COMPARATIVE ANALYSIS OF TREE-BASED INTRUSION DETECTION MODELLING AND MACHINE LEARNING CLASSIFICATION MODELS USING CYBER-SECURITY DATASET

BY

MOTLATSO SAREL MOKOELE

DISSERTATION

Submitted in fulfilment of the requirements for degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

In the

FACULTY OF SCIENCE AND AGRICULTURE

(School of Mathematical and Computer Sciences)

At the

UNIVERSITY OF LIMPOPO

SUPERVISOR:
PROF. SN MOKWENA

2024

DEDICATION

I dedicate this research to my father, whose unwavering support and boundless love remain in my heart. Your wisdom, guidance, and the values you instilled in me continued to shape my path and fuelled my aspirations. Although you may not be here to witness this milestone, your spirit lives on in every endeavour I undertake. With deep gratitude for the permanent mark, you left on my life, I dedicate this work to your memory, forever cherishing the lessons you imparted and the love you shared.

DECLARATION

The dissertation, titled ‘COMPARATIVE ANALYSIS OF TREE-BASED INTRUSION DETECTION MODELLING AND MACHINE LEARNING CLASSIFICATION MODELS CYBER-SECURITY DATASET’, represents an original contribution to the field of cybersecurity and machine learning. I, _____Motlatso Sarel Mokoele_____, hereby declare that the work presented in this dissertation is entirely my own, and all the sources used in this study have been properly cited and acknowledged. Furthermore, I declare that this work has never been submitted before for credit at any university.

Signature:  Date: 29/11/2023

ACKNOWLEDGEMENTS

I want to express my gratitude to the Almighty for the strength and guidance that have helped me on this journey. I am very grateful to my mother for her constant love and support, which have meant the world to me. I also wish to thank Prof. SN Mokwena, my research supervisor, for his guidance. I appreciate my classmates, MR Mulaudzi T and MS Ndleve NM, for the camaraderie and encouragement we shared.

To my family, I am grateful for their support and understanding. My friend, MR Kekana MTM, has been a constant source of motivation. MR Ramasenya GW has been a great motivator for my research. Last but not least, I am grateful for the support of my girlfriend, Maepa MR, who has been there for me throughout this journey.

ABSTRACT

Cybersecurity has become an ever-pressing concern in the modern digital landscape, demanding robust and efficient intrusion detection systems. In this research, we conducted a comparative analysis of tree-based intrusion detection modelling and several popular machine learning classification models, using the widely used KDD99 dataset. To enhance the efficiency of the proposed model, we employ a hybrid feature selection method that combines the Gini index and information gain and incorporates them using the concepts of a decision tree (DT). Models under evaluation include DT, Support Vector Machine (SVM), K-Nearest Neighbours (KNN), and Logistic Regression (LR).

We present a comprehensive evaluation of these models based on various performance metrics, including accuracy, F1 score, confusion matrix, precision, recall, and execution time. The dataset is meticulously pre-processed to eliminate noise and address any biases that may affect the results. The findings of this research reveal important insights into the strengths and weaknesses of different intrusion detection models. Our analysis sheds light on the performance variation between the tree-based model and SVM, KNN, and LR. In addition, we discuss the factors that contribute to the observed effectiveness of the model.

The results demonstrate the effectiveness of the hybrid feature selection approach in enhancing the performance of tree-based models. In addition, we identify the most suitable models for specific performance criteria, guiding practitioners in selecting the appropriate model for their specific intrusion detection requirements.

The results of this study contribute significantly to the advancement of intrusion detection techniques and provide valuable guidance to cybersecurity practitioners and researchers. The research highlights potential areas for further investigation and improvement, paving the way for more efficient and accurate intrusion detection systems in the future.

Keywords: Cyber-security, Intrusion Detection, Machine Learning, Hybrid Feature Selection, Tree-based Intrusion Detection Modelling, Support Vector Machine, K-Nearest Neighbours, logistic regression, Decision Tree

