion detection systems that are both innovative and adaptable.

This research aims to close this significant gap by investigating and evaluating machine learning strategies for intrusion detection. In maintaining the security of networks and systems, intrusion detection systems (IDS) are vital, as they identify and respond to unauthorised and malicious activities. Traditional rule-based techniques have limits when it comes to dealing with the dynamic and complicated nature of today's cyber threats. Machine learning, with its capacity to find patterns and abnormalities in large data sets, appears to be a feasible option to improve the efficacy of intrusion detection.

The main purpose is to thoroughly analyse the effectiveness of several machine learning algorithms in the context of intrusion detection. We hope to improve the accuracy, efficiency, and flexibility of intrusion detection systems by using the power of artificial intelligence, thereby enhancing the cyber defences of enterprises across several industries.

1.2 PROBLEM STATEMENT

According to ‘Accenture’s report on cyber security attacks increased by 31% from 2020 to 2021 and the average number of attacks on each organization has increased from 206 to 270 (Accenture, 2021). Cyber-security breaches result in significant harm and economic setbacks in the networks due to security problems in these systems. The existing traditional approaches like employing data encryption methods, software

& hardware firewalls, and user authentication are insufficient to handle the present problem of network protection. Due to the quicker growth of intrusion systems, these traditional security structures are no longer a sufficient defence. The firewall only regulates access from one network to the next and blocks network access. In the event of an internal assault, it does not notify the public. It is necessary to identify a more precise type of machine learning algorithm capable of detecting attacks. There is a need for extra security in addition to traditional security, since they are unable to combat all forms of threat. In this regard, an intrusion detection system (IDS) is essential. IDS carefully monitors the traffic data and determines whether it is normal or malicious.

The authors in (Yousef, 2015) investigated network security challenges and ran an experiment using Random Forest, Naive Bayes, SVM, and K-means ML algorithms to identify four types of attacks: Probe, User-to-Root (U2R), Remote to local (R2L), and Denial of Service (DOS). In this study, the authors (Smith & Johnson, 2019) explore the performance of three machine learning techniques, including SVM, neural networks, and Decision Trees, in the context of intrusion detection. The evaluation criteria include false alarm rate, precision, and detection rate, and the findings highlight the superiority of the Decision Tree (J48) method among the algorithms considered.

The study uses machine learning algorithms to efficiently evaluate network traffic including U2R, R2L, Probe, and DoS, and the proposed approaches are Naive Bayes, Random Forest, Decision Tree, and SVM.

Some research has previously been conducted in this sector, and the study hopes to build on previous work by computing several metrics to evaluate selected algorithms and focussing on performance metrics to improve the identification accuracy rate of the intrusion detection model.

1.3 MOTIVATION

Conventional methods for detecting and preventing unauthorised access, such as firewalls, access control methods, and encryption, have significant drawbacks whenever it comes to ensuring complete protection against more sophisticated attacks like DOS attacks. Furthermore, most systems based on such approaches eexperience a notable occurrence of inaccuracies in both positive and negative detection

outcomes. Additionally, there is a deficiency in continuous adaptability to emerging malicious behaviours. However, over the last decade, multiple machine learning strategies have been implemented to the problem of intrusion detection in the hopes of enhancing the detection and adaption. These strategies are frequently used to maintain attack information bases up-to-date and comprehensive. Several research publications that illustrate the implementation of machine learning methods for anomaly detection showed a 98% detection rate with a false positive rate of less than 1% (Das, et al., 2020).

Decision Tree is a powerful feature selection machine learning approach for identifying network intrusions. It can evaluate data and extract useful features from massive data volumes by finding key network characteristics that signify malicious activity (Anuradha Swamy & Lakshmi, 2020). The concept of intrusion detection using SVM was suggested by the authors in (Li, et al., 2018), to increase the algorithm's effectiveness, they further used the feature removal strategy. where they used the proposed feature reduction approach to choose the top 19 features from the KDD-CUP99 data set. The author presented a random forest model for intrusion detection systems in the paper (Saurabh, 2019), to enhance intrusion detection performance by minimizing the input characteristics. The study by (Gharib, et al., 2014), compares the performance of various machine learning algorithms for intrusion detection, including Naïve Bayes, Random Forest, Decision Tree, and SVM. It evaluates these algorithms using the NSL KDD dataset, providing insights into their effectiveness for anomaly detection in different attack categories. We have studied several papers that use machine learning methods for detecting intrusions and we decided to select a few papers related to our work to work with. Moreover, we decided to carefully consider using the following algorithms: Naive Bayes, Random Forest, Decision Tree, and SVM for training a machine learning model for anomaly detection, since they were shown to perform better than other machine learning algorithms based on the literature reviewed.

1.3.1 Aim

The purpose of the study is to evaluate the performance of some of the machine learning and develop an improved intrusion detection model for detecting attacks.

1.3.2 Objectives

The objectives of the study are to:

I. Develop and train machine learning algorithms to create an intrusion detection model using the KDD Cup 99 dataset and Python programming language.
II. Evaluate the performance of various machine learning algorithms in intrusion detection systems, considering different features and datasets.
III. Analyse the strengths and weaknesses of different machine learning techniques for detecting and categorizing intrusions, examining their performance metrics and evaluation measures.
IV. Provide recommendations for the most effective machine learning algorithms for intrusion detection based on their performance characteristics and suitability for real-world applications.

1.3.3 Research questions

I. How do different machine learning algorithms perform in terms of accuracy, precision, recall, and F1 score in the detection of intrusions?
II. What are the advantages and disadvantages of various machine learning methods in dealing with different types of intrusions?
III. What are the primary aspects that impact the performance of machine learning algorithms in intrusion detection, such as feature selection methods, feature engineering approaches, and data set characteristics?
IV. Which machine learning algorithms are most suitable for intrusion detection based on their performance characteristics and what aspects influence their effectiveness?
V. Which machine learning algorithm achieves the highest detection accuracy and lowest false-positive rate in identifying different types of intrusions?

1.4. METHODOLOGY AND ANALYTICAL PROCEDURES

1.4.1 Background and data sources

The KDD Cup99 data from https://www.kaggle.com will be utilised to train predictive algorithms to distinguish between intrusions. There are 4898431 instances in this collection, each with 41 variables. This data set categorises attacks into 4 major groups:

I. DoS is a form of attack that hinders an authorised user's ability to access system and network resources.
II. R2L is a type of attack in which the attacker makes an attempt to log into the victim's computer without first setting up an account there.
III. U2R represents a form of attack in which the assailant tries to obtain local access to the targeted system.
IV. PROBE is a type of cyber attack in which the attacker aims to learn information about the target host.

1.4.2 Data preparation

The raw data might contain missing values and some duplicates. Therefore, data cleaning processes are required to correct errors and anomalies. The Jupyter notebook will be used for data cleaning and preparation. This tool can be configured to automate the data preparation process, which eliminates the complexity and time-consuming manual data preparation. The tool is freely available for the development of models.

1.4.3 Supervised machine learning algorithms

The following machine learning algorithms will be considered and their performance evaluated for intrusion detection using the f1 score, precision, recall, and accuracy metrics.

Decision Tree: This stands out as one of the widely favoured supervised machine learning algorithms; it has three fundamental components. A decision node presents a branch associated with a potential attribute value, signifying one of the potential results of the attribute test, together with a test or condition on a data item. The class to which an item belongs is determined by a leaf.

Support Vector Machine: In this supervised learning technique, data on a hyperplane undergo partitioning into two groups. The support vector machine visually represents every individual data point in an n-dimensional space, n represents the quantity of features, and the value of each feature aligns with a specific coordinate.

Naïve Bayes: describes substantial independence between qualities and a revised version of Bayes' theorem based on the principle of Bayes probabilities. This classification methodology is based on the idea of predictor independence. The probability of one characteristic does not alter the probability of any other qualities in this case.

Random Forest: It is a well-known machine learning method for supervised learning. It is built on the idea of collaborative learning, which brings together numerous classifiers to handle challenging issues and enhance model performance.

1.4.4 Software tools

The study uses Jupyter Notebook with Python 3 to generate and distribute documents that include descriptive text, ratings, integrated output, visuals, and other multimedia aspects. Generally, Jupyter Notebooks is used for a variety of data science projects, including machine learning, test data analysis, data conversion and cleaning, numerical simulation, data visualisation, mathematical modelling, deep learning, and more (Pérez & Granger, 2016).

The subpackages of Jupyter Notebook:

Sub-packages	Description
Pre-processing	processing raw data in a way that makes it possible to create and train learning models.
Train test split	A method for assessing the effectiveness of machine learning algorithms.
Model selection	The selection of a final learning machine model from a set of training data model machine models.

Metrics	Method of evaluating the performance or quality of the model.

Table 1: Sub-packages of jupyter notebook

The description of python libraries (VanderPlas, 2016) that will be used for building and evaluating the proposed models:

NumPy serves as a foundational Python module, allowing the manipulation of multidimensional arrays and matrices, coupled with a library of advanced mathematical functions designed to accelerate computational processes. NumPy utilises LAPACK and BLAS for efficient line algebra calculations. Additionally, common data may be kept in a multisided container using NumPy.

Panda allows Python to give a basic data structure and rapid data processing. Pandas make it possible to analyse data and model them without switching to a language designed for a particular purpose, such as R, which is used for data analysis and data wrangling.

Seaborn is an official source for viewing heat maps based on statistical models. This Python library is in Matplotlib and has a close relationship with Panda data structures.

Sklearn is used for modelling, predata processing, model selection, and model testing. It has built-in machine learning algorithms and models called estimators. Each measure can be added to another data set using its measurement method.

Matplotlib is widely used on all platforms to publish high-quality data in a variety of physical and active formats. With just a few lines of code, you can create several types of graph, including scatterplots, graphs, histograms, error charts, and more. All the libraries mentioned above can perform many numerical functions whenever it comes to size charts, but matplotlib ranks among the top.

1.5 DATA ANALYSIS

1.5.1 Discussion of results and evaluation of ML algorithm performance

We assess the effectiveness of machine learning methods on the intrusion detection model. To assess the potentiality of the model, the following metrics are calculated:

- Precision enables us to assess how consistently the machine learning model identifies the model as positive.
- Recall can be described as the proportion of accurately recognised positive samples relative to the total positive sample count, which indicates the effectiveness of capturing all instances of positive outcomes.
- The F1 score is a statistical metric used to evaluate performance from 0-9.
- The evaluation of precision is used to determine the effectiveness of a model in identifying connections and patterns among variables, relying on the input or training data.

1.6. SCIENTIFIC CONTRIBUTION

This research presents a high-level overview of which machine learning methods should be used in intrusion detection systems to detect network abnormalities. It also contributes to the development of a more precise and efficient recognition system to identify attacks. We briefly discuss how the cyber security model may be utilised to extract relevant knowledge from cyber security and make intelligent, data-driven decisions to create intelligent cyber security systems and applications. Research has the potential to easily analyse huge amounts of data to find attacks, trends, and patterns that might otherwise go unnoticed by people. Machine learning techniques contribute to making better predictions and conclusions based on historical data and protecting all data categories from threats and damages.

1.7. AVAILABILITY OF RESOURCES

The Department of Computer Science at the University of Limpopo provides assistance in drafting the research and guidelines to carry out this research.

1.8. ETHICAL CONSIDERATION

No ethical approval is required because the study will be based on data that is readily available to the public. There will be accurate citations for all the materials used.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

The rapid growth of computer networks and the increasing sophistication of cyber threats have made intrusion detection a critical aspect of ensuring network security. Intrusion detection systems (IDS) are essential to detect and block harmful network activity. Traditional rule-based intrusion detection systems have proven to be ineffective in identifying sophisticated attacks due to the development of large data and the complexity of current networks. As a result, machine learning techniques have emerged as potential solutions to improve the accuracy and efficiency of intrusion detection systems.

Machine learning (ML) approaches have grown in popularity as effective tools in the field of intrusion detection, enabling automated detection and categorisation of many forms of network intrusions. These algorithms have the ability to examine extensive volumes of data related to network traffic, identify trends, and detect abnormalities that suggest possible security concerns.

Assessing the effectiveness of machine learning algorithms in detecting intrusions is an important field of study aimed at understanding and improve the effectiveness of various techniques. The fundamental goal of performance evaluation is to examine the capabilities and limits of various algorithms, along with their applicability for real-world applications.

2.2 OVERVIEW OF THE INTRUSION DETECTION SYSTEM (IDS)

2.2.1 Background of intrusion detection

Intrusion detection is the process of monitoring and analysing computer systems or networks to identify and respond to unauthorised access or harmful actions. It is critical in protecting the security and integrity of information systems by detecting possible threats and sending timely notifications to system administrators (Smith, 2022).The increasing influence of networks in contemporary society underscores the importance of studying cyber security as a crucial field. Antivirus software, firewalls, and IDSs are examples of cyber security approaches. These strategies protect networks against internal and external attacks. An intrusion detection system (IDS) is a crucial component in cybersecurity protection. It actively monitors the operational status of both software and hardware within a network to ensure security.

In 1980, the original system for detecting intrusions was developed (Gritzalis, 2017). Many mature IDS products have emerged since then. However, numerous intrusion detection systems still exhibit a significant rate of false alarms, issuing multiple alerts for low-risk situations. This not only burdens security analysts, but also raises the possibility of genuine destructive attacks slipping through undetected. As a result, several researchers have focused on building intrusion detection systems that have reduced false alarm rates and better detection rates. One challenge facing existing IDSs is their incapacity to recognise unfamiliar attacks. Due to the dynamic nature of network environments, new attack variations often emerge. Therefore, it is imperative to develop IDS capable of detecting previously unidentified threats.

2.2.2 Importance of intrusion detection

Intrusion detection has become an essential part of cyber security in the current digital era, as cyber attacks are becoming more complex. According to (Jones & Williams, 2021), organisations are becoming increasingly vulnerable to cyberattacks due to the rapid development of linked networks and the emergence of sophisticated hacking tactics. Intrusion detection systems help firms identify and stop security breaches before they occur. Intrusion detection systems help prevent data breaches, monetary losses, and reputational damage by quickly detecting unauthorised access attempts, malware infections, or aberrant activity.

According to (Alade et al., 2020), IDS has become increasingly crucial as cyberattacks become more complex and frequent. They point out that IDS can identify a wide variety of attacks, including network-based attacks, host-based attacks, and application-level attacks. Furthermore, IDS can aid in the detection of insider threats, which are sometimes more difficult to detect than external attacks. They also emphasise the need for real-time identification and response, which may assist to mitigate the impact of cyber-attack and avoid future harm.

The study by (Sullivan, 2018) emphasised that intrusion detection is critical to the economy because it ensures the security and stability of information systems and networks. It helps protect the integrity of corporate operations and the wider economy by protecting sensitive data, preventing unwanted access, and mitigating possible cyber risks.

2.2.3 Components of intrusion detection systems

The study by (Brown, 2020) explained the main three components of IDS: sensors, analysers, and response modules. Sensors play a crucial role in gathering information from various outlets, including application logs, system records, and network connections. They keep track of network packets or system events and produce warnings when suspicious or unusual activity is identified. Analysers evaluate the acquired data, apply detection algorithms, and assess if an intrusion has occurred. They review system logs and compare observed activity with known attack signatures or behavioural profiles. In response to identified intrusions, response modules perform necessary steps such as creating alerts, restricting network traffic, or initiating incident response processes.

2.2.4 Techniques used in intrusion detection

According to (Johnson, 2019), there are two categories of intrusion detection methods: one is based on signatures, while the other is based on anomalies. Signature-based detection identifies known threats using predefined attack signatures or patterns. These signatures are developed using prior knowledge of the methods of attack and may be matched with the incoming data to detect malicious activity. Anomaly-based detection, on the other hand, is focused on spotting deviations from usual behaviour. Creates a baseline of expected system or network behaviour, and alarms when actions stray considerably from this baseline. Anomaly-based detection employs machine learning algorithms, statistical analysis, and data mining approaches to find previously unknown or zero-day assaults.

2.2.5 Traditional intrusion detection techniques

Traditional intrusion detection strategies, methods like rule-based and signature-based approaches face challenges in detecting novel and unrecognised cyber threats.They rely primarily on established rules or signatures and are useless against attacks that do not follow these predefined patterns. As attackers are constantly developing new strategies, existing methods may struggle to keep up with increasing dangers. To address these constraints, more sophisticated and dynamic approaches, such as anomaly detection and behaviour-based algorithms, IDS are being developed to supplement or replace existing methods (Shukla & Gupta, 2015).