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

IDS – Intrusion Detection Systems

ML – Machine Learning

IoT – Internet of things

TP – True positive

FP – False Positive

MCC – Matthews correlation coefficient

DR – Detection Rate

FN – False Negative

SVM – Support Vector Machine

DT – Decision Tree

KNN – K-Nearest Neighbours

LR - logistic regression

SAE – Sparse autoencoder

HIDS – Hybrid Intrusion Detection System

NBFS- Nave Base Feature Selection

OSVM – Optimised Support Vector Machine

AIDS - Anomaly-Based Intrusion Detection System

SIDS – Signature Intrusion Detection System

AHIDS – Advanced Hybrid Intrusion Detection System

WDA – Welcome Deluge Attack

WA – Wormhole Attack

HNIDS – Hybrid Network-Based Intrusion Detection System

EGA-PSO – Enhanced Genetic Algorithm and Particle Swarm Optimization

IRF – Improved Random Forest

RF – Random Forest

OC-SVM - One-Class Support Vector Machine

NB - Nave Bayes

ADFA – Australian Defence Force Academy

IntruDTree – Intrusion Detection Tree

AV - autonomous vehicle

ITS – Transportation system

V2X – Vehicle-to-everything

IoV – Internet of Vehicles

SET – Stacking Ensemble Technique

DT-RFE – Decision Tree Recursive Feature Elimination

TCP – Transmission Control Protocol

ICMP – Internet Control Message Protocol

UDP – User Datagram Protocol

RNN – Recurrent neural network

DNN – Deep neural network

NID – Network Intrusion Detection

ID – Intrusion Detection

CHAPTER 1 – INTRODUCTION

1. INTRODUCTION

Cybersecurity is a life-threatening concern, with the proliferation of cyber threats necessitating the development of robust intrusion detection systems (IDS). These systems play an essential role in protecting sensitive data, critical infrastructure, and the integrity of digital ecosystems. Incorporating machine learning (ML) with cybersecurity has opened up a new prospect for intrusion detection (ID). Among ML techniques, tree-based models, such as DTs, random forests (RFs), and gradient boosting, have gained prominence for their interpretability and classification capabilities.

This dissertation aims to provide a comprehensive comparative analysis of tree-based ID models and traditional ML classification models, using a cyber-security data set. It aims to elucidate the strengths and limitations of these models, offering information on their performance in different threat scenarios. The study addresses key research questions, including performance matrix, development of a tree-based ID model, application of the cyber security dataset to ML classification algorithms, selection and ranking, and efficiency comparisons between tree-based and traditional ML models, identification of the most effective tree-based model and the factors influencing model selection in the cybersecurity context.

As the cybersecurity landscape evolves, understanding model selection becomes paramount for improving identity. Research results are expected to make a significant contribution to the field, helping cybersecurity practitioners make informed decisions about the selection of ID models. By comparing tree-based models with traditional ML classifiers, this study aims to shed light on the most suitable approaches to counter evolving cyber threats in our increasingly digital world. A research article based on this work was published in International journal of Computer Application as indicated in section 1.11.

1.1. PROBLEM STATEMENT

The complexity of data produced by organisations through IDSs can be difficult to analyse. Existing data and network access defences, including hardware, firewalls, user authentication, and data encryption techniques, are insufficient against the variety of cyber threats that affect computer networks [1]. Due to the rapid evolution of intrusion techniques, traditional security mechanisms are insufficient as a form of access defence. A firewall merely regulates network access, which means that it restricts access between networks. There are several types of IDS employed to manage these forms of threat in companies and organisations. For example, they are host-based and network-based IDSs. The host-based IDSs reside on a separate system and keep a watch on operating system files for inconsistencies and anomalies in the activities. The network-based IDSs system, on the other hand, monitors and probes network connections for unauthorised traffic. This approach has challenges in identifying new unknown network attacks and may be subjected to high false alarm rates. For this reason, ML classification models are studied and employed as an optimal technique to handle these challenges. But for security purposes, the potential and fruitfulness of ML-based ID modelling is a challenge for IT workers, e-commerce and software developers. A cyber-security data collection, in general, comprises several types of cyber-attacks with important parameters. Therefore, certain classifiers may not produce an adequate prediction accuracy and real prediction rate, depending on a variety of attack types and variables. Therefore, it is a need to create accurate machine learning-based ID models to correctly detect these difficulties that threaten organisations.

1.2. MOTIVATION

Today, the requirement for cyber security and fortification in contrast to many forms of cyber security issues is continuously accumulating. The primary reason for this accumulation is due to the influence of the Internet of Things (IoT), the enormous amount of development and advances in computer networks, and the large number of vital applications used by people or organisations [2]. Cyber-attacks such as denial of service, computer malware, and unauthorized access caused catastrophic damage and financial losses. For example, as stated by Alqahtani [3] in May 2017, a single ransomware virus incurred a large cost to various companies and industries, together with finance, healthcare, energy, and tertiary institutions, causing a loss of

approximately 8 billion dollars. Sarker in [2] has shown that a data breach expenditure on an impacted company usually costs 3.9-8.19 million US dollars. Once again on 9 May 2022, in its State of Ransomware 2022, the British cybersecurity firm Sophos stated that 51% of South African organisations it surveyed had experienced ransomware in 2021. Other key findings included a significant portion (49%) of organisations paying ransom remands and the effects of a ransomware attack, with a cost of recovery of approximately R11.5 million [4]. On 18 March 2022, it was revealed that the South African Credit Bureau TransUnion had been hacked for ransom and that hundreds of businesses were in danger. According to reports, hackers used an authorised client's login information to access the bureau's server. They were referred to as "criminal third parties" [5]. The organisation suffers these attacks because its network access control comprises traditional security mechanisms. The utilisation of classifiers that produce inadequate prediction accuracy may also cause these damages since the cyber-security dataset comprises several types of cyberattacks with important parameters.