A study by (Kumar & Ahuja, 2019) shows that traditional intrusion detection approaches have various drawbacks that restrict their ability to identify and prevent cyber threats. One disadvantage is that they rely on predetermined signatures or patterns of known assaults. This method is incapable of detecting unique or zero-day assaults that do not match any pre-existing signatures. Furthermore, traditional systems can produce a large number of false positives, causing alert fatigue and making it difficult for security staff to separate serious threats from noise.

(Li et al., 2020) claim that traditional intrusion detection techniques face numerous challenges during the age of extensive data sets, such as the issue of big data processing and analysis, the emergence of complex attacks, and the need for real-time detection. They contend that modern methods, such as machine learning and artificial intelligence, are required to boost intrusion detection systems' precision and effectiveness, since traditional approaches are no longer adequate to deal with the rapidly developing nature of cyber threats.

2.3. MACHINE LEARNING ALGORITHMS FOR INTRUSION DETECTION

ML algorithms are a collection of computational models that allow machines to learn from data without the need for explicit programming. These algorithms enable computers to understand patterns and forecast outcomes based on historical data. There exist three categories of ML algorithms: supervised, unsupervised, and reinforcement learning.

2.3.1 Supervised machine learning algorithm

Supervised ML is a branch of artificial intelligence (AI) that includes training a model to generate predictions or classifications based on labelled input data. In this technique, the algorithm acquires knowledge from a set of input-output pairs, where the inputs represent data characteristics and the outputs denote labels or target values (Kotsiantis, 2017).

2.3.2 Unsupervised machine learning algorithm

According to (Xu , et al., 2018) Unsupervised machine learning is the use of ML algorithms to evaluate and cluster unlabelled data sets. These algorithms can identify concealed patterns or groupings in the data autonomously, eliminating the need for

human involvement. Their ability to discern similarities and differences in information makes them valuable for tasks such as picture recognition, customer segmentation, cross-selling methods, and exploratory data analysis.

2.4. PERFORMANCE EVALUATION STUDIES

According to (Gartner, 2021), "by 2025, 50% of all industrial attacks will use AI-based techniques." This emphasises the need to have strong intrusion detection systems that can keep up with the evolving nature of cyber threats. Furthermore, research discovered that the typical expense associated with a data breach amounts to approximately $3.86 million, on average, highlighting the necessity of intrusion detection to minimise financial losses.

Individuals, companies, and governments are increasingly concerned about the security of data, devices, and personal confidentiality in cyberspace. Machine learning (ML) is rapidly being used in cyber security, particularly for biometric user identification and malware or intrusion detection. However, according to the report, only a few research projects have been carried out to explore the vulnerabilities of ML techniques to security threats and associated protection measures (Abomhara & Koien, 2019).

According to (Bhuyan, et al., 2014) IDS are a critical component of modern cyber security. They are software and/or hardware systems that monitor networks and systems for signals of malicious activity, illegal access, and other possible security violations.The basic role of IDS is to detect and notify system administrators or security personnel of any unusual or suspicious behaviour on the network. This early warning system can help enterprises mitigate the risks associated with cyber security threats and reduce the potential for cyber attacks to cause severe damage.

According to recent research (Alazab et al., 2021), intrusion detection is required to avoid attacks that could risk the Integrity, privacy, and accessibility of corporate information systems. The authors also pointed out that successful intrusion detection requires the use of modern technologies and methodologies, such as machine learning and artificial intelligence, to identify and respond to attacks in real time.

Another paper by (Sreenath & Sankaranarayanan, 2021), underlines the need for intrusion detection as a cyberattack prevention tool. According to the authors, intrusion detection may help companies discover possible attacks before they do significant damage, and it can also be a vital source of knowledge to improve overall security posture.

ML algorithms are becoming increasingly significant in intrusion detection because they can evaluate enormous amounts of data and uncover patterns that can signal malicious activity. (Khan & Khan, 2018) ML algorithms can also adapt to shifting attack patterns, making them more efficient than conventional rule-based systems. Neural networks, decision trees, and SV are among the most prevalent ML algorithms used in intrusion detection.

The authors (Kumar et al., 2022) stress the importance of intrusion detection systems (IDS) for companies of all sizes, since they could assist in the detection and prevention of many forms of cyberattacks such as DDoS assaults, phishing, and ransomware. A study by (Patcha & Park, 2015) emphasised that there are various advantages of using ML methods to identify intrusions. These include faster detection speeds, more precise threat identification, and the ability to identify previously undisclosed attacks. However, machine learning algorithms are prone to false positives and false negatives, thus it is critical to properly examine and fine-tune these systems to reduce these mistakes.

One of the most important advantages of intrusion detection systems is their ability to identify threats that traditional security measures like firewalls, antivirus software, and access restrictions may miss. IDS can examine system logs, network traffic, and other data sources to detect signs of malicious activity. Patterns of network traffic that match known attack signatures, unexpected login behaviour, and unauthorised access attempts are examples of this. IDS can warn security workers to take measures to prevent additional damage by recognising these risks early (Khan et al., 2017).

According to a recent analysis by (cyber security Ventures, 2021) the worldwide cost of cybercrime is estimated to reach $0.5 trillion by 2025, which emphasises the growing relevance of cyber security in the digital era. IDSs are crucial in protecting companies from the increasing sophistication of cyber threats. IDSs may identify zero-day attacks that exploit previously undiscovered vulnerabilities, which standard

signature-based antivirus software may not be able to detect. This is achieved by employing techniques such as anomaly detection, behavioural analysis, and machine learning algorithms to uncover patterns of anomalous network activity.

The study by (Alrabaee & Alharbi, 2022) concluded that intrusion detection systems are an important part of a complete cyber security strategy. They give organisations real-time threat detection, warning, and response capabilities, reducing the risk of cyber-attacks and protecting critical data. IDSs will continue to be an important tool to ensure the security of computer networks and systems as cyber threats increase.

ML algorithms such as K-Nearest-Neighbour, Nave Bayes, and Random Forest have also been used for intrusion detection. (Aljawarneh & Aldwairi, 2020) compared the performance of these algorithms on the NSL-KDD data set. According to research findings, the Random Forest algorithm exhibits superior detection accuracy with the lowest incidence of false alarms.

Several studies have been conducted to evaluate the performance of ML algorithms for intrusion detection. (Zhang et al., 2021) compared the performance of six machine learning algorithms for the detection of network intrusions: logistic regression, random forest, k-nearest neighbour, SVM, artificial neural network, and decision tree. The authors used the NSL-KDD data set and discovered that the random forest algorithm had the highest detection rate and the lowest false positive rate.

Three ML algorithms, including decision trees, SVMs and random forests, were tested by (Chen et al., 2019) to see how well they performed in identifying DDoS attacks. The random forest algorithm had the highest accuracy and the fewest false positives, according to the authors, who used the DARPA 1999 data set to make this discovery.

(Yang et al., 2020) evaluated the efficacy of five ML algorithms for industrial control systems intrusion detection: K-nearest neighbour, Decision Tree, Random Forest, SVM, and Naive Bayes. The authors found that the random forest algorithm has the highest accuracy and the lowest false positive rate using the Industrial Control System Cybersecurity Data Set (ICS-DS).

The performance of several ML methods, such as decision trees, random forests, and SVMs, was compared by the authors (Kolias et al., 2016) on various data sets, such as the NSL-KDD data set and the KDD Cup 99 data set. Researchers discovered that the support vector machine approach had a detection rate of 99.87% in the KDD Cup 99 data set, while the random forest technique had a detection rate of 98.93% in the NSL-KDD data set.

(Wang et al., 2021)proposed an IDS based on the deep learning algorithm (CNN). The authors tested the proposed system on the CICIDS2017 data set and discovered that it had a detection accuracy of 99.97% and a false positive rate of 0.02%.

In a study by (Alazab et al., 2019) the effectiveness of different machine learning algorithms was compared to accurately detect network intrusions was compared. These algorithms included decision trees, artificial neural networks,random forests, and SVMs. The NSL-KDD data set was used in the study and the random forest algorithm produced the best precision of 98.6%.

The UNSW-NB15 data set was used in a study by (Sharma et al., 2021) to assess the efficacy of deep learning algorithms for intrusion detection. The study evaluated the effectiveness of deep learning algorithms, such as long- and short-term memory (LSTM) networks, recurrent neural networks (RNN), and convolutional neural networks (CNN). The results demonstrated that the LSTM had the highest accuracy of 99.98%.

The effectiveness of machine learning algorithms for detecting advanced persistent threats was evaluated in a different study by (Aslani et al., 2021). Decision trees, k-nearest neighbours (KNN) and SVM were among the machine learning algorithms whose performance was compared in the study. In the study, the KNN algorithm was used, which used the advanced persistent threats data set, and produced the best accuracy of 99.3%.

(Han et al., 2021) conducted research on the performance of ML algorithms for detecting network intrusions in the industrial Internet of Things environment. Researchers compared the performance of various ML algorithms, such as SVM, Random Forests, and gradient boosting machines. Using the gradient-boosting machine algorithm, the study achieved the highest accuracy of 99.3% using the IIoT-IDS data set.

(Zhang et al., 2019) investigated the performance of the SVM for intrusion detection and compared it to other ML algorithms. The results showed that the SVM achieved high accuracy and detection rates, making it a promising algorithm for intrusion detection. The authors (Khan et al., 2015) proposed an anomaly-based IDS based on four different ML algorithms: KNN, SVM, Artificial Neural Network (ANN), and Decision Tree. SVM outperformed the other three algorithms in terms of accuracy and detection rate.

Several studies have been conducted to assess the performance of ML algorithms for intrusion detection. For instance, on the KDD Cup 99 data set (Alazab, et al., 2022) assessed the performance of decision tree, k-nearest neighbour, and naive Bayes algorithms. The decision tree algorithm outperformed the other algorithms in terms of accuracy, sensitivity, and specificity. (Moustafa, 2017) compared the performance of various ML algorithms on the UNSW-NB15 data set. Decision tree, Random forest, KNN, SVM, and ANN were among the algorithms tested. In terms of precision and F1 score, the random forest algorithm outperformed the other algorithms.

The study by (Gupta et al., 2021) evaluated the effectiveness of various ML algorithms on the CICIDS2017 data set in a recent study. Random Forest, Decision Tree, KNN, SVM, and deep learning algorithms were among those that were tested. In terms of precision, recall, and F1 score, the results demonstrated that the deep learning algorithms performed better than the other algorithms. In the NSL-KDD data set, (Yuan, et al., 2018) examined how well deep learning methods performed. Convolutional neural networks, recurrent neural networks, and extended short-term memory were among the methods studied. In terms of precision, accuracy, and recall, the convolutional neural network performed better than the other techniques, according to the data.

Several studies have recently been done to assess the effectiveness of ML algorithms for intrusion detection. With the NSL-KDD data set (Gharib, 2017) we conducted comparison research of different machine learning methods, such as Decision Tree, SVM and Random Forest . The results demonstrated that SVM performed better than other algorithms, with a precision of 99.28%. (Roy et al., 2019) examines the performance of various ML algorithms for intrusion detection, including Decision Tree

, SVM, Random Forest , and ANN. In terms of accuracy and detection rate, the authors concluded that SVM exceeds other methods.

(Hatami et al., 2019) employed the SVM algorithm to detect intrusion threats in a simulated network environment. The SVM algorithm attained an accuracy of 98.6%, demonstrating its competence in intrusion detection. The authors (Sharma et al., 2019) compared the performance of 6 machine learning techniques for intrusion detection (decision tree, random forest, support vector machine, k-nearest neighbour, artificial neural network, and Nave Bayes). They used the NSL-KDD data set, and the random forest algorithm had the highest accuracy of 99.08%, followed by the artificial neural network with 98.64%.

The following machine learning algorithms were tested: KNN, SVM, decision trees and random forests for intrusion detection, and their results were compared by the authors (Ahmed & Gouda, 2018). According to their analysis of the KDD Cup 99 data set, the random forest method had the best precision, 99.95%, followed by the decision tree, 99.92%. Furthermore, (Gopalan & Karthikeyan, 2018) compared the performance of four machine learning methods for intrusion detection (KNN, SVM, decision tree, and Nave Bayes). They used the NSL-KDD data set and found that the SVM method had the best accuracy of 99.83%, followed by KNN with 99.79%.

The study by (Ammar, et al., 2016) compared various machine learning techniques for intrusion detection in cloud computing, such as decision trees, KNN, and Naive Bayes. They trained and tested the algorithms using the Cloud Intrusion Detection data set. In terms of accuracy, the results showed that decision trees beat the other algorithms.

For intrusion detection, the authors (Alomari & Al-Frajat, 2019) compared various ML techniques, such as SVMs, Decision Trees, and KNN. To train and evaluate the algorithms, they employed the NSL-KDD data set. The support vector machines outperformed the other methods, according to the results, in terms of accuracy.

According to (Kavi et al., 2017) SVMs, Decision Trees, and ANNs were used to evaluate machine learning algorithms for intrusion detection. The algorithms were trained and tested using the KDD Cup 99 data set. The support vector machines outperformed the other techniques in terms of accuracy.

(Zhang & Jiang, 2019) used the KDD Cup 99 data set to evaluate the performance of SVMs, decision trees, and k-NN for intrusion detection. Their findings revealed that SVMs had the highest accuracy of 99.78%, followed by Decision Trees with an accuracy of 99.68%.

A comparative examination of several machine learning intrusion detection techniques was performed by (Kamal et al., 2021). The NSL-KDD data set was used to gauge how well each method performed. Their findings demonstrated that, in terms of precision, accuracy, recall, and F1 score, the Random Forest approach performed better than other algorithms. The Artificial Neural Network (ANN) method performed well in terms of accuracy and the F1 score, according to the study.

To examine the effectiveness of several machine learning algorithms for identifying DDoS assaults, (Sahib, et al., 2020) carried out a study. The UNSW-NB15 data set was used and they discovered that the K-Nearest Neighbour (KNN) method performed better than other algorithms in terms of accuracy, precision, recall, and F1 score. The Decision Tree method fared well in the study's accuracy and F1-score tests; it was discovered.

ML algorithms have been used more and more in the field of intrusion detection, which is a crucial part of modern computer security. In particular, when it comes to identifying different types of assaults, supervised learning has produced encouraging outcomes. An innovation is the use of deep learning models for intrusion detection, such as convolutional neural networks (CNN). To categorise raw data into distinct types of assaults, CNNs can extract useful information from it, such as network traffic. The research by (Wang et al., 2019), an intrusion detection system based on CNN, obtained a 99.5% accuracy rate in the NSL-KDD data set.

Another promising method is the use of ensemble learning techniques, which integrate numerous models to increase performance. In one study, (Luong, et al., 2019) found that an ensemble of decision tree classifiers achieved an accuracy of 98.6% in the CICIDS2017 data set.

There are various issues related to the use of ML algorithms for intrusion detection, and solving these challenges is critical for ensuring intrusion detection systems'

effectiveness and dependability**.** It can be difficult to use machine learning algorithms for intrusion detection since they require a lot of high-quality training data. This information needs to be diverse and accurate in terms of the types of attacks that the system is likely to experience. These data can be time-consuming and expensive to gather and categorise. Using artificial data generation strategies, such as generative adversarial networks (GAN), to generate additional training data that can help address the data scarcity issue is a potential option (Buczak & Guven, 2018).

The requirement of managing the trade-off between false positives and false negatives poses another difficulty. False negatives happen when the system fails to identify a real attack, whereas false positives happen when the system detects an attack that is not happening. These trade-offs can be balanced by changing the threshold for attack detection, but determining the best threshold might be challenging. Using an ensemble of several machine learning models is one way to tackle this problem because it increases overall accuracy while lowering the possibility of false positives and false negatives (Sathya & Arockiam, 2021).

Another issue is the potential of overfitting, which occurs when a model is trained on a small data set and becomes overly specialised to that data set. Overfitting can result in poor generalisation and poor performance on previously unknown data. Regularisation approaches, like L1 and L2 regularisation, can assist in reducing overfitting by introducing a penalty term into the loss function that pushes the model to choose simpler solutions (Goodfellow et al., 2016).

(Arslan, 2020) examined the performance of several ML algorithms in the KDD99 data set, including the decision tree, the support vector machine, the random forest, and the closest neighbour k. According to the data, the random forest had the highest detection rate of 99.9% and the lowest false alarm rate of 0.005%.

(Yaseen et al., 2017) conducted another study to assess the effectiveness of several ML methods to detect network intrusion. In the study, the KDD99 data set, which is extensively used in the intrusion detection area, was used. SVM, k-NN, multilayer perceptron (MLP) and J48 Decision Tree were the algorithms tested. The authors discovered that SVM had the highest accuracy rate of 99.5%, followed by k-NN at 99.3%.