Sarker in [2] suggested an ID tree that takes into account the priority of ranking security characteristics before building a tree-based algorithm based on the selected significant feature. According to the study, the problem is that attacks such as denial of service attacks, ransomware, or unauthorised access caused irrevocable damage and monetary losses in a large network. The purpose was to investigate methods to reduce security risks and developing a powerful data-driven ID model.

Al-Omari in [6] used artificial intelligence to detect and defend against cyberattacks on heterogeneous devices. The objective is to suggest an intelligent ID methodology to anticipate and recognise cyberattacks. The author identified a concern that the massive volume of data, which has various dimensions and security aspects, can reduce the systems' ability to make accurate predictions and raise their computational complexity.

Buczak in [7] focuses on a literature assessment of data mining and ML techniques for cyber analytics in support of ID, while use-based techniques (a form of cyber analytic methodology used to enhance IDSs) are excellent at recognising certain types of assault without causing a high number of false alarms; they usually require human

updates to the database's rules and signatures. They are unable to detect novel (zero-day) attacks.

Andresini in [8] designed a new deep learning neural network (DNN) architecture combining an unsupervised and supervised stage of multichannel learning feature sets. According to the study, current IDSs are still struggling to improve detection accuracy by lowering false alarm rates and recognising unidentified threats. The purpose was to see if learning and incorporating class-specific network flow features into the original could improve the model accuracy.

1.3. AIM

The purpose of the study was to comparatively analyse tree-based ID model and ML classification algorithms using a cyber security dataset.

1.4. OBJECTIVES

The objectives of the study are the following.

i. Develop a tree-based detection model using the concepts of DT.
ii. Conduct security feature selection and ranking to select features of significant importance.
iii. Apply the tree-based ID model and traditional ML classification models to the KDD-99 dataset.
iv. Evaluate the effectiveness of these models by measuring performance matrix, namely accuracy, precision, recall, F1 score, and confusion matrix.

1.5. RESEARCH QUESTIONS

The research questions for the study are as follows.

i. How to develop a tree-based ID algorithm based on the concepts of a DT?
ii. How to conduct the selection and ranking to select features of significant importance?
iii. How to apply a cybersecurity dataset to a tree-based ID model and traditional ML classifiers?
iv. How to evaluate the effectiveness of the models using the performance matrix?

1.6. METHODOLOGY

This section outlines the comprehensive research methodology employed in this study to investigate the effectiveness of ML classification algorithms in the domain of cybersecurity. The data set under consideration is extracted from the UCI ML repository, which was originally utilised for the 3rd International Knowledge Discovery and Data Mining Tools Event held in conjunction with KDD-99. This data set contains a standard set of auditable data that includes various simulated intrusions into a military network environment, comprising 494021 instances, each characterised by 42 properties.

The research methodology begins with an exploration of the tools used, focussing on the use of Google Collaboration as the platform for conducting experiments. This cloud-based Jupyter notebook environment provides a user-friendly interface and facilitates data analysis and model training. Following this, the study delves into data analysis, where the quality and characteristics of the dataset are examined. This includes the validation of the number of records and features, analysis of instances identified as attacks, variable categorisation, and feature correlation with the predicted attribute. Subsequently, the data preprocessing and standardisation steps are discussed, involving the elimination of redundancies, the transformation of categorical variables into dummy variables, and feature scaling to ensure consistent units of measurement. Additionally, feature selection and ranking methodologies, specifically the Gini index technique and information gain, are presented to identify attributes that have a significant influence on decision-making processes. Finally, we introduce tree-based ID modelling, highlighting the construction of a model based on security features with reduced dimensions and improved accuracy. The selection of root nodes using the Gini index is highlighted, followed by a discussion of the traditional ML classification algorithms, such as SVM, KNN and LR, to be applied in this study. A detailed discussion of each of these research methods will follow in chapter 3 of the study, providing a comprehensive understanding of the approach taken to investigate the effectiveness of ML classification algorithms in cybersecurity, using the KDD-99 dataset.