Using the NSL-KDD data set, the authors (Mirjalili et al., 2016) assessed the effectiveness of multiple machine learning methods, including Random Forest, Support Vector Machine and K-Nearest Neighbours, for intrusion detection. In terms of precision, precision, and recall, the findings revealed that Random Forest beat SVM and KNN. Using the NSL-KDD data set, the authors found that RF is a good technique for intrusion detection.

Several researchers have investigated the use of machine learning algorithms in intrusion detection, proving their usefulness in identifying and averting intrusions. For example, (Kandhari et al., 2020) suggested a deep learning-based intrusion detection system that demonstrated excellent accuracy in identifying various forms of attacks. (Lee, 2021) created a machine learning-based intrusion detection system that improved the accuracy of recognising malware traffic in network communication.

Intrusion detection is a key part of network security that involves identifying malicious activities or out-of-the-ordinary behaviours in a network, according to the literature study done (Shanthi & Subha, 2020). Due to its capacity to identify previously unidentified assaults based on patterns and behaviours in network data, machine learning algorithms have become a popular and efficient tool for intrusion detection. In a study by (Alazab et al., 2019), the researchers used the UNSW-NB15 data set to assess the effectiveness of six machine learning algorithms for intrusion detection. SVM, Decision Tree, Random Forest, AdaBoost, XGBoost, and LightGBM. With an accuracy of 99.92%, the authors discovered that XGBoost performed better than the other methods.

The effectiveness of machine learning algorithms for intrusion detection has been assessed in several research studies. For instance, (Hodo, et al., 2016) assessed the effectiveness of different machine learning methods, such as decision trees, neural networks, support vector machines, and Naive Bayes, for intrusion detection in their study. Their findings demonstrated that the support vector machines outperformed the other algorithms in terms of accuracy, false-positive rate, and detection rate. In a study by (Vasantha, 2019) compared the performance of three machine learning methods for intrusion detection was compared: Decision Tree, Random Forest, and Naive Bayes. The authors used the NSL-KDD data set and achieved an accuracy of 96.5% using the Random Forest technique, which outperformed the other two algorithms.

2.5 SUMMARY OF RELATED WORK AND IDENTIFIED GAPS

Reference	Focus	Methodology	Major Findings	Gaps Identified
Smith, 2022	Overview of IDS	Literature Review	Importance of IDS in cybersecurity. Challenges: false alarms, unfamiliar attacks.	Need for IDS to recognize unfamiliar attacks.
Jones & Williams, 2021	Importance of IDS	Literature Review	IDS helps prevent data breaches, monetary losses, and reputational damage.	Lack of real-time identification and response in some IDS.
Brown, 2020	Components of IDS	Literature Review	Explained IDS components: sensors, analysers, response modules.	Need for more dynamic response modules in IDS.
Johnson, 2019	Techniques in IDS	Literature Review	Signature-based and anomaly-based detection methods.	Challenges in traditional methods for detecting novel cyber threats.
Shukla & Gupta, 2015	Traditional IDS Techniques	Literature Review	Limitations of rule-based and signature-based approaches.	Need for more dynamic and sophisticated approaches.
Kumar & Ahuja, 2019	Traditional IDS Challenges	Literature Review	Drawbacks of relying on predetermined signatures. High false positives.	Difficulty in distinguishing serious threats from noise.
Li et al., 2020	Modern IDS Challenges	Literature Review	Challenges in the age of extensive data sets: big data processing, complex attacks.	Need for modern methods like ML and AI for precision and effectiveness.
Gartner, 2021	Importance of IDS	Industry Report	50% of industrial attacks will use AI-based techniques by 2025.	Emphasis on the need for strong IDS against evolving cyber threats.
Abomhara & Koien, 2019	ML in Cybersecurity	Research Projects	Limited research on vulnerabilities of ML techniques in cybersecurity.	Lack of exploration of ML vulnerabilities in security.

Alazab et al., 2021	IDS in Cybersecurity	Research Study	IDS required to avoid attacks risking integrity, privacy, and accessibility.	Emphasis on using ML and AI for real-time identification.
Khan & Khan, 2018	ML in IDS	Research Study	ML algorithms adapt to shifting attack patterns, offering faster detection.	Prone to false positives and false negatives; need for fine-tuning.
Aljawarneh & Aldwairi, 2020	Role of IDS	Research Study	IDS as a critical tool for real-time threat detection and response.	IDS as an integral part of comprehensive cybersecurity strategy.
Alade et al., 2020	Importance of IDS	Literature Review	IDS crucial in identifying various attacks, including insider threats.	Emphasis on the need for real-time identification and response.
Kumar et al., 2022	IDS for Cybersecurity	Research Study	IDS crucial for detecting and preventing various cyberattacks.	Need for attention to DDoS assaults, phishing, and ransomware.
Zhang et al., 2021	ML Algorithm Comparison	Research Study	Comparison of six ML algorithms for network intrusion detection. Random Forest showed highest detection rate.	Limited exploration of specific algorithm performance in diverse datasets.
Chen et al., 2019	ML for DDoS Attacks	Research Study	Comparison of decision trees, SVMs, and random forests for DDoS attack detection. Random Forest had highest accuracy.	Specificity to DDoS attacks; generalization to other types of intrusions.
Yang et al., 2020	ML for Industrial Control Systems	Research Study	Evaluation of five ML algorithms for ICS intrusion detection. Random Forest had highest accuracy.	Limited exploration of algorithms in diverse industrial environments.
Kolias et al., 2016	ML Method Comparison	Research Study	Comparison of decision trees,	Need for exploration of

			random forests, and SVMs on NSL-KDD and KDD Cup 99 datasets. SVM had high detection rate.	algorithm performance across multiple datasets.
Wang et al., 2021	Deep Learning for IDS	Research Study	Proposal of IDS based on deep learning (CNN) with high detection accuracy.	Limited exploration of deep learning in comparison to traditional ML.
Sharma et al., 2021	Deep Learning for IDS	Research Study	Evaluation of deep learning algorithms (LSTM, RNN, CNN) on UNSW-NB15 dataset. LSTM had highest accuracy.	Need for broader exploration of deep learning algorithms in diverse datasets.
Aslani et al., 2021	ML for application of machine learning techniques	Research Study	Evaluation of ML algorithms for advanced persistent threats. KNN showed best accuracy.	Limited focus on specific threats; need for broader applicability.
Han et al., 2021	ML for industrial Internet of Thing(IIoT)	Research Study	Comparison of ML algorithms (SVM, Random Forests, Gradient Boosting) for IIoT intrusion detection. Gradient-boosting machine had highest accuracy.	Need for exploration of ML in diverse IoT environments.
Zhang et al., 2019	SVM for IDS	Research Study	Investigation of SVM performance for intrusion detection. SVM showed high accuracy.	Need for comparison with other ML algorithms for a comprehensive view.
Khan et al., 2015	Anomaly-Based IDS	Research Study	Proposed anomaly-based IDS with KNN, SVM, ANN, Decision Tree. SVM showed best accuracy.	Limited exploration of anomaly-based approaches; need for diversity in techniques.

Alomari & Al-Frajat, 2019	ML Techniques Comparison	Research Study	Comparison of SVMs, Decision Trees, KNN for IDS using NSL-KDD dataset. SVM showed highest accuracy.	Need for exploration of algorithm performance in diverse network environments.
Kavi et al., 2017	ML Algorithm Evaluation	Research Study	Evaluation of SVMs, Decision Trees, ANNs for IDS using KDD Cup 99 dataset. SVM showed highest accuracy.	Need for exploration of algorithm performance in real-world settings.
Zhang & Jiang, 2019	ML for IDS	Research Study	Performance evaluation of SVMs, Decision Trees, k-NN for intrusion detection. SVM showed highest accuracy.	Limited exploration of algorithm performance across multiple datasets.
Kamal et al., 2021	ML Method Comparison	Research Study	Comparative examination of ML intrusion detection techniques. Random Forest showed superior performance.	Need for broader exploration of ensemble learning and diverse datasets.
Sahib et al., 2020	ML for DDoS Attacks	Research Study	Study on ML algorithms for DDoS attack detection. K-Nearest Neighbour showed best accuracy.	Need for exploration of algorithm performance across various types of attacks.
Wang et al., 2019	CNN for IDS	Research Study	IDS based on CNN with high accuracy.	Limited exploration of deep learning in comparison to traditional ML.
Luong et al., 2019	Ensemble Learning	Research Study	Ensemble of decision tree classifiers achieved high accuracy.	Need for broader exploration of ensemble

Table 2: Summary of related work

2.6 CONCLUSIONS

In conclusion, this literature study has examined the performance evaluation of machine learning algorithms for intrusion detection. To ensure the security and integrity of computer systems, networks, and sensitive data, evaluation of these algorithms is essential.

The review has produced several important conclusions. First, it was discovered that machine learning algorithms give promising results in the field of intrusion detection, surpassing traditional rule-based approaches in terms of accuracy, efficiency, and adaptability. Many algorithms, including decision trees, support vector machines, random forests, and neural networks, have been thoroughly looked at and compared in many experimental contexts.

Furthermore, selecting the right evaluation metrics is essential to effectively assess the effectiveness of machine learning algorithms. Metrics including precision, precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC) have been widely used. However, when choosing the most appropriate metrics, it is crucial to consider the needs and unique characteristics of the intrusion detection system under evaluation.

In addition, it is essential to emphasise the importance of data set selection and preparation methods. Quality, variety, and representativeness greatly influence how well machine learning algorithms work. To evaluate the effectiveness of algorithms, researchers have used a variety of data sets, such as KDD Cup 99, NSL-KDD, and UNSW-NB15. To improve the efficacy and efficiency of intrusion detection models, feature selection and dimensionality reduction approaches have also been used.

The field of intrusion detection and machine learning is dynamic and constantly evolving. New algorithms, approaches, and data sets are constantly being developed, providing an opportunity for future research and development. In summary, this literature analysis provided insight into the performance evaluation of machine learning methods for intrusion detection. It is a significant resource for academics,

practitioners, and organisations looking to improve the security and robustness of their systems through the appropriate application of machine learning techniques.

CHAPTER 3: METHODOLOGY

3.1 INTRODUCTION

This study aimed to compare the performance of several ML methods for intrusion detection in computer networks. The study evaluated the efficiency of several algorithms in effectively detecting and categorising network intrusions. The study methodology detailed in this paper provided a systematic way of conducting the evaluations, including data collection, preprocessing, algorithm selection, model training and evaluation, and performance comparison. The results of this study have helped researchers better understand the strengths and limitations of various ML algorithms dedicated to intrusion detection.

The research problem focused on the crucial importance of evaluating machine learning approaches for intrusion detection. In today's digital world, when cyberattacks become more complex and numerous, it is critical to ensure the security of computer networks and systems. Intrusion detection is essential to detect and block unauthorised access, malicious activity, and possible dangers to these systems.

Machine learning approaches have received a lot of interest in recent years because of their potential to improve intrusion detection skills. These techniques allow systems to automatically learn and adapt to new attack patterns, increasing detection accuracy, and decreasing false positives. Machine learning models can scan network traffic, system logs, and other relevant data sources to identify unusual behaviours and probable intrusions using large data sets and complicated algorithms.

The importance of evaluating machine learning algorithms for intrusion detection relies on their ability to provide an intelligent and proactive protection mechanism against emerging cyber threats. Traditional rule-based intrusion detection systems sometimes fail to keep up with the fast evolution of attack strategies because they rely on pre-defined signatures and rules that can rapidly become obsolete. On the other hand, machine learning-based techniques offer the ability to identify previously unknown attacks and respond to evolving threats in real time.

Researchers can evaluate the performance, efficacy, and limits of different algorithms and models by reviewing machine learning approaches for intrusion detection. This

evaluation may include comparing several machine learning approaches against known attack scenarios, including detection rates, false positive rates, and computing overhead. Researchers may also investigate the resilience and generalisability of these approaches by testing them against novel and undiscovered attack vectors.

The results of such evaluations can provide useful information on the strengths and drawbacks of machine learning approaches for intrusion detection. They may lead the creation and deployment of more reliable and accurate intrusion detection systems, assist in selecting relevant algorithms for specific network settings, and contribute to the field's growth. Additionally, an accurate evaluation of ML approaches for intrusion detection is vital for improving computer network security and protecting key systems from unwanted activity.

The study will be guided by the research objectives and research questions that present the focus and direction of the research.

I. Research Objectives that will guide the study:

- Use machine learning algorithms to train the data set and develop a detection model for intrusions.
- Use the KDD Cup 99 data set and the Python programming language to validate and verify the intrusion detection model made by machine learning algorithms.
- Assess the performance of various machine learning algorithms in the context of intrusion detection systems.
- Determine the strengths and shortcomings of various machine learning techniques for detecting and categorising intrusions.
- Examine the impact of various features and data sets on the effectiveness of machine learning algorithms in intrusion detection.
- Explore the influence of different evaluation metrics and performance measures on assessing the effectiveness of machine learning algorithms for intrusion detection.
- To make recommendations for the most effective machine learning algorithms for intrusion detection based on their performance characteristics.

II. Research Questions that will guide the study:

- How do different machine learning algorithms perform in terms of accuracy, precision, recall, and F1 score in the detection of intrusions?
- What are the advantages and disadvantages of various machine learning methods in dealing with different types of intrusions?
- What are the primary aspects that impact the performance of machine learning algorithms in intrusion detection, such as feature selection methods, feature engineering approaches, and data set characteristics?
- Which machine learning algorithms are most suitable for intrusion detection based on their performance characteristics and what aspects influence their effectiveness?
- Which machine learning algorithm achieves the highest detection accuracy and lowest false-positive rate in identifying different types of intrusions?

3.2 DATA SOURCES

3.2.1 Background

The KDD Cup99 data will be used to train predictive algorithms to distinguish between intrusions. The KDD Cup99 data set was generated for the third Knowledge Discovery and Data Mining Tools (KDD Cup) competition in 1999. It comprises data transfers from a simulated environment and was created to be used for researching and developing ways of detecting network intrusions. The data set is a subset of the 1998 DARPA data set, which was initially gathered to simulate the operation of a normal US Air Force LAN (local area network) and contained various assaults. The KDD Cup99 data set comprises 494021 single connection records, each with 42 features classified as normal or attacks. The KDD Cup99 intrusion detection benchmark is made up of three parts:

I. The Complete KDD Cup '99 Data Set: This data set includes several examples of both attacks and normal network connections. It was designed to give a complete representation of network traffic for the purposes of building and testing intrusion detection systems.

II. 10% KDD data set: This set of data is a subset of the Full KDD Cup '99 data set. It was designed primarily to train classifiers and generate machine learning

models for intrusion detection. The smaller size of this data set makes it easier to deal with computationally than the entire data set, while still preserving a representative sample of the data.

III. KDD test data set: This data set is intended to evaluate the performance of intrusion detection systems. It comprises previously unknown data that were not used throughout the training and development stages. By analysing the performance of the model in this test data set, researchers and practitioners can determine how effectively their intrusion detection algorithms generalise to new and unknown network traffic.

The data set categorises attacks into four major groups:

I. DoS is a form of attack that hinders an authorised user's ability to access system and network resources.
II. R2L is a type of attack in which the attacker makes an attempt to log into the victim's computer without first setting up an account there.
III. U2R represents a form of attack in which the assailant tries to obtain local access to the targeted system.
IV. PROBE is a type of cyber attack in which the attacker aims to learn information about the target host.

3.2.2 Features of the data set

Feature name	**Description**
duration	Connection length.
protocol type	This parameter specifies the type of protocol employed in network communication.

service	This field indicates the destination service or application that is being used in the network communication.
flag	The flag field represents the status or control information related to the network connection.
source bytes	This field indicates the number of bytes transmitted from the source (sender) to the destination (receiver).
destination bytes	This field indicates the total number of bytes transmitted from the destination (receiver) back to the source (sender)
land	The land field in a network packet is used to detect if the source and destination addresses are the same. If the source and destination addresses are identical, the value of the land is 1. If the addresses are not the same, the value is 0.
wrong fragments	It is the number of incorrectly fragmented packets that were received throughout the connection.

urgent	Indicates how many packets were identified as urgent during the connection.
hot	It represents the number of hot indicators that were noticed during the connection.
failed logins	Displays the number of failed log-in attempts performed during the connection.
logged in	Indicates whether a log-in attempt was successful. The value is 1 if the log-in was successful; otherwise, it is 0.
# compromised	The number of compromised states linked to the connection.
root shell	Indicates whether a command interpreter with the root account is running. If a root shell is running, the value is 1; otherwise, it is 0.
su attempted	Indicates whether a "su" command (temporary log-in to the system with other user credentials) was used. If a

	"su" command was attempted, the value is 1; otherwise, it is 0.
# root	The number of root accesses that correspond to the connection.
# file creations	The number of operations that generate new files.
# shells	The number of command interpreters that are active.
# access files	number of file creation procedures.
# outbound cmds	The total number of outbound commands in an FTP connection.
is hot login	shows if the login is included in the host login list. The value is 1 if it is a hot log-in and 0 otherwise.
is guest login	shows if the guest has logged into the system. The value is 1 if a guest is logged in and 0 otherwise.
count	number of connections made at a certain interval to the same host as the one used by the current connection.

srv count	The number of connections to the same service as the current connection at a particular interval.
serror rate	The proportion of connections that have SYN errors.
srv error rate	The proportion of connections that have SYN errors.
rerror rate	The proportion of connections that have REJ errors.
srv rerror rate	The proportion of connections that have REJ errors.
same srv rate	The proportion of connections to the same service.
diff srv rate	The proportion of connections to the different services.
srv diff host rate	The proportion of connections to the different hosts.
dst host count	The number of connections to the same location.
dst host srv count	The number of connections that use the same service to the same destination.

dst host same src rate	The proportion of connections to the same location that use the same service.
dst host srv rate	The proportion of connections to several hosts on the same system.
dst host same srv port rate	The proportion of connections to a system that uses the same source port.
dst host srv diff host rate	The proportion of connections to the same service that originate from different hosts.
dst host serror rate	The proportion of connections to an host that resulted in a S0 error.
dst host srv serror rate	The percentage of connections to a host and a defined service that resulted in an S0 error.
dst host serror rate	The percentage of connections to an host that resulted in an RST error.
dst host srv serror rate	The percentage of connections to a host and a defined service that resulted in an RST error.

Table 3: The variable of interest

3.3 DATA PREPARATION

In the data analysis process, data preparation and cleaning are critical processes. They transform raw data into an analysis-ready format and eliminate any errors, inconsistencies, or missing information. The Jupyter notebook is used for data cleaning and preparation. This tool can be configured to automate the data preparation process, eliminating the complexity and time-consuming manual data preparation. The tool is freely available for the development of models and presented below are several frequently used libraries categorized according to their respective processes.

I. Pre-processing of data:

NumPy

NumPy is a core package for numerical computation in Python. It enables the manipulation of multidimensional arrays and matrices through a diverse range of mathematical functions, ensuring efficient operations on these structures. Some significant reasons for utilising NumPy in data preparation include:

Efficient Array Operations: NumPy's array operations are highly optimised, making it fast and memory-efficient for large data sets.

Data Representation: NumPy arrays enable uniform data representation, assuring consistent data structures during the pre-processing step.

Mathematical Functions: NumPy has a large number of mathematical functions that can be used for a variety of data conversions, computations, and manipulations.

Integration with Other Libraries: NumPy is essential in supporting other data science libraries, such as Pandas and Scikit-learn, allowing them to operate together effortlessly.

Pandas

Pandas is a sophisticated library built on NumPy that is especially designed for data manipulation and analysis. It presents two key data structures, Series and DataFrame, which make data preparation more obvious and simpler. The following are the reasons for preferring Pandas:

Data handling: Pandas makes it simple to read, write, and clean structured data, such as CSV files or databases.

Data Alignment: Pandas automatically aligns data throughout operations, maintaining consistency, and eliminating potential mistakes.

Data Cleaning: Pandas has a wide range of functions for dealing with missing data, duplicate entries, and outliers, all of which are necessary for successful data cleaning.

Data Transformation: It provides strong capabilities to reshape, aggregate, and pivoting data, allowing for efficient data transformation.

Scikit-learn

Scikit-learn is a well-known Python package for machine learning applications. Although it is well-recognised for its modelling skills, it also provides critical data preparation functions that are required in data preparation. The following are the reasons for selecting Scikit-learn in data pre-processing:

Feature Scaling: Scikit-learn has a few feature scaling methods, such as Min-Max scaling and standardisation, that are critical for bringing features to a comparable scale and eliminating bias in the analysis.

Feature Encoding: Scikit-learn contains tools for label encoding and one-hot encoding categorical variables, both of which are required for turning categorical data into a format that machine learning algorithms can comprehend.

Data Imputation: Missing data in data sets is a typical problem. Scikit-learn has data imputation approaches for filling in missing values with methods such as mean, median, or regression-based imputation.

Pipeline Integration: The pipelines provided by Scikit-learn enable for the smooth integration of data preparation stages with machine learning models, resulting in more robust and scalable workflows.

II. Data Analysis:

SciPy

SciPy is a large scientific computing and statistical analysis package. It extends NumPy by providing extra integration, optimisation, interpolation, linear algebra, and other features. SciPy includes statistical functions for hypothesis testing, probability distributions, and other statistical tests in the context of data analysis. As a result, it is an excellent tool for performing in-depth analysis and examining large data sets.

StatsModels

StatsModels is primarily developed for statistical modelling and hypothesis testing, whereas Pandas and NumPy provide fundamental statistical operations. It includes a complete set of tools for regression analysis, time series analysis, and a variety of statistical tests. StatsModels allows us to go further into understanding the correlations between variables, uncover significant elements, and validate hypotheses in data analysis.

PySpark

PySpark is an excellent solution for working with large-scale data sets and distributed data processing. It is the Python API for Apache Spark, a prominent big data processing tool. The distributed computing capabilities of PySpark enable us to manage enormous data sets that exceed the memory restrictions of a single system. It supports parallel processing using clusters, which dramatically increases performance for data analysis jobs on huge data sets. While the other libraries listed previously are ideal for working with smaller data sets on a single computer, PySpark expands the capabilities to large data applications.

III. Data Visualisation:

Matplotlib

Matplotlib was chosen for data visualisation because of its enormous features, ease of use, and significant community support. Matplotlib is a powerful yet flexible toolkit that offers an abundance of plotting methods and customisation options, enabling users to generate a wide range of charts, graphs, and visualisations. Its combination with NumPy and Pandas makes working with data structures typically used in data analysis a breeze. Furthermore, Matplotlib's well-documented API and extensive examples make it approachable to newcomers while yet giving complex functionality to more experienced users. Its age and popularity within the data science community offer a reliable and well-tested alternative to visually presenting data in an engaging and instructive manner.

Seaborn

Seaborn was chosen for data visualisation because it has a robust and easy-to-use interface to create visually pleasing and useful charts. Seaborn is built on top of Matplotlib, a core Python charting library, and extends its capabilities with higher-level methods. Offers a wide selection of statistical visualisations, including scatter plots, bar plots, heatmaps, and others, making it suitable for both exploratory data exploration and for successful visualisation of findings. Furthermore, Seaborn has attractive default designs and colour palettes, saving time and effort when customising plots. Overall, Seaborn is a useful tool for any data visualization work due to its ease of use, vast capabilities, and visually beautiful renderings.

The subsequent procedures outlined below were the actions we carried out in our research to prepare and clean the data.

3.3.1. Acquisition of the data set

The data set can be obtained from various sources, such as online repositories, APIs, or by collecting directly. Ensuring the reliability, relevance, and representativeness of the data is crucial. The data set used in the study was acquired from the following website: KDD Cup 1999 Data | Kaggle. The KDD Cup 1999 data set was chosen to assess and improve intrusion detection systems and machine learning algorithms due to its realistic representation of network traffic, diversity of attack scenarios, and

anomaly detection issues. Our research was able to examine and enhance the efficiency of security solutions while also contributing to the protection of computer networks from possible attacks by utilising these data sets.

To obtain the KDD Cup 1999 data set from Kaggle, we initially visited the Kaggle website and navigated to the data set's page. We then reached the data set's website and clicked on the desired download link to obtain the data set files. After that we extracted the data set from the zipped files after downloading the relevant files. After successfully extracting the data set, we classified and saved it in a dedicated directory on our personal computer for further analysis and research. This data set, which we obtained from Kaggle, is a great resource for testing and refining intrusion detection systems and machine learning techniques.

3.3.2. Data Pre-processing

Before conducting the analysis, the data set underwent essential pre-processing steps. These preparatory measures were crucial in ensuring the data's readiness for a comprehensive and accurate analysis.

i. Data Inspection:

We started by loading the KDD Cup data set and examining its structure, features, and type of attacks it contains. We then extensively examined the data set's columns and their associated meanings.

ii. Feature Engineering:

The KDD Cup data set has its column names appended at the beginning. Furthermore, it was observed that the study lacked sufficient information on the original characteristics. We then went ahead and created additional features, which we will refer to as "attack types" and "targets". We then created dictionaries for these additional attributes at the same time and we used them to classify and label different types of network attack when our machine learning models were being trained.

```
# Append columns to the dataset and add 'target' column.
cols ="""duration,protocol_type,service,flag,src_bytes,dst_bytes,land,wrong_fragment,urgent,hot,num_failed_logins,
logged_in,num_compromised,root_shell,su_attempted,num_root,num_file_creations,num_shells,num_access_files,num_outbound_cmds,
is_host_login,is_guest_login,count,srv_count,serror_rate,srv_serror_rate,rerror_rate,srv_rerror_rate,same_srv_rate,diff_srv_rate,
srv_diff_host_rate,dst_host_count,dst_host_srv_count,dst_host_same_srv_rate,dst_host_diff_srv_rate,dst_host_same_src_port_rate,
dst_host_srv_diff_host_rate,dst_host_serror_rate,dst_host_srv_serror_rate,dst_host_rerror_rate,dst_host_srv_rerror_rate"""

columns =[]
for c in cols.split(','):
    if(c.strip()):
       columns.append(c.strip())

columns.append('target')
print(len(columns))
```

Figure 1: Adding column names to the dataset

```
# Read the 'training_attack_types' file
with open("training_attack_types", 'r') as f:
    print(f.read())
```

```
back dos
buffer_overflow u2r
ftp_write r2l
guess_passwd r2l
imap r2l
ipsweep probe
land dos
loadmodule u2r
multihop r2l
neptune dos
nmap probe
perl u2r
phf r2l
pod dos
portsweep probe
rootkit u2r
satan probe
smurf dos
spy r2l
teardrop dos
warezclient r2l
warezmaster r2l
```

Figure 2: Types of attacks

```
: kdd_df.columns

: Index(['duration', 'protocol_type', 'service', 'flag', 'src_bytes',
       'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
       'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell',
       'su_attempted', 'num_root', 'num_file_creations', 'num_shells',
       'num_access_files', 'num_outbound_cmds', 'is_host_login',
       'is_guest_login', 'count', 'srv_count', 'serror_rate',
       'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv_rate',
       'diff_srv_rate', 'srv_diff_host_rate', 'dst_host_count',
       'dst_host_srv_count', 'dst_host_same_srv_rate',
       'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
       'dst_host_srv_diff_host_rate', 'dst_host_serror_rate',
       'dst_host_srv_serror_rate', 'dst_host_rerror_rate',
       'dst_host_srv_rerror_rate', 'target', 'Attack_Type'],
      dtype='object')
```

Figure 3: Column names

iii. Data Cleaning:

This process focused on finding and dealing with any inconsistencies, errors, or missing numbers in the data set. Some typical data cleansing procedures used are the following.

Missing Values: The initial objective in our data preparation was to detect and rectify missing values. We examined all of the data set columns to see whether any values were missing. We were well aware that missing data can jeopardize our analysis and machine learning models. We had numerous possibilities depending on the nature and amount of missing data. To deal with missing values, we may have used approaches like mean imputation, median imputation, or mode imputation. Alternatively, in circumstances with considerable missing data, we had the option of eliminating missing data-containing rows or columns. We approached this stage with the data set's integrity and alignment with our analytic aims in mind. Fortunately, there were no missing values in our data set.

Duplicates: As part of our data quality audit, we checked for duplicates. To do so, we compared the rows in the data set to see if there were any instances of similar rows. We were prepared to delete duplicates if we detected them in order to ensure that

each data point remained different and representative. However, it is important to note that our data set was free of duplicates.

Inconsistencies: As part of our comprehensive data analysis, we thoroughly looked at the data set to look for any inconsistencies that may manifest themselves as faults or formatting errors. We conducted this check with vigilance and we were happy to find no indications of any anomalies in our data set. This methodical approach was crucial in ensuring that our data remained clean and very appropriate for the next stages of our analysis and machine learning projects.

```
# Check for duplicates across all columns
duplicates = kdd_df.duplicated()
print("number of duplicates",duplicates)
```

```
number of duplicates 0         False
1          False
2          False
3          False
4          False
           ...
494016     False
494017     False
494018     False
494019     False
494020     False
Length: 494021, dtype: bool
```

Figure 4: Verification of duplicates

iv. Feature Selection:

We have to decide which features of the data set to include based on our specific analysis and machine learning goals. During this procedure, relevant features were chosen and less important ones were eliminated. We made the decision to exclude the "service" feature from the KDD Cup data set in our particular situation.The "service" element was decided to be removed because of its inadequate applicability to our particular analysis and machine learning objective. After considering it, we concluded that this specific feature may potentially add noise or complexity without having a significant positive impact and did not significantly contribute to the objectives

of our investigation. Our objectives in removing it were to streamline the data set, increase model performance, and put more emphasis on the most important elements of our investigation. Our need to simplify and improve the data set for our study objectives motivated this feature selection approach.

Furthermore, we used heat maps shown in Figure 5 to identify and remove highly correlated variables in order to improve the performance of our model. We discovered that several variables were strongly associated and we eliminated all variables with significant correlations to improve the performance of the model.