1.7. SCIENTIFIC CONTRIBUTION

Organisations are experiencing difficulties in dealing with cyber-attacks on daily basis. This study will have a significant contribution to organisations facing these issues by

accurately improving the detection of cyberattacks and finding the appropriate model to be used when detecting these attacks. This will help organisations reduce the amount of money used to maintain and upgrade firewalls and antivirus applications. The study will compare several popular ML classification algorithms with the proposed tree model to help the organisation understand and know which model is required to manage all cyber security issues facing organisations. Helping the organisation to safely protect their systems and data encourages transparent working environment and strong data protection in the organisation.

1.8. AVAILABILITY OF RESOURCES AND INFRASTRUCTURE

The University of Limpopo's Department of Computer Science provides the necessary resources and infrastructure, including software and strong network connectivity to undertake this research.

1.9. ETHICAL CONSIDERATION

The study does not require ethical approval because the study will be conducted on the data set that was used before and it is available on websites for extraction.

1.10. SCOPE AND DELIMINATION

This research focusses on comparative analysis of tree-based ID models and traditional ML classification algorithms using the KDD-99 cybersecurity dataset. The study explores the development of a tree-based ID model, feature selection and ranking techniques, and the application of a tree-based model and traditional ML classifiers to the dataset. We evaluate their effectiveness by measuring various performance metrics. The study is limited to the specific cyber security data set mentioned, and the findings may not be generalised to other data sets. Additionally, the research does not include ethical considerations in detail as it does not involve human subjects, but rather focusses on the technical aspects of ID modelling.

1.11. RESEARCH OUTPUT FROM THE DISSERTATION

Motlatso Mokoele, Sello Mokwena . Comparative Analysis of Tree-based Intrusion Detection Modelling and Machine Learning Classification Models using Cyber-Security Dataset. International Journal of Computer Applications. 186, 13 (Mar 2024), 33-40. DOI=10.5120/ijca2024923480

1.12. SUMMARY

In this chapter, we explore the realm of cybersecurity, recognising its paramount significance in our digital age, where the surge in cyber threats requires the development of robust ID systems. With the integration of ML into the cybersecurity domain, tree-based models such as DTs have emerged as crucial tools, renowned for their interpretability and classification capabilities. This chapter introduces the central objective of our dissertation to perform a comprehensive comparative analysis of these tree-based ID models and traditional ML classification models, employing a cyber-security dataset. Our goal is to reveal the strengths and limitations of these models, offering valuable insights into their performance in various threat scenarios, and identifying the factors influencing the selection in the cybersecurity field. This research aspires to make a substantial contribution to the field, equipping cybersecurity professionals with the knowledge to make informed decisions about the selection of ID models, ultimately reinforcing data protection in our increasingly digital world.

1.13. OVERVIEW OF DISSERTATION

The rest of the dissertation is organised as follows.

Chapter 2 provides a review of the literature.

Chapter 3 presents a detailed description of the design and implementation of the DT, SVM, KNN, and LR.

Chapter 4 presents the research findings of the study.

Chapter 5 presents a summary and recommendations for future work.

CHAPTER 2 – LITERATURE REVIEW

2. INTRODUCTION

Intrusion detection (ID) techniques have drawn an impactful attention from most researchers in the past few years, as cited in most of the studies below. ML classification algorithms are very important for identifying a category in a given data set. Although classifiers have many benefits, their major drawback in cyber security is that they tend to produce inadequate prediction accuracy and real prediction rate, depending on a variety of attack types and variables. ID techniques have challenges in identifying new unknown network attacks and may be subjected to high false alarm rates.

Several factors that have been investigated in the literature and have illustrated the use of ML models in cyber security. The cyber security data set was extracted and will be further used in our proposed study to better understand which classifier can outperform a tree-based ID model.

This chapter is structured as follows: discussed in Section 1.1 ID techniques contain Section 1.1.1 misuse detection technique, 1.1.2 anomaly-based detection technique, and 1.1.3 hybrid detection technique. In Section 1.2 we discuss tree-based ID modelling.

2.1. INTRUSION DETECTION TECHNIQUES

In the ever-evolving landscape of cybersecurity, identification plays a pivotal role in protecting systems and networks from unauthorised access and malicious activities. Recent years have witnessed a surge in sophisticated cyber threats, necessitating the constant advancement of identification techniques to effectively mitigate risks. Signature-based detection, a traditional approach, relies on predefined patterns or signatures of known attacks. Although effective against well-established threats, it struggles to detect novel or evolving attacks [9]. Recent research has focused on improving signature-based systems through ML algorithms, enabling more adaptive and dynamic detection capabilities. Anomaly-based detection aims to identify deviations from normal system behaviour [10]. This technique uses statistical models and ML algorithms to establish a baseline of normal activities and raise alerts when deviations occur. Recent studies have explored the integration of deep learning and

anomaly-based approaches, demonstrating improved accuracy in detecting subtle and previously unseen intrusions.