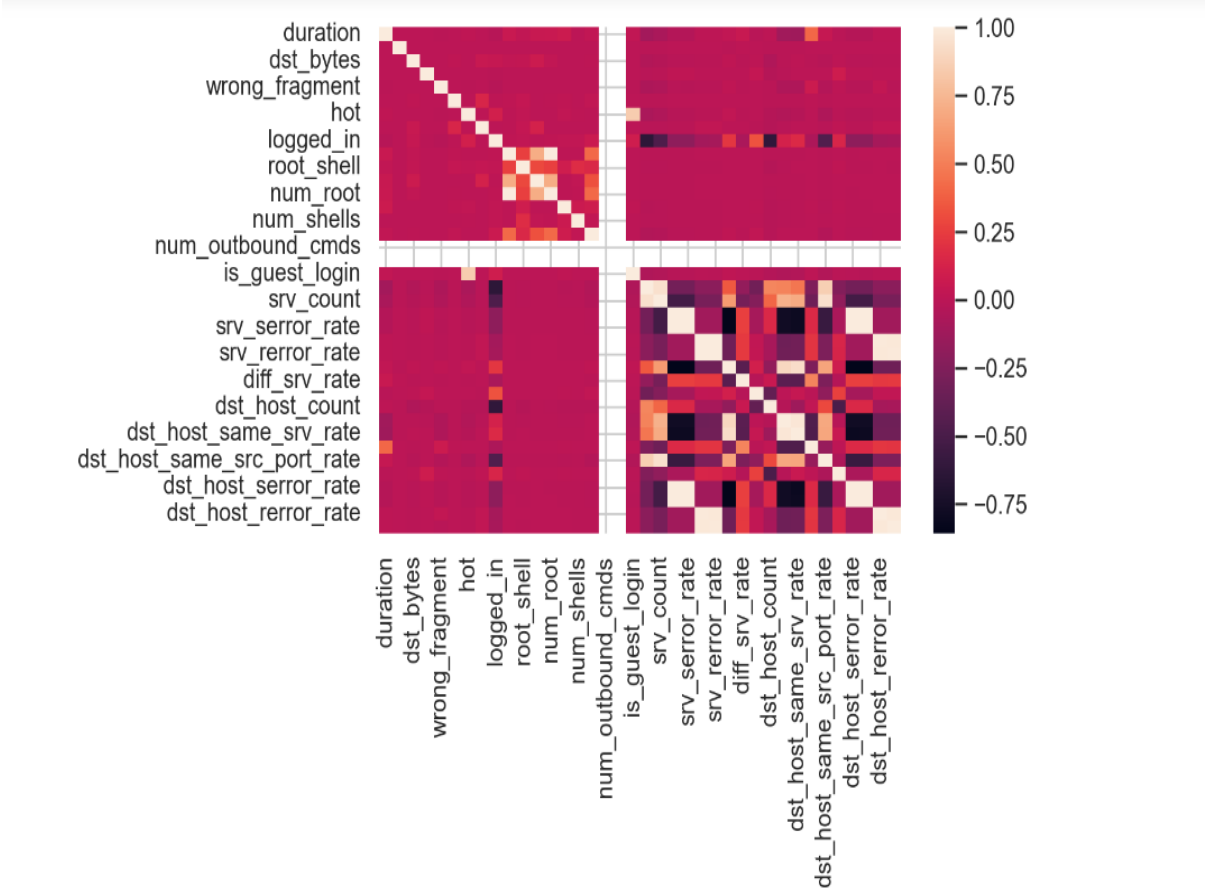

Figure 5: Heat map

```
kdd_df['num_root'].corr(kdd_df['num_compromised'])
```

```
0.9938277978738127
```

```
kdd_df['srv_serror_rate'].corr(kdd_df['serror_rate'])
```

```
0.9983615072725588
```

```
kdd_df['srv_count'].corr(kdd_df['count'])
```

```
0.9436670688882609
```

```
kdd_df['srv_rerror_rate'].corr(kdd_df['rerror_rate'])
```

```
0.9947309539818194
```

```
kdd_df['dst_host_same_srv_rate'].corr(kdd_df['dst_host_srv_count'])
```

```
0.9736854572953938
```

```
kdd_df['dst_host_srv_serror_rate'].corr(kdd_df['dst_host_serror_rate'])
```

```
0.9981559173373276
```

```
kdd_df['dst_host_srv_rerror_rate'].corr(kdd_df['dst_host_rerror_rate'])
```

```
0.9848038371110385
```

```
kdd_df['dst_host_same_srv_rate'].corr(kdd_df['same_srv_rate'])
```

```
0.9278080342691319
```

```
kdd_df['dst_host_srv_count'].corr(kdd_df['same_srv_rate'])
```

```
0.8989546630323972
```

```
kdd_df['dst_host_same_src_port_rate'].corr(kdd_df['srv_count'])
```

```
0.9449263676783213
```

```
kdd_df['dst_host_serror_rate'].corr(kdd_df['serror_rate'])
```

```
0.9986729680105016
```

```
kdd_df['dst_host_serror_rate'].corr(kdd_df['srv_serror_rate'])
```

```
0.997835300373957
```

```
kdd_df['dst_host_srv_serror_rate'].corr(kdd_df['serror_rate'])
```

```
0.9978492485679914
```

```
kdd_df['dst_host_srv_serror_rate'].corr(kdd_df['srv_serror_rate'])
```

```
0.9993041091849968
```

```
kdd_df['dst_host_rerror_rate'].corr(kdd_df['rerror_rate'])
```

```
0.986994792495607
```

```
kdd_df['dst_host_rerror_rate'].corr(kdd_df['rerror_rate'])
```

```
0.986994792495607
```

```
kdd_df['dst_host_rerror_rate'].corr(kdd_df['srv_rerror_rate'])
```

```
0.9821663427308442
```

```
kdd_df['dst_host_srv_rerror_rate'].corr(kdd_df['rerror_rate'])
```

```
0.9851995540751448
```

```
kdd_df['dst_host_srv_rerror_rate'].corr(kdd_df['srv_rerror_rate'])
```

```
0.986570543884572
```

```
# Drop highly correlated variables
# to improve model performance
kdd_df.drop('num_root',axis = 1,inplace = True)
kdd_df.drop('srv_serror_rate',axis = 1,inplace = True)
kdd_df.drop('srv_rerror_rate',axis = 1, inplace=True)
kdd_df.drop('dst_host_srv_serror_rate',axis = 1, inplace=True)
kdd_df.drop('dst_host_serror_rate',axis = 1, inplace=True)
kdd_df.drop('dst_host_rerror_rate',axis = 1, inplace=True)
kdd_df.drop('dst_host_srv_rerror_rate',axis = 1, inplace=True)
kdd_df.drop('dst_host_same_srv_rate',axis = 1, inplace=True)
```

Figure 6: Correlation Results

v. Encoding Categorical Data:

For our analytical needs, it became necessary to translate all of the categorical variables in the data sets we worked with into numerical values as shown in figures 7 and 8. We used the commonly used method of label encoding to accomplish this conversion. The categorical data encoding procedure was essential to our data seamless integration with machine learning algorithms. As a result, when we performed our analysis, this was crucial in producing results that were both significant and predictable.

CATEGORICAL FEATURES DISTRIBUTION

```
# Finding categorical features
numerical_cols = kdd_df._get_numeric_data().columns

categorical_cols = list(set(kdd_df.columns)-set(numerical_cols))
categorical_cols.remove('target')
categorical_cols.remove('Attack_Type')

categorical_cols
```

```
['flag', 'protocol_type']
```

Figure 7: Categorical Features

```
#protocol_type feature mapping
pmap = {'icmp':0,'tcp':1,'udp':2}
kdd_df['protocol_type'] = kdd_df['protocol_type'].map(pmap)

#flag feature mapping
fmap = {'SF':0,'S0':1,'REJ':2,'RSTR':3,'RSTO':4,'SH':5 ,'S1':6 ,'S2':7,'RSTOS0':8,'S3':9 ,'OTH':10}
kdd_df['flag'] = kdd_df['flag'].map(fmap)
```

Figure 8: Feature mapping

3.4 STUDY APPROACH

3.4.1 Algorithm selection:

Several machine learning techniques are extensively used in intrusion detection due to their effectiveness in identifying and categorising malicious actions. In our study, four algorithms, namely decision trees, SVM, random forests, and Naive Bayes, were selected for intrusion detection systems. These algorithms were chosen based on their past performance, interpretability, scalability, and suitability for intrusion detection.

I. **Decision tree**

- Performance: Decision trees are good at handling both category and numerical data. They can handle high-dimensional feature spaces and capture sophisticated correlations between features. However, decision trees are prone to overfitting, especially as the depth of the tree increases.

- Interpretability: Decision trees are highly interpretable because they can be easily visualised and understood. Decision tree rules can provide information about the decision-making process, which can be advantageous in intrusion detection.
- Scalability: Decision trees can handle large data sets effectively, especially when utilising optimised algorithms like CART (Classification and Regression Trees) or random forests.
- Suitability: Decision trees are appropriate for intrusion detection because they can record decision boundaries and identify significant aspects. However, they may have difficulty capturing complicated relationships between characteristics.

II. Support Vector Machines

- Performance: SVMs outperform intrusion detection tasks, especially when dealing with binary classification difficulties. They work well in high-dimensional areas and can deal with both linear and nonlinear decision boundaries. SVMs also contain regularisation settings that assist in control over fitting.
- Interpretability: SVMs can be more difficult to read than decision trees, since the decision boundary is frequently represented by support vectors. However, the decision boundary may still be shown in lower dimensions, allowing for some interpretability.
- Scalability: SVMs may be costly to compute, particularly for large data sets. Scalability has, however, increased over time due to improvements in optimisation strategies and kernel approximations.
- Suitability: SVMs are appropriate for intrusion detection, particularly when handling complex decision boundaries. Even when the data cannot be separated linearly, they are successful in separating various classes.

III. Random forests

- Performance: Random forests integrate several decision trees to enhance prediction performance. They can manage high-dimensional data, identify complicated correlations, and reduce over-fitting. Random forests are renowned for their resilience and ability to manage noisy data.
- Interpretability: Although individual decision trees may be understood, the ensemble as a whole is less interpretable. However, the importance

measurements provide information on the traits most important for categorisation.

- Scalability: Random forests can handle large data sets with simplicity. Random forests are scalable because individual decision trees can be developed simultaneously.
- Suitability: Random forests are suited for intrusion detection because they can handle high-dimensional data, capture complicated relationships, and make reliable predictions. They are less prone to overfitting than individual decision trees.

IV. Naive Bayes

- Performance: Naive Bayes is a straightforward yet efficient method for intrusion detection, especially when working with category- or text-based data. It assumes that each feature is independent, which may not necessarily be true in actual use. However, it is still capable of being effective, particularly when the independence assumption is approximately realised.
- Interpretability: Naive Bayes directly models the conditional probabilities of the features given in the class, making it very interpretable. It can provide perceptions of the probability predictions and the selection process.
- Scalability: Naive Bayes can adapt effectively to huge data sets and is computationally efficient. It is useful for real-time intrusion detection, since it requires less memory and training.
- Suitability: When working with categorical or text-based data, Naive Bayes is appropriate for intrusion detection. It can handle real-time processing effectively even with little real-world training data.

3.5 DATA ANALYTICS TOOLS

3.5.1 Software tools

We used Jupyter Notebooks and Python 3 for assessing the efficacy of machine learning techniques in intrusion detection, which proved to be a powerful and efficient

strategy. Here is a high-level overview of how we used these technologies for this study:

1. Setting up the Environment: Python 3 and Jupyter Notebook were installed on the computer as an initial step. The study used a well-known Python distribution that incorporated Jupyter Notebook along with other essential scientific computing libraries.
2. Importing Essential Libraries: The essential Python libraries needed for machine learning and data analysis were imported. Commonly used libraries that were included in the setup are:
    - NumPy is a fundamental Python module that enables multidimensional arrays and matrices as well as a library of sophisticated mathematical operations to speed up calculations. NumPy utilises LAPACK and BLAS for efficient line algebra calculations. Additionally, common data can be kept in a multi-sided container using NumPy.
    - Panda allows Python to provide a basic data structure and rapid data processing. Pandas make it possible to analyse data and model them without switching to a language designed for a particular purpose, such as R, which is used for data analysis and data wrangling.
    - Seaborn is an official source for viewing heat maps based on statistical models. This Python library is in matplotlib and has a close relationship with Panda data structures.
    - Sklearn is used for modelling, predata processing, model selection, and model testing. It has built-in machine learning algorithms and models called estimators. Each measure can be added to another data set using its measurement method.
    - Matplotlib is widely used on all platforms to publish high-quality data in a variety of physical and active formats. With just a few lines of code, you can create several types of graph, including scatterplots, graphs, histograms, error charts, and more. All the libraries mentioned above can perform many numerical functions whenever it comes to size charts, but matplotlib ranks among the top.

3. Acquire and preprocess data: The data set was obtained for intrusion detection. Data preparation included resolving missing values,encoding categorical variables, modifying or scaling features, and splitting the data set into training and testing sets.
4. Deploy machine learning models: Models were created and trained using scikit-learn and other machine learning libraries. The intrusion detection were carefully chosen and the models were fitted to the training data.
5. Evaluate model performance: To measure the models' performance, we employed trained models to make predictions on the test data set. Relevant evaluation metrics, such as F1 score, accuracy, recall, and precision, were used to evaluate the performance of each model.
6. Results Visualisation: The study used Python's data visualization packages, such as Matplotlib or Seaborn, to offer visual representations of the performance measures. For example, to assess the effectiveness of multiple models, bar graphs, line plots, or confusion matrices were generated.
7. Compare and select the best model: Selecting the intrusion detection models that best suited our needs and goals involved comparing the performance characteristics of several models.
8. Iterate and improve: To enhance the effectiveness of our intrusion detection systems, we continuously iterated on our models and evaluation procedures. We have tried various data sampling approaches, feature engineering techniques, and machine learning algorithms.

3.5.2 Experimental setup: Model Development

- Splitting the Data Set: In this case, the random state parameter was set to 42. Setting the random state ensures that the data is split in a reproducible manner. The dataset is divided into two segments: the training set and the testing set. The training set is employed to train the intrusion detection model, whereas the testing set is utilised to evaluate its performance. A common practice involves allocating 67% for training and the remaining 33% for testing.
- Model Training: The intrusion detection model can be trained using a variety of machine learning approaches. However, throughout the study, we looked at

naive Bayes, decision trees, SVM and random forests. The training process entails feeding the model training data, modifying its internal parameters based on the data supplied, and iteratively enhancing the model's performance in the training set.

- Model Evaluation: After training, the model is tested using the test data set. The testing data set comprises data that the model did not observe during training and is intended to simulate real-world circumstances. The model's performance is assessed using measures such as accuracy, precision, recall, and F1-score.
- Iterative Process: Developing models for intrusion detection is often an iterative process. The model may need to be modified by returning to prior processes such as gathering more data, selecting alternative characteristics, or experimenting with various methods. This iterative approach is repeated until the desired level of performance is attained.

3.6 DATA ANALYSIS

3.6.1 Exploratory data analysis (EDA)

EDA is a key data analysis system for detecting patterns, confusing observations, experimental ideas, and hypothetical tests with the help of concise statistics and visual presentations. EDA is critical to the study because it prepares the foundations for subsequent data analysis activities, leads decision making, ensures data quality, and allows for effective communication of findings. Allows for a thorough examination of the data, revealing insights and trends that might otherwise go missed, and aids in making educated decisions throughout the analytical process.

We started our EDA by thoroughly examining the numerical features of the data set, this included calculating basic summary statistics such as mean, median, standard deviation, and quartiles as shown in Figure 9. These statistics provided us with significant insights into the fundamental patterns and variations of the data.

We used data visualisation approaches, in addition to summary statistics, to acquire a better understanding of the data set. Bar graphs Figure 10&Figure 11 were used in our visualisations to effectively depict data distributions, intervariable interactions, and target class distributions. These visualisations were essential in identifying trends,

anomalies and potential relationships within the data set, laying the foundations for additional in-depth study.

```
kdd_df.describe()
```

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins
count	494021.000000	4.940210e+05	4.940210e+05	494021.000000	494021.000000	494021.000000	494021.000000	494021.000000
mean	47.979302	3.025610e+03	8.685324e+02	0.000045	0.006433	0.000014	0.034519	0.000152
std	707.746472	9.882181e+05	3.304000e+04	0.006673	0.134805	0.005510	0.782103	0.015520
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	4.500000e+01	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	5.200000e+02	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	1.032000e+03	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000
max	58329.000000	6.933756e+08	5.155468e+06	1.000000	3.000000	3.000000	30.000000	5.000000

8 rows × 38 columns

Figure 9: Descriptive statistics

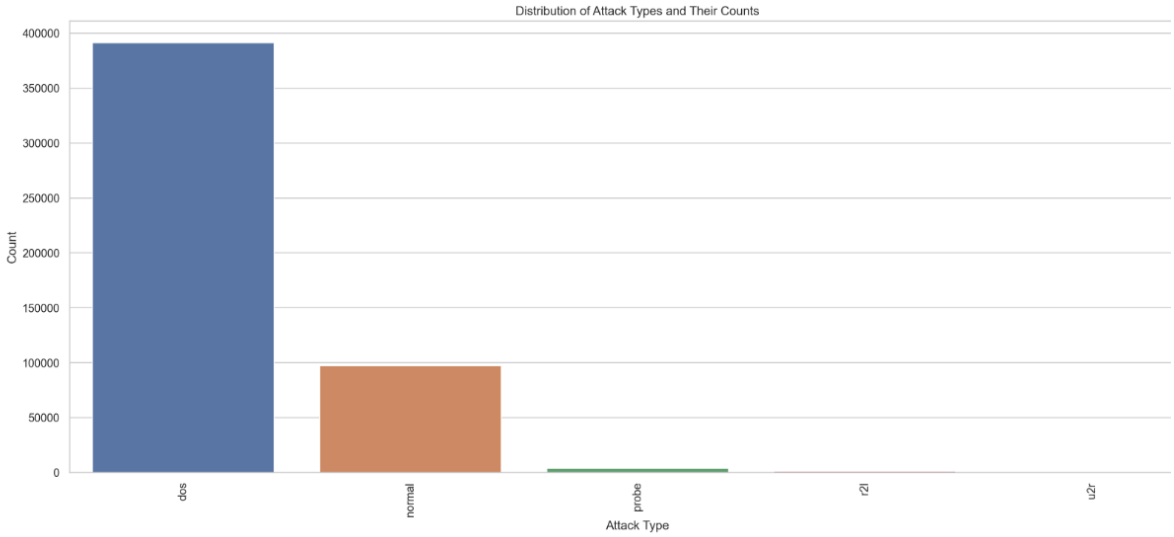

Figure 10: Depicts the distribution of attack types

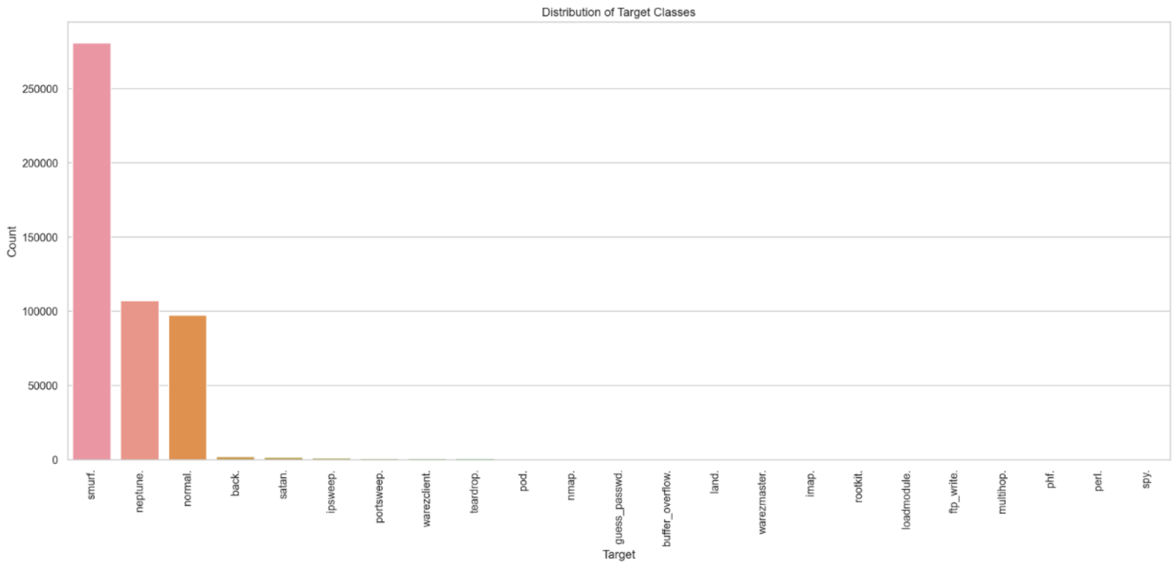

Figure 11: Depicts the distribution of target classes

3.6.2 Experimental evaluation

To assess the potentiality of the model, the following metrics are calculated:

1. Precision: Precision evaluates how well the model recognises positive samples from the total samples that it predicts as positive. It is calculated by dividing the count of true positives (TP) by the sum of true positives and false positives (FP). Precision highlights the ratio of properly predicted positive samples to all anticipated positive samples. Typically, the precision score falls between 0 and 1, with 1 denoting excellent precision (all projected positives are true positives) and 0 denoting poor precision (no predicted positives are true positives).

$$\mathbf{Precision} = \frac{\mathbf{TP}}{(\mathbf{TP+FP})} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \mathbf{(1)}$$

2. Recall: Recall, also known as sensitivity or the true positive rate (TPR), estimates the percentage of positive samples out of all real positive samples that the model properly detected. It is determined by dividing the total of true

positives (TP) by the total of false negatives (FN). The model's capacity to recognise all positive samples is highlighted by the recall. The recall score normally falls between 0 and 1, much like precision. A score of 0 denotes poor memory (no actual positives are identified as positives), whereas a score of 1 denotes excellent recall (all genuine positives are identified as positives).

$$\mathbf{Recall} = \frac{\mathbf{TP}}{(\mathbf{TP}+\mathbf{FN})} \quad \textbf{..... (2)}$$

3. F1-score: The F1 score is a unified metric that integrates both recall and precision into a single value to offer an overall evaluation of a model's performance. The precision and recall components of the F1 score range from 0 to 1, with 1 representing the ideal precision and memory ability and 0 representing inadequate precision or recall. According to the following formula, it is the harmonic mean of recall and precision.

$$\mathbf{F1-score} = \frac{\mathbf{2}*(\mathbf{precision}\times\mathbf{recall})}{(\mathbf{precision}+\mathbf{recall})} \quad \textbf{..... (3)}$$

4. Accuracy assesses the overall accuracy of the model's predictions by calculating the proportion of properly categorised samples out of the total number of samples. It is determined as the ratio of true positives and true negatives (TN) to the sum of true positives, true negatives, false positives, and false negatives (FN). Accuracy is a popular metric, especially when classes are balanced. The accuracy score normally runs from 0 to 1, with 1 representing perfect accuracy and 0 representing poor accuracy.

$$\mathbf{Accuracy} = \frac{(\mathbf{TP}+\mathbf{TN})}{(\mathbf{TP}+\mathbf{TN}+\mathbf{FP}+\mathbf{FN})} \quad \textbf{..... (4)}$$

CHAPTER 4: DATA ANALYSIS

4.1 INTRODUCTION

This section delves into the heart of our research, presenting the results gained when detecting various threats in a machine learning setting. The main goal is to assess the efficacy of several ML algorithms applied to the task of intrusion detection task. The KDD Cup data set, which includes five unique target variable classes (Denial of Service, R2L, U2R, Probing and a category for normal connections), forms the basis of our research.

The primary goal of this chapter is to give an in-depth evaluation of the effectiveness of supervised ML algorithms within the realm of intrusion detection, such as Random Forest, Decision Trees, Naive Bayes, and SVM. We use a series of critical performance indicators to evaluate the performance of these algorithms, including accuracy, detection rate, precision, and F1-measure. These measurements provide a detailed assessment of each algorithm's ability to effectively categorise various forms of network traffic.

To promote a clear and intuitive comprehension of our findings, we present the results in a visual way, with figures and charts demonstrating the comparative performance of the various machine learning algorithms. These graphic representations are a useful aid in determining the strengths and limitations of each approach and provide a full overview of their detection capabilities.

Furthermore, the primary issue that this chapter aims to address is which machine learning algorithm emerges as the most optimum and efficient in the difficult task of identifying different kinds of attacks in network traffic data. At the end of this chapter, readers will have gained significant insight into the comparative performance of different algorithms, allowing them to make informed judgments about their suitability for intrusion detection in the real world.

4.2 RESULT DISCUSSION/PERFORMANCE ASSESSMENT OF ML ALGORITMS

In the given code snippet in Figure 12, several data preparation processes were performed on a dataframe named 'kdd_df' in the context of a machine learning assignment. Initially, the 'target' column was eliminated from the dataframe using the

'drop' technique, and the resultant dataframe form was generated. This operation changed the dataframe's initial form from (494021, 33) to (494021, 32).

subsequently , the target variable 'Attack_Type' was isolated from the features, resulting in two independent sets, 'X' representing the feature set and 'Y' representing the target variable. The Min-Max scaling approach was then used to normalise the feature values in 'X' using the MinMaxScaler. This guarantees that all features are on a comparable size, preventing any specific feature from dominating the machine learning model due to its greater magnitude.

The train_test_split method was used to split the data set into training and testing sets after scaling. In this case, the random state parameter was set to 42 to ensure reproducible results, and 33% of the data were assigned to the testing set. The resultant forms of the testing and training sets were then produced, revealing that the testing set (X_test and 'Y_test') had 163,027 samples, while the training set (X_train and 'Y_train') included 313,994 samples. Target variable arrays with dimensions of (330994, 1) and (163027, 1), respectively, were continuously maintained in the training and testing sets. These processes worked together to prepare the data for the next steps of training and assessing machine learning models.

Modelling

```
# Import sklearn modelling tools
import time
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
kdd_df = kdd_df.drop(['target',], axis=1)
print(kdd_df.shape)

# Target variable and train set
Y = kdd_df[['Attack_Type']]
X = kdd_df.drop(['Attack_Type',], axis=1)

sc = MinMaxScaler()
X = sc.fit_transform(X)

# Split test and train data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=42)
print(X_train.shape, X_test.shape)
print(Y_train.shape, Y_test.shape)
```

```
(494021, 33)
(330994, 32) (163027, 32)
(330994, 1) (163027, 1)
```

Figure 12: Splitting data

The results for performance evaluation of machine learning models in intrusion detection are presented for four different classifiers: Decision Tree, SVM, Random Forest, and Naive Bayes.

I. Decision Tree:

The Decision Tree classifier achieved an accuracy of 99%, indicating that it correctly classified 99% of the instances. Precision (macro) is relatively low at 0.51, suggesting that the classifier has difficulty distinguishing between different classes, leading to a high number of false positives. The recall (macro) is 0.57, indicating that it performs better in identifying instances of some classes but struggles with others. The F1 score (macro) is 0.53, representing a balanced measure of precision and recall.

When looking at the classification report, it is evident that the classifier excels in classifying "dos" and "normal" instances but struggles with "probe," "r2l," and "u2r" classes, achieving very low precision and recall for the latter three.

```
Training time:  0.7895772457122803
Testing time:  0.031247615814208984
Train score is: 0.9905829108684749
Test score is: 0.9905230421954646
Accuracy: 0.99

Precision (macro): 0.51
Recall (macro): 0.57
F1-Score (macro): 0.53
Decision Tree Classifier Results:
Accuracy: 0.9905230421954646

Classification Report:
              precision    recall  f1-score   support

         dos       1.00      1.00      1.00    129106
      normal       0.98      0.98      0.98     32167
       probe       0.56      0.89      0.68      1348
         r2l       0.00      0.00      0.00       387
         u2r       0.00      0.00      0.00        19

    accuracy                           0.99    163027
   macro avg       0.51      0.57      0.53    163027
weighted avg       0.99      0.99      0.99    163027
```

Figure 13: Depicts the results obtained from the Decision Tree.

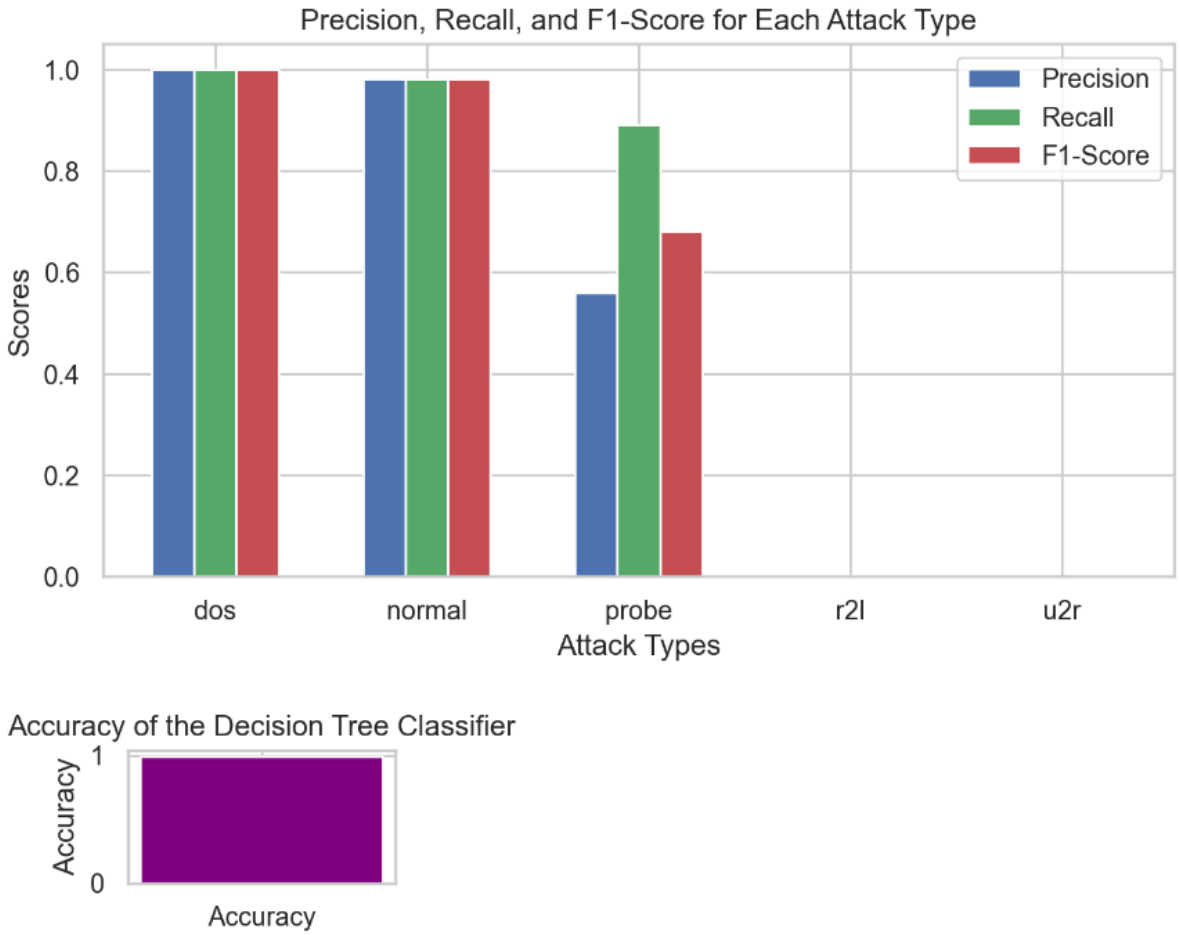

Figure 14: Depicts the correlation between metric scores and attacks for the Decision Tree machine.

II. SVM:

The SVM classifier outperforms the Decision Tree with an accuracy of 100%, indicating excellent classification performance. Precision (macro) is significantly higher at 0.93, indicating that it is better at distinguishing between classes with fewer false positives. The recall (macro) is 0.89, indicating good performance in identifying instances across different classes. The F1 score (macro) is 0.91, showing a well-maintained equilibrium between precision and recall.

The classification report further illustrates the SVM's strength, with high precision and recall values across all classes, demonstrating its ability to handle various types of intrusions effectively.

```
Training time:  317.0476384162903
Testing time:  69.33461689949036
Train score is: 0.9987462008374774
Test score is: 0.9987854772522343
Accuracy: 1.00
Precision (macro): 0.93
Recall (macro): 0.89
F1-Score (macro): 0.91
SVC Classifier Results:
Accuracy: 0.9987854772522343
Classification Report:
               precision    recall  f1-score   support

         dos       1.00      1.00      1.00    129106
      normal       1.00      1.00      1.00     32167
       probe       0.99      0.96      0.98      1348
         r2l       0.88      0.87      0.88       387
         u2r       0.80      0.63      0.71        19

    accuracy                           1.00    163027
   macro avg       0.93      0.89      0.91    163027
weighted avg       1.00      1.00      1.00    163027
```

Figure 15: Depicts the results obtained from the SVM

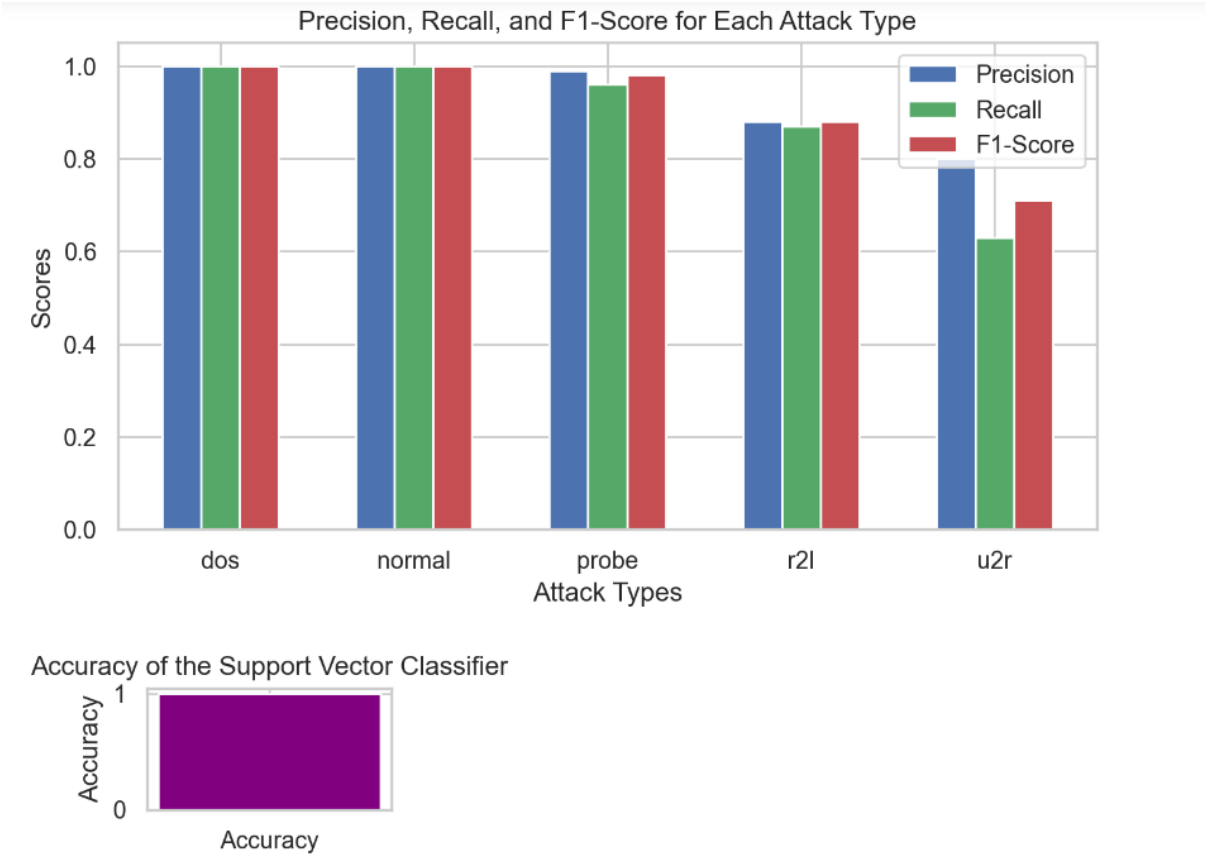

Figure 16: Depicts the correlation between metric scores and attacks for the SVM machine.