To address the limitations of individual detection techniques, researchers have increasingly advocated for hybrid approaches that combine the strengths of signature-based and anomaly-based methods [24], [38]-[51]. These integrated systems leverage both historical knowledge of known attacks and the ability to adapt to emerging threats. Recent studies highlight the effectiveness of such hybrid models in improving overall detection accuracy and reducing false positives. Behavioural analysis focusses on understanding user and system behaviour to detect anomalous activities [11]. ML algorithms analyse patterns and trends, allowing the identification of deviations that may indicate a security threat. Recent advances in behavioural analysis involve the incorporation of contextual information and real-time monitoring for more precise and proactive identification [12] and [13]. With the increasing adoption of cloud computing, intrusion detection has expanded its scope to address the unique challenges posed by cloud environments. Recent research explores techniques tailored for cloud-based architectures, considering factors such as multitenancy, virtualisation, and the dynamic nature of cloud resources [14]. The landscape of ID techniques continues to evolve rapidly, driven by the relentless innovation of cyber threats. Recent research emphasises the integration of ML, deep learning, and hybrid models to improve the accuracy and adaptability of IDSs. As the cybersecurity community faces new challenges, staying abreast of these developments is crucial to maintaining robust defence mechanisms against emerging threats.

ID techniques play an important role in safeguarding the integrity and security of modern information systems. There are methods designed to identify and respond to malicious activities, cyberattacks, and unauthorised access within a network or computing environment. With the ever-evolving landscape of cyber [15] threats, researchers and practitioners have been constantly developing and refining ID techniques to stay one step ahead of cybercriminals. This introduction will briefly explore some recent developments and key techniques in this field, including tree-based ID, SVM, KNN, and LR. Tree-based ID models, such as RFs and DTs, have gained significant attention due to their versatility and adaptability to identify anomalous activities within network traffic. Recent studies have demonstrated the effectiveness of RF and DT in accurately detecting network intrusions by analysing network data patterns [16], [17], and [18]. SVMs are another class of ID techniques

which were illustrated in [18] to efficiently classify network traffic into normal and malicious categories. KNN and LR are also noteworthy techniques in ID. These were studied in [19] and [1]. These techniques collectively contribute to the ongoing efforts to improve the security of digital systems by enabling early identification and mitigation of potential threats. Researchers are continually refining these methods and exploring novel approaches to address the ever-evolving landscape of cybersecurity challenges. In order to provide a comprehensive view of security requirements in a cloud environment [20], investigations were carried out to collect and classify both attacks and vulnerabilities related to the different cloud models. This study led to the development of a taxonomy that delineates cloud security threats and proposes possible measures to mitigate them. The main objective of this research was to emphasise the importance of detection and prevention of intrusions as a service offered in cloud environments.

To ensure Internet security, effective detection and mitigation of distributed denial-of-service (DDoS) attacks [21], novel collaborative intrusion prevention architecture (CIPA) was proposed, with the aim of antagonising coordinated intrusion activities. The architecture was deployed as a virtual network of an artificial neural network over the substrate of networks. The CIPA takes advantage of the parallel and simple mathematical manipulation of neurones in a neural network, since it can separate its light-weight computation supremacy from the programable switches of the substrate.

2.1.1. MISUSE DETECTION TECHNIQUE

In the cybersecurity domain, the concept of misuse detection is characterised by its objective of pinpointing potential instances of network attacks by assessing ongoing activities and comparing them to the typical behaviour of an intruder.[22] introduced a novel perspective on the task of misuse detection, using the robust analytical capabilities of neural networks. This study also presented its preliminary findings, which shed light on the efficacy of this innovative approach. The motivation behind exploring this alternative approach stemmed from the limitations of conventional methods in misuse detection, which predominantly rely on rule-based expert systems to identify established signs of well-known attacks. However, such techniques tend to fall short when it comes to detecting attacks that deviate from anticipated patterns.

To avoid illegitimate use and poor detection accuracy of any intruder in the network [23], a hybrid misuse ID model was proposed containing the advantage of feature

selection and ML techniques. The model was constructed by analysing the feature selection methods and identifying the essential features of the NSL-KDD data set.

In this chapter, [24] discussed the critical role of ML and data mining algorithms in the development of ID systems. These systems were categorised into two main types: misuse detection, which focused on identifying known attack signatures by matching network activity patterns, and anomaly detection, which highlighted deviations from established normal system behaviour. The chapter provided a comprehensive exploration of these approaches, shedding light on their strengths and limitations in the ever-evolving field of cybersecurity. Moreover, the chapter investigated the integration of these two approaches through the emergence of hybrid detection systems. These systems aimed to harness the advantages of both misuse and anomaly detection to create a more resilient and adaptable defence mechanism against a broad spectrum of cyber threats. As the chapter unfolded, it became evident that ongoing research and innovation in the form of advanced algorithms were imperative to keep IDSs proactive and responsive to the dynamic tactics employed by malicious actors in the digital realm. In summary, the chapter offered valuable insights into the state of ID, paving the way for future developments in the field of cybersecurity.