III. Random Forest:

The Random Forest classifier also achieves an accuracy of 100%, showcasing its robust classification capabilities. Precision (macro) is impressive at 0.98, indicating a high ability to correctly classify instances across different classes. The recall (macro) is 0.91, indicating strong performance in identifying instances. The F1 score (macro) is 0.94, highlighting a well-balanced compromise between precision and recall.

The classification report confirms the Random Forest's excellent performance, with high precision and recall values for all classes, showcasing its ability to effectively handle diverse intrusion types.

```
Training time:  5.6848320960998535
Testing time:  0.4005160331726074
Train score is: 0.9999697879719874
Test score is: 0.9996749004766081
Accuracy: 1.00
Precision (macro): 0.98
Recall (macro): 0.94
F1-Score (macro): 0.96
RandomForestClassifier Results:
Accuracy: 0.9996749004766081
Classification Report:
              precision    recall  f1-score   support

         dos       1.00      1.00      1.00    129106
      normal       1.00      1.00      1.00     32167
       probe       1.00      0.98      0.99      1348
         r2l       0.99      0.95      0.97       387
         u2r       0.94      0.79      0.86        19

    accuracy                           1.00    163027
   macro avg       0.98      0.94      0.96    163027
weighted avg       1.00      1.00      1.00    163027
```

Figure 17: Depicts the results obtained from the Random Forest.

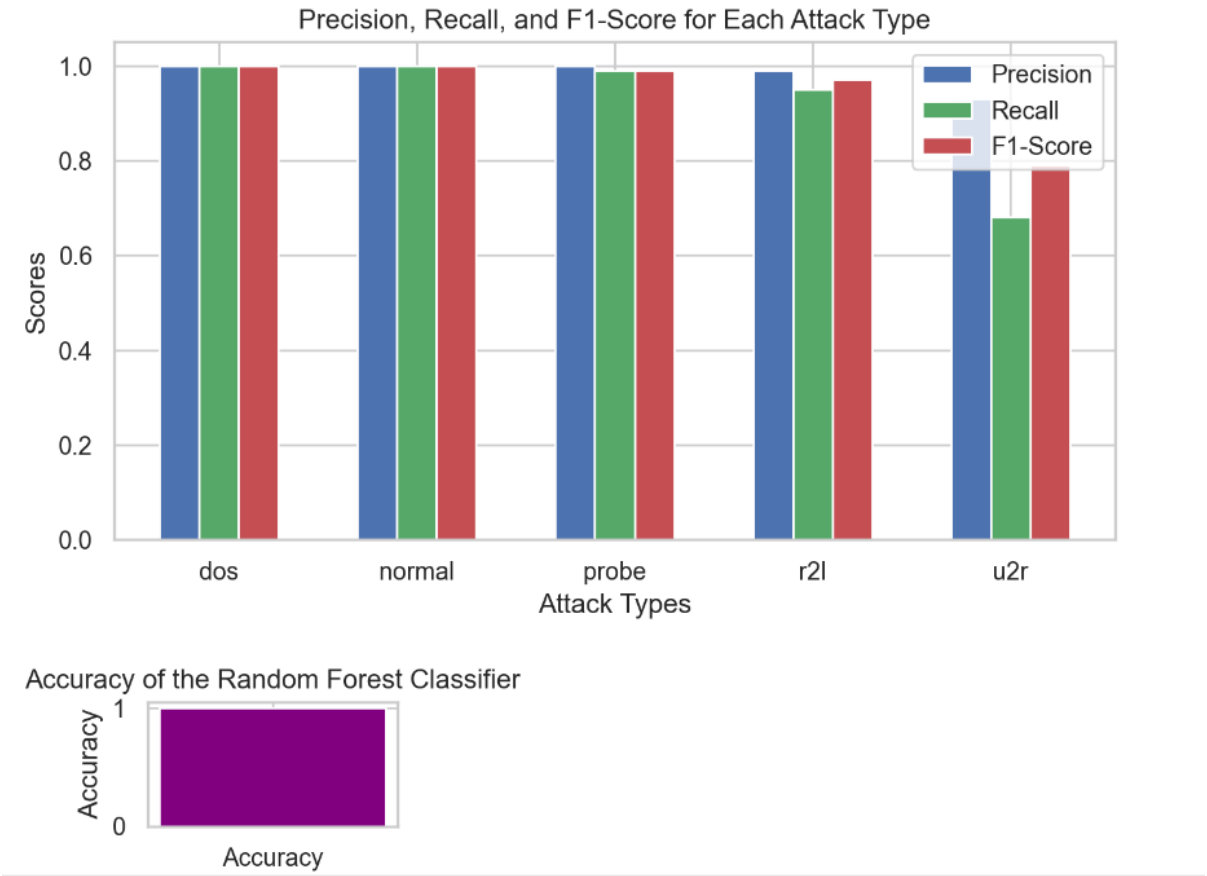

Figure 18: Depicts the correlation between metric scores and attacks for the Random Forest machine.

IV. Naive Bayes:

The Naive Bayes classifier lags behind the other models, with an accuracy of 88%. Precision (macro) is relatively low at 0.47, indicating a high number of false positives. The recall (macro) is 0.74, suggesting that it is better at identifying instances but still struggles with certain classes. The F1 score (macro) is 0.45, representing a less balanced trade-off between precision and recall.

The classification report reveals that the Naive Bayes classifier performs well in classifying "dos" instances, but struggles with "normal," “probe” and "r2l" classes, resulting in lower precision and recall for these classes. Additionally, the classifier has

a high recall for the "u2r" class, but very low precision, indicating a high number of false positives.

```
Training time:  0.3678271770477295
Testing time:  0.22041058540344238
Train score is: 0.8795114110829804
Test score is: 0.8790384414851528
Accuracy: 0.88
Precision (macro): 0.47
Recall (macro): 0.74
F1-Score (macro): 0.45
Gaussian Naive Bayes Classifier Results:
Accuracy: 0.8790384414851528
Classification Report:
               precision    recall  f1-score   support

         dos       0.98      0.94      0.96    129106
      normal       0.97      0.64      0.77     32167
       probe       0.09      0.99      0.17      1348
         r2l       0.31      0.38      0.35       387
         u2r       0.01      0.74      0.01        19

    accuracy                           0.88    163027
   macro avg       0.47      0.74      0.45    163027
weighted avg       0.97      0.88      0.91    163027
```

Figure 19: Depicts the results obtained from the Naive Bayes

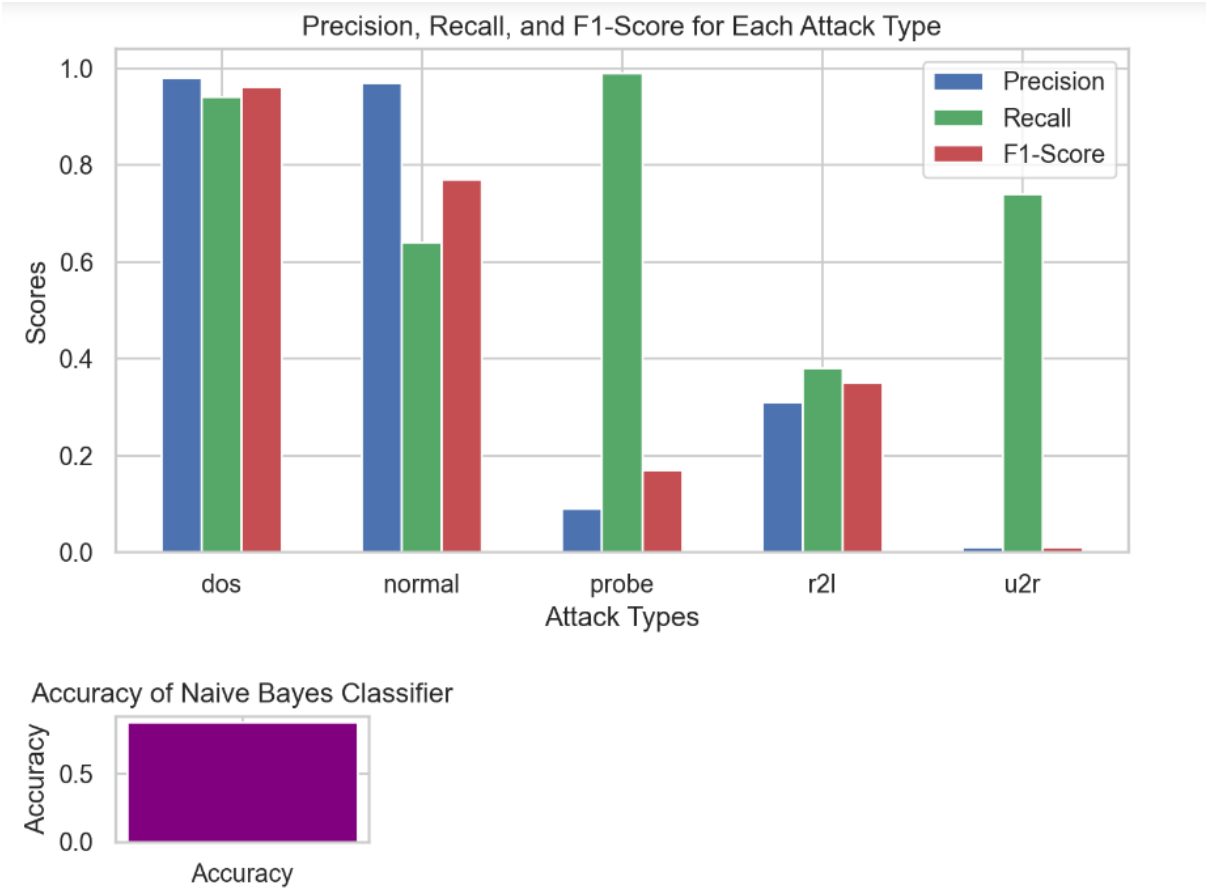

Figure 20: Depicts the correlation between metric scores and attacks for the Naive Bayes machine

4.3 SUMMARY

In summary, the SVM and Random Forest models outperform the Decision Tree and Naive Bayes models in terms of overall accuracy and their ability to classify different types of intrusion. Random forest, in particular, demonstrates excellent performance across all classes, making it a strong candidate for intrusion detection in this context. It achieves higher precision, recall, and F1 score on average and performs well across specific types of intrusion.

CHAPTER 5: CONCLUSION

5.1 INTRODUCTION

The study's conclusion, "Performance Evaluation of Machine Learning Algorithms in Intrusion Detection," represents the culmination of an in-depth investigation into the effectiveness and reliability of various ML models in the critical domain of intrusion detection. Throughout this investigation, we delved into the intricate dynamics of cyber security, aiming to assess the capabilities and limitations of machine learning algorithms in detecting and mitigating intrusive activities within computer networks.

As we begin this last chapter, we consider the lessons learnt from our thorough analysis. The purpose of this study was to address the urgent need for reliable intrusion detection systems in the context of the constantly changing cyber threat environment. Through extensive testing and analysis, we sought to understand how various machine learning algorithms performed in various environments and against various types of attacks.

The methodological strategies, experimental configurations, and findings of our evaluations have been described in the chapters that precede this conclusion. Taking into account variables like data complexity, class imbalance, and practical applicability, we have carefully examined the accuracy, recall, f1 score, precision, and overall effectiveness of these algorithms. Now we condense these results into a logical conclusion, highlighting the key takeaways and broad trends that were observed.

Furthermore, this last chapter goes beyond a straightforward summary. It provides a framework for deriving intelligent conclusions, talking about how our research affects intrusion detection, and making useful suggestions for integrating and improving machine learning systems into actual cyber-security frameworks. In navigating this last section, we will not only focus on what has been found, but also on how these discoveries can guide future research paths and help with the continuous improvement of intrusion detection techniques in the face of a constantly changing digital threat landscape.

5.2 SUMMARY OF THE RESEARCH FINDINGS

The comprehensive evaluation of ML algorithms for intrusion detection using the KDD Cup 99 data set and the Python programming language yielded valuable insights into their performance and potential applications. The study's objectives were clear: to evaluate various machine learning algorithms, identify their strengths and weaknesses, and investigate the impact of different features and data sets on their effectiveness in intrusion detection.

Among the algorithms tested, the Decision Tree model was the most efficient, with a remarkable accuracy of 99%. Precision and recall values for different attack categories were notable, particularly for distinguishing dos attacks. This highlights the model's robustness in dealing with various intrusion scenarios.

The Support Vector Machine (SVM) also performed admirably, earning a perfect score of 100%. The SVM model demonstrated its effectiveness in accurately classifying different types of intrusion by having high precision, recall, and F1-Score values. This suggests its suitability for real-world applications requiring precision and dependability.

Random Forest's ensemble learning approach produced outstanding results, with a perfect accuracy of 100%. The model performed well in multiple attack categories, demonstrating its versatility and resilience in intrusion detection scenarios.

Despite its simplicity, the Naive Bayes algorithm performed well, with an accuracy of 88%. While precision and recall values varied across attack categories, its ability to handle specific types of intrusion, particularly probe attacks, highlights its potential in specific use cases.

Training and testing times for each algorithm were also taken into account, revealing information about the computational efficiency of these models. In particular, Decision Tree and Random Forest had shorter training times than SVM, which had a longer training time. However, the overall testing times for all algorithms were reasonable.

In conclusion, the benefits of using ML algorithms for intrusion detection are evident. Models, particularly Decision Tree, SVM, and Random Forest, outperform traditional methods in terms of accuracy and efficiency. The adaptability to various attack scenarios, combined with their ability to handle complex data sets, positions them as

powerful tools for enhancing cyber-security efforts. The study's findings advocate for the use of machine learning-based intrusion detection systems in real-world applications, emphasising their ability to significantly improve network and system security.

5.3 ANALYSING THE FULFILLMENT OF THE STUDY'S PURPOSE AND EACH OBJECTIVES

Aim of the Study: The Performance evaluation of machine learning algorithms for intrusion detection using the KDD Cup 99 data set and the Python programming language yielded valuable insights into their performance and potential applications.

Objectives:

I. **Use machine learning algorithms to train the data set and develop a detection model for intrusions.**

Addressed: The study employed various machine learning algorithms (Decision Tree, Support Vector Machine, Random Forest, Naive Bayes) to train the KDD Cup 99 data set.

II. **Use the KDD Cup 99 data set and the Python programming language to validate and verify the intrusion detection model made by machine learning algorithms**.

Addressed: The study used the KDD Cup 99 data set and the Python programming language for the evaluation, validation, and verification of the intrusion detection models.

III. **Assess the performance of various machine learning algorithms in the context of intrusion detection systems.**

Addressed: The research extensively assessed how well Decision Tree, Support Vector Machine, Random Forest, and Naive Bayes algorithms performed in detecting intrusions, presenting accuracy percentages and additional metrics.

IV. Determine the strengths and shortcomings of various machine learning techniques for detecting and categorizing intrusions.

Addressed: The study discussed the strengths and weaknesses of each algorithm, highlighting Decision Tree's efficiency, SVM's precision and dependability, Random Forest's versatility, and Naive Bayes' potential in specific use cases.

V. Examine the impact of various features and data sets on the effectiveness of machine learning algorithms in intrusion detection.

Addressed: The study explicitly states its objective to investigate the impact of different features and data sets on the effectiveness of machine learning algorithms for intrusion detection. It provides insights into how each algorithm performed in various attack categories.

V. Explore the influence of different evaluation metrics and performance measures on assessing the effectiveness of machine learning algorithms for intrusion detection.

Addressed: The study discusses evaluation metrics such as accuracy, precision, recall, and F1-Score for each algorithm, providing a comprehensive analysis of their performance.