The study in [25], focused on the role that IDS in safeguarding information and communication technology (ICT) infrastructures. Specifically, the study examined the merits and limitations of a misuse IDS, which had shown stability in achieving high attack detection rates while maintaining an acceptable level of false alarms. However, the research highlighted a notable challenge faced by IDSs that are misused: their inherent lack of adaptability to new and "unknown" environmental conditions. This limitation placed a considerable responsibility on security administrators, necessitating ongoing, labour-intensive efforts to keep the IDS up-to-date, particularly as it encountered fluctuations in efficiency. To address this challenge, the study proposed a pioneering methodology that skilfully harnessed the benefits of self-taught learning and the MAPE-K framework. This combined approach led to the creation of a scalable, self-adaptive, and autonomous use IDS. By leveraging deep learning-based techniques, the IDS acquired the ability to decipher the nature of attacks based on generalised feature reconstructions derived directly from the unknown environment and its unlabelled data. The experimental results substantiated the effectiveness of

this methodology, revealing its capacity to revitalise IDS without the constant requirement of manual updates to its training set. The study's evaluation, conducted using a range of classification metrics, underscored that in critical scenarios where a statically trained IDS would have faltered, the attack detection rate of the IDS increased substantially, by up to 73.37%.

2.1.2. ANOMALY-BASED DETECTION TECHNIQUE

The investigation carried out with reference to [26] revealed a significant gap in the existing body of knowledge on the comprehensive assessment of various algorithms used in anomaly-based ID. Specifically, prior to this study, no extensive examination had been carried out to systematically compare a diverse array of ID algorithms across a wide spectrum of attack datasets and attack types. To address this gap, the research project undertook the task of evaluating 12 unsupervised anomaly detection algorithms using five distinct attack datasets. The results of this evaluation provide a wealth of information, ranging from the performance characteristics of individual algorithms to the appropriateness of specific datasets for the purpose of anomaly detection. The study's author succeeded in identifying and categorising algorithms that exhibit greater efficacy in the realm of ID, as well as those that demonstrate resilience in the face of varying configuration parameters. Furthermore, the study's experimental findings confirmed the inherent challenges in detecting attacks characterised by unstable and non-repeatable behaviours. It also highlighted the observation that data sets that contain a limited number of anomaly events tend to yield superior detection accuracy.

In response to the increasing surge in network data traffic, which has led to a slight increase in system viruses and cyberattacks, the study described in the reference [27] sought to establish an efficient network intrusion anomaly detection methodology. To address this challenge, the researchers introduced a novel computer network intrusion detection (NID) model that uses recurrent neural networks. The primary objective of this research was to develop an architecture for a network security emergency response system that incorporates a variety of collaborative modules powered by the recurrent neural network model. The core innovation lay in the creation of a NID model that included a risk analysis and processing module. This module combined bidirectional long- and short-term memory (BiLSTM) techniques with deep neural networks (DNN) to uncover relationships between various features, while also

employing DNN to extract deeper and more complex features within the data. The experiments carried out as part of this study served to validate the reliability and effectiveness of the methodologies introduced by the author.

In pursuit of a more accurate IDS while minimising false alarms, the study described in reference [28], introduced an improved method for outlier detection. This enhancement was designed to increase the precision of ID by incorporating density-based outlier detection techniques into data mining, using Hamming densities as a means of assessing the abnormality of data points. This involved the calculation of Hamming densities by dividing the KNN by the Hamming distance. To measure the effectiveness of the proposed model, the researchers conducted experiments using the KDD CUP'99 intrusion dataset sourced from the UCI repository, running simulations and comparing their approach with existing algorithms such as LOF and LOF. The findings revealed that the proposed method achieved higher accuracy and increased sensitivity, thereby improving the ability to detect true positive alarms (TP).

There are methods and tools at our disposal to identify threats within a computer network, but among these, ML techniques stand out as one of the most effective resources to identify and distinguish abnormal network traffic with precision and precision [29]. This study introduced an approach to differentiate whether incoming network traffic falls within the domain of normalcy or deviates into the abnormal, all through the use of ML techniques. The researchers evaluated multiple classifiers using the NSL-KDD data set. The experimentation encompassed the application of various models, including KNN, DT, naïve Bayes (NB), LR, RF and an ensemble approach. The principal objective of this research was to achieve a high level of performance by implementing a straightforward feature selection strategy on the dataset, which was subsequently employed in both the traditional ML and the ensemble learning method.

To address the difficulties associated with the implementation of anomaly detectors [30], a pioneering strategy rooted in supervised ML was put forth. The methodology involved the simultaneous application of various established detectors to performance data, facilitating the extraction of anomaly features. These identified features, along with the corresponding labels, were used to train a RF classifier. The primary function of this classifier was to autonomously determine suitable detector parameter

combinations and thresholds. Consequently, the introduced system allows operators to label data within a matter of minutes, a stark contrast to the traditional method that required more than ten days for the arduous task of selecting and fine-tuning detectors. It should be noted that such traditional efforts may not produce satisfactory results in the end.

The research outlined in [31] conducted a comprehensive systematic review of the literature, which focused on the analysis of the ML models employed for the detection in various applications. The review examined these models from four key perspectives: the diverse applications of anomaly detection, a range of ML techniques employed, the performance metrics used to evaluate these ML models, and the categorisation of anomaly detection methods. The study meticulously identified and reviewed a total of 290 papers published between the years 2000 and 2020 that discussed ML techniques for anomaly detection. Upon thorough analysis of the selected research papers, the author highlighted 43 distinct applications where anomaly detection is applied. Furthermore, the research also identified and presented 29 unique ML models that are commonly used to identify anomalies, along with 22 different data sets that are frequently used in experiments involving anomaly detection.

According to the insights provided in reference [32], intrusions into computer or network systems are defined as actions that disrupt the established attributes of a secure and stable system, thus compromising its security with regard to confidentiality, availability, or integrity. The central theme of this book revolves around the characterisation of anomalies within networked computer systems and the subsequent utilisation of ML techniques for their detection. The book investigates the vulnerabilities that network systems encounter across different layers, often stemming from weaknesses in protocols or other contributing factors. The author introduces a ML-based approach to combat network intrusions, categorising these approaches into several distinct groups, including supervised learning, unsupervised learning, probabilistic learning, soft computing, and combination learners.

The review paper [33], discussed the types of ID with a brief introduction of categories of anomaly detection which is one of the types of IDS. The ML-based anomaly detection techniques are also discussed, which were presented to base on an explicit

or implicit model that enables the patterns analysed to be categorised into Genetic Algorithms, Fuzzy Logic, Neural Networks, Bayesian Network, and outlier detection.

The study in [34], conducted a thorough evaluation of 12 different ML algorithms, specifically focussing on their ability to identify and flag anomalous behaviours within network operations. To carry out this evaluation, the study employed three openly available datasets. CICIDS-2017, UNSW-NB15 and Industrial Control System (ICS) cyberattack datasets. The experimental phase was carried out using the ALICE high-performance computing facility at the University of Leicester. Upon analysis of the results of these experiments, the study presented a comprehensive evaluation of various ML models. In particular, the results confirmed that the RF algorithm consistently demonstrated the highest performance across multiple metrics, including the F1 score, recall, precision, accuracy, and receiver operating characteristic (ROC) curves, for all the aforementioned data sets. It is worth noting that other algorithms closely trailed behind RF in terms of performance, suggesting that the choice of which ML algorithm to employ should depend on the specific data characteristics generated by the system application.

The study in [35] introduced a novel approach, involving the development of two-tier classification models rooted in ML techniques, specifically NB, a certainty factor voting version of KNN classifiers, and Linear Discriminant Analysis for dimension reduction. These models undertook rigorous experimentation based on the NSL-KDD dataset. The study's findings yielded highly encouraging results, showcasing a significant improvement in the detection rate and a reduction in false alarms when compared with existing models. To train these models, two balanced training sets were generated using the SMOTE (Synthetic Minority Oversampling Technique) method. This allowed for an assessment of the chosen similarity measure to address the challenges posed by imbalanced network anomaly datasets. A noteworthy advantage of this two-tier model was its efficiency in terms of computation time, attributed to optimal dimension reduction and feature selection. Additionally, the model exhibited strong detection capabilities, particularly in identifying rare and intricate attack types that pose a heightened risk due to their close resemblance to normal behaviours, such as user-to-root and remote-to-local attacks.

The study referenced [36], conducted a comparative analysis to assess the effectiveness of various ML models in accurately predicting attacks and anomalies within IoT systems. The ML algorithms employed in the study included LR, SVM, DT, RF, and artificial neural network. The performance of these models was evaluated using several key metrics, including accuracy, precision, recall, F1-score, and the area under the ROC curve. Notably, the system achieved a test accuracy of 99.4% for DT, RF, and Artificial Neural Network. However, despite the comparable test accuracy, the RF outperformed the other models in terms of other performance metrics, highlighting its superiority in various aspects of prediction and anomaly detection.

To address the issue of centralisation, which can result in severe disruption of the network system when potent malicious code is introduced [37], experiments were undertaken. These experiments employed supervised ML techniques for the development of a network anomaly detection system. The aim was to mitigate low communication costs and minimise network bandwidth usage. The UNSW-NB15 dataset was employed to evaluate the system's performance in terms of accuracy and processing time required for a classifier to construct a model. Among the ML algorithms examined, the best performer on the network dataset was determined to be AODE, achieving a commendable accuracy of 97.26%. The model construction process took approximately 7 seconds, demonstrating efficiency in both accuracy and processing time.

2.1.3. HYBRID DETECTION TECHNIQUE

The study referenced in [38], introduced an innovative and layered IDS, using a combination of various ML techniques and feature selection methods to achieve robust ID across a spectrum of attack types. The system's development process begins with data pre-processing on the NSL-KDD dataset. Subsequently, several feature selection algorithms are employed to reduce the dataset's size. This study introduces two novel approaches for conducting feature selection operations. The system's architecture adopts a layered approach, with the selection of ML algorithms tailored to the specific type of attack being considered. Performance evaluations, encompassing metrics like accuracy, Detection Rate (DR), TP rate, false positive (FP) rate, F-Measure, Matthews Correlation Coefficient (MCC), and the time required for system operation, are

conducted using the NSL-KDD dataset. To determine the effectiveness of the proposed system, it is systematically compared with existing studies in the literature, thus facilitating a comprehensive performance evaluation. The results affirm that the proposed system consistently delivers high accuracy and exhibits a low FP rate across all categories of attacks.

Tahir in [39] proposed a novel hybrid ML technique for NID, employing a combination of K-means clustering and SVM classifier. The primary aim of this experimental study was to mitigate the occurrence of FP alarms, reduce the false negative (FN) alarm rate, and improve overall DR in IDSs. The NSL-KDD dataset serves as the basis for this technique. To enhance the classification performance, various data preprocessing and refinement measures have been applied to the dataset. Subsequently, the classification task is carried out using the SVM. Following the training and testing phases of the hybrid ML technique proposed, the outcomes unequivocally illustrate its success in attaining a higher DR. Simultaneously, it adeptly reduces the incidence of false alarms, marking a notable enhancement in NID.

Aljamal in [40] proposed a network-based anomaly detection system situated at the cloud hypervisor level. This system leverages a hybrid algorithm, specifically a blend of the K-means clustering algorithm and the SVM classification algorithm, aimed at enhancing the precision of the anomaly detection system. The evaluation of this approach utilises the dataset from the UNSW-NB15 study, and the obtained results are compared with findings from previous studies. Notably, the accuracy of the proposed K-means clustering model slightly surpasses that of others. However, it is imperative to highlight that the accuracy derived from the SVM model remains comparatively low within the context of supervised techniques.

The study in [41], proposed an IDS for the IoT network and detected various categories of intrusions based on the hybrid convolutional neural network (CNN) model. The proposed paradigm is appropriate for a wide variety of IoT applications. The proposed research work is validated and contrasted with conventional ML and deep learning models. The experimental results demonstrate that the proposed hybrid model is more sensitive to attacks on the IoT network.

The investigation documented in [42] introduced an efficient hybrid layered IDS designed to identify both pre-existing and zero-day attacks. Notably, a two-layered system is proposed, encompassing misuse detection systems and anomaly ID. The initial layer incorporates a misuse detector, capable of identifying and thwarting known attacks. The subsequent layer comprises an anomaly detector, proficient in efficiently detecting and blocking previously unidentified attacks. The misuse detector is modelled on the FRs classifier, while the anomaly detector is formulated using the bagging technique with an ensemble of One-Class Support Vector Machine (OC-SVM) classifiers. Data pre-processing involves automatic feature selection and data normalization. The experimental outcomes underscore the superior performance of the proposed IDS when compared to other well-known IDSs. Its efficacy extends to the detection of both previously known and zero-day attacks, affirming its robustness in enhancing network security.

A two-stage hybrid methodology for ID was introduced by the study referenced in [43]. In the initial stage, an unsupervised Sparse Autoencoder (SAE) utilising smoothed L1 regularisation was implemented. This particular regularisation method was employed to enforce sparsity within the autoencoder, and it effectively facilitated the learning of sparse feature representations. The second stage involved the use of a DNN for predicting and classifying attacks. The classifier performed multi-attack classification based on the features extracted by the unsupervised SAE, which was optimised for efficient model training. The experimental outcomes