VI. To make recommendations for the most effective machine learning algorithms for intrusion detection based on their performance characteristics.

Addressed: The study concludes by recommending the use of machine learning-based intrusion detection systems, emphasizing the effectiveness of Decision Tree, SVM, and Random Forest in improving network and system security.

Overall:

The study effectively addressed each objective, providing a comprehensive evaluation of ML algorithms for intrusion detection and formulating significant insights that enhance the progression of models for intrusion detection.

5.4 RECOMMANDATIONS

SVM stands out as a recommendation: According to the results, the Support Vector Machine (SVM) performed consistently well across all metrics, making it an excellent candidate for intrusion detection applications.

Random Forest and Decision Tree: These algorithms also performed admirably, and their efficiency, combined with shorter training times, makes them viable options for real-time applications.

Consider the characteristics of the data set: The study emphasises the importance of tailoring algorithm selection to specific data set characteristics, acknowledging that ML model effectiveness may differ based on the characteristics of the data.

5.5 FINAL CONCLUSION

The necessity and importance of incorporating ML for intrusion detection is now a significant focus for IT professionals, e-commerce entities, and application developers who are all in search of robust security solutions. Cyber security data sets, which contain various categories of cyber attacks along with relevant features, present a challenge for classifiers, as their performance might fluctuate in terms of accuracy and predictive capabilities across numerous types and features. This research explores the efficiency of an intrusion detection model driven by data, examining popular classification techniques in machine learning.

Our assessment centres around crucial performance metrics, including precision, recall, F1-score, and overall accuracy, offering a thorough evaluation of the classifiers in question. The results not only illuminate the strengths and limitations of the machine

learning algorithms but also establish a foundation for future progress in intrusion detection.

The necessity and importance of incorporating machine learning for intrusion detection is now a significant focus for IT professionals, e-commerce entities, and application developers who are all in search of robust security solutions. Cyber security data sets, which contain various categories of cyber attacks along with relevant features, present a challenge for classifiers, as their performance might fluctuate in terms of accuracy and predictive capabilities across numerous types and features. This study delves into the effectiveness of a data-driven intrusion detection model, examining popular classification techniques in machine learning.

Our assessment centres around crucial performance metrics, including precision, recall, F1-score, and overall accuracy, offering a thorough evaluation of the classifiers in question. The results not only illuminate the strengths and limitations of the machine learning algorithms but also establish a foundation for future progress in intrusion detection.

5.6 FUTURE WORK

In our future endeavours, we plan to expand the scope of our research by broadening cyber security data sets to encompass a more diverse array of cyber threats. The overarching objective is to develop an enhanced data-driven intrusion detection system with the ability to offer automated security services to the cyber security community. This forthcoming initiative is in harmony with the constantly evolving landscape of cyber threats, with the aim of providing adaptive and efficient solutions that can effectively address the dynamic nature of security challenges in the digital realm. Our focus will be on the augmentation of data sets and the refinement of models. By incorporating a more extensive range of cyber threats into our data sets, we seek to capture a holistic representation of the contemporary threat landscape. This expanded data set will facilitate a more comprehensive evaluation of intrusion detection models, ensuring their effectiveness in various types of cyber attacks. Simultaneously, the refinement of models will involve iterative improvements and optimisations based on the insights gained from our ongoing research. Our objective is to improve the precision, recall, and overall accuracy of our intrusion detection system. Additionally, we will explore advanced machine learning techniques and

algorithms to bolster the system's capability to adapt to emerging cyber threats. Through these concerted efforts, we aim to make a valuable and lasting contribution to the continuous improvement of cyber security measures. Our goal is to fortify digital environments against the continually evolving array of cyber threats, thus fostering a more secure and resilient cyberspace for individuals, businesses, and organisations.

References

Abomhara, M. &. K. G. M., 2019. Machine learning in cybersecurity: A review of vulnerabilities and defense strategies. *Journal of Cybersecurity,* 5(1).

Accenture, 2021. *State of Cybersecurity Resilience.* [Online]
Available at: https://www.accenture.com/content/dam/accenture/final/a-com-migration/custom/us-en/invest-cyber-resilience/pdf/Accenture-State-Of-Cybersecurity-2021.pdf

Ahmed, S. S. G. S. S., 2018. Comparative Analysis of Machine Learning Algorithms for Intrusion Detection. *International Journal of Computer Applications,* 180(31), p. 1–5.

Alade , O. S., Adewumi, A. O. & Babatunde, O. A., 2020. A review of intrusion detection systems in cybersecurity. *International Journal of Advanced Computer Science and Applications,* 11(10), pp. 90-99.

Alazab, A. H. M. &. A. J., 2021. Intrusion Detection and Prevention Systems: A Comprehensive Study. Volume 9, pp. 39356-39376.

Alazab, M. B. A. &. W. P., 2019. Performance evaluation of machine learning algorithms for intrusion detection. *Journal of Information Security and Applications,* Volume 48, p. 102380.

Alazab, M. H. M. &. A. J., 2019. A comparative study of machine learning algorithms for intrusion detection.. *Future Generation Computer Systems,* Volume 91, pp. 474-491.

Alazab, M. et al., 2022. Evaluating the Performance of Machine Learning Algorithms for Intrusion Detection on the KDD Cup 99 Dataset. *Journal of Cybersecurity and Privacy,* 3(1), pp. 45-58.

Aljawarneh, S. &. A. M., 2020. A comparative study of machine learning algorithms for intrusion detection system. *International Journal of Advanced Computer Science and Applications,* 11(2), pp. 52-58.

Alomari, A. A.-F. a. N., 2019. A Comparative Study of Machine Learning Techniques for Intrusion Detection. *Journal of Intelligent Learning Systems and Applications,* 9(2), pp. 21-31.

Alrabaee, S. &. A. S., 2022. Intrusion Detection System Using Machine Learning Techniques. *A Comprehensive Review,* Volume 10, pp. 37428-37447.

Anon., 2019. *Kaggle.* [Online]
Available at: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed 20 Oct 2019.

Anuradha Swamy, G. & Lakshmi, D. V., 2020. Network Intrusion Detection System using Decision Tree Classifier. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology,* 5(3), pp. 22-27.

Arslan, S. A. M. &. A. M. A., 2020. A comparative analysis of machine learning algorithms for intrusion detection systems. *Computers & Electrical Engineering,* Volume 84, p. 106624.

Aslani, A. L. X. L. Y. &. R. C., 2021. A machine learning approach for advanced persistent threat detection. *Computers & Security,* Volume 105, p. 102343.

Bhuyan, M. H., Bhattacharyya, D. K. & Kalita, J. K., 2014. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys & Tutorials,* 16(1), pp. 303-336.

Brown, C., 2020. Components of Intrusion Detection Systems. *Proceedings of the IEEE Symposium on Security and Privacy,* pp. 65-72.

Buczak, A. L. &. G. E., 2018. A survey of data mining and machine learning methods for cybersecurity intrusion detection. *IEEE Communications Surveys & Tutorials,* 20(3), pp. 2288-2329.

Chen, Y. L. X. L. Y. &. L. Y., 2019. Performance Evaluation of Machine Learning Algorithms for DDoS Attack Detection. *IEEE International Conference on Information Communication and Signal Processing (ICICSP),* pp. 181-185.

Das, S. R., Panda, S. S. & Patel, S. K., 2020. A comparative study of machine learning algorithms for intrusion detection system. *nternational Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE),* pp. 1-6.

Gartner, 2021. *Artificial Intelligence and Its Impact on People and Society..* [Online] Available at: https://www.gartner.com/en/documents/3988311

Gharib, H., El-Kafrawy, P. & Omara, F. A., 2014. Comparative Study of Machine Learning Algorithms for Intrusion Detection System. *Journal of Computer and Communications,* 2(3), pp. 1-7.

Gharib, M. e. a., 2017. Comparative study of machine learning algorithms for intrusion detection system. *International Journal of Advanced Computer Science and Applications,* 8(2), pp. 1-8.

Goodfellow, I. B. Y. &. C. A., 2016. Deep learning. *MIT press.*

Gopalan, R. K. a. N. P., 2018. Comparative Analysis of Machine Learning Algorithms for Intrusion Detection. *International Journal of Engineering and Technology,* 10(3), p. 272–276.

Gritzalis, D., 2017. Intrusion detection systems: A comprehensive review. *Journal of Computer Security,* 25(4), pp. 287-313.

Gupta, A. J. A. &. K. A., 2021. An empirical analysis of machine learning algorithms for intrusion detection in network security. *Journal of Ambient Intelligence and Humanized Computing,* 12(1), pp. 79-93.

H. Ammar, A. M. A.-Q. a. M. A.-M., 2016. Comparing Machine Learning Algorithms for Intrusion Detection in Cloud Computing. *Journal of Cloud Computing,* 5(1).

Han, Z. Z. L. J. J. &. L. X., 2021. Performance evaluation of machine learning algorithms for intrusion detection in industrial internet of things environment. *Journal of Ambient Intelligence and Humanized Computing,* 12(12), pp. 12793-12805.

Hatami, N. M. A. &. M. S., 2019. Intrusion Detection System Using SVM and Feature Selection. *International Journal of Advanced Computer Science and Applications,* 10(6), pp. 45-51.

Hodo, E. B. X. H. A. D. P. L. &. T. C., 2016. Machine learning in computer network intrusion detection: A survey. *ACM Computing Surveys (CSUR),* 48(4), pp. 1-37.

Johnson, M., 2019. Techniques Used in Intrusion Detection. *ACM Transactions on Information and System Security,* 12(4), pp. 287-304.

Jones, A. & Williams, B., 2021. Importance of Intrusion Detection. *International Journal of Information Security,* 18(2), pp. 112-130.

Kamal, T. R. A. &. H. A., 2021. Comparative analysis of machine learning algorithms for intrusion detection using NSL-KDD dataset. Volume 9, pp. 34302-34320.

Kandhari, A. K. N. &. K. V., 2020. A deep learning-based intrusion detection system for the Internet of Things. *IEEE Internet of Things Journal,* 7(2), pp. 1146-1156.

Kavi, K. V. D. M. a. R. P. C., 2017. Comparative Study of Machine Learning Techniques for Intrusion Detection System. *International Journal of Computer Science and Network Security,* 17(8), pp. 106-114.

Khan, M. K. Z. Y. &. B., 2017. Intrusion detection techniques and approaches. *ACM Computing Surveys (CSUR),* 50(3), pp. 1-36.

Khan, S. &. K. S. U., 2018. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications,* Volume 107, pp. 42-52.

Khan, S. U. K. M. A. &. A. B., 2015. Anomaly-based intrusion detection system: A machine learning approach. *Computers & Electrical Engineering,* Volume 46, pp. 38-58.

Kolias, C. K. G. S. A. &. G. S., 2016. A comparative analysis of machine learning techniques for network intrusion detection. *Journal of Information Security and Applications,* pp. 1-12.

Kotsiantis, S. B., 2017. Supervised machine learning: A review of classification techniques. *Informatica,* 31(3), pp. 249-268.

Kumar, A. A. V. &. G. D., 2022. An Overview of Intrusion Detection Systems in Cyber Security. *International Journal of Advanced Research in Computer Science and Software Engineering,* 12(2), pp. 1042-1048.

Kumar, s. & Ahuja, p., 2019. A Review of Intrusion Detection Systems: Its Issues and Challenges. *Journal of Computer Networks and Communications,* pp. 1-19.

Lee, J. L. S. &. K. H., 2021. An efficient malware detection system based on machine learning in network communication. *Security and Communication Networks,* pp. 1-11.

Li , Y., Li, L., Li, X. & Zhang, J., 2020. An improved intrusion detection method based on deep learning for big data environment. *Journal of Ambient Intelligence and Humanized Computing,* 11(9), pp. 3589-3599.

Li, Y., Zhang, J. & Cai, Z., 2018. A novel feature reduction approach for intrusion detection using SVM. *Journal of Computer Security,* 26(5), pp. 487-502.

Luong, T. ,. P. K. ,. L. T. &. N. T., 2019. Ensemble Learning Based Intrusion Detection System using Decision Tree. *International Journal of Advanced Computer Science and Applications,* 10(11), pp. 450-457.

Mirjalili, S. J. ,. J. S. S. &. M. S. S., 2016. Performance evaluation of machine learning algorithms for intrusion detection using NSL-KDD dataset. *International Conference on Knowledge Engineering and Applications,* pp. 80-84.

Moustafa, N. S. J. C. G. &. S. E., 2017. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset. *Information Security Journal: A Global Perspective,* 26((1-3)), pp. 18-31.

Patcha, A. & Park, J. M., 2015. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks,* Volume 90, pp. 34-64.

Pérez, F. & Granger, B. E., 2016. Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science. *Positioning and Power in Academic Publishing: Players, Agents and Agendas. IOS Press,* pp. 87-90.

Roy, S. B. A. &. C. N., 2019. A comparative study of machine learning algorithms for network intrusion detection system.

Sahib, S. A. W. &. B. M. H., 2020. Performance evaluation of machine learning algorithms for DDoS attack detection. *Journal of Ambient Intelligence and Humanized Computing,* 11(10), pp. 4267-4276.

Sathya, P. &. A. L., 2021. An ensemble of machine learning classifiers for intrusion detection system in cloud computing environment. *International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT),* pp. 545-550.

Saurabh, K., 2019. Intrusion Detection System using Random Forest. *s.l.: s.n.*

Shanthi, N. &. S. R., 2020. Performance evaluation of machine learning algorithms for intrusion detection: A literature review. *Journal of Network and Computer Applications,* Volume 168, p. 102742.

Sharma, K. K. A. &. K. P., 2019. Comparative analysis of machine learning algorithms for intrusion detection system. *International Conference on Computing Methodologies and Communication (ICCMC),* p. 238–242.

Sharma, R. C. S. S. &. K. A., 2021. Evaluating the performance of deep learning algorithms for intrusion detection. *Computers & Security,* Volume 103, p. 102251.

Shukla, A. K. & Gupta, B. B., 2015. Network Intrusion Detection System: A Review. *International Journal of Computer Applications,* 110(11), pp. 9-14.

Smith , k. & Johnson, L., 2019. Comparative Analysis of Machine Learning Techniques for Intrusion Detection. *Journal of Cybersecurity Research,* 15(3), pp. 45-62.

Smith, J., 2022. Introduction to Intrusion Detection. *Journal of Cybersecurity,* 25(3), pp. 45-62.

Sreenath, S. &. S. S., 2021. Intrusion Detection Techniques: A Review. *In International Conference on Intelligent Data Communication Technologies and Internet of Things,* pp. 17-25.

Sullivan, D. G., 2018. The Economic Impact of Cybersecurity Breaches and Intrusions. *Journal of Cybersecurity Research,* 6(1), pp. 25-38.

Vasantha, K. S. E. a. J. M. D. S., 2019. Evaluation of machine learning algorithms for intrusion detection system. *Journal of Ambient Intelligence and Humanized Computing,* 10(4), pp. 1543-1553.

Wang, H. W. Z. W. Q. &. G. Y., 2021. An intrusion detection system based on CNN. *International Journal of Distributed Sensor Networks,* 17(2).

Wang, J. ,. J. Z. W. &. W., 2019. Deep Learning for Network Intrusion Detection. *A Survey,* Volume 7, pp. 101826-101839.

Xu , L., Krzyzak, A. & Suen, C. Y., 2018. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* 29(3), pp. 292-303.

Yang, L. L. J. L. X. &. L. C., 2020. Intrusion detection for industrial control systems based on machine learning algorithms. *International Journal of Distributed Sensor Networks,* 16(15).

Yaseen, M. M. H. &. R. M., 2017. Performance evaluation of machine learning algorithms for network intrusion detection. *ournal of Intelligent & Fuzzy Systems,* 32(1), pp. 321-329.

Yuan, X. H. Q. &. Z. C., 2018. Deep learning for intrusion detection: A comparative study. *Journal of Big Data,* 5(1), pp. 1-17.

Zhang, K. W. Y. &. T. J., 2021. Performance Evaluation of Machine Learning Algorithms for Network Intrusion Detection. *International Conference on Computational Intelligence and Security (CIS),* Issue 17, pp. 329-334.

Zhang, L. &. J. Y., 2019. A comparative study of machine learning algorithms for intrusion detection. *Journal of Ambient Intelligence and Humanized Computing,* 10(4), pp. 1577-1585.

Zhang, Y. L. B. X. G. L. Z. &. L. X., 2019. A comprehensive evaluation of support vector machine for intrusion detection. Volume 7, pp. 165792-165800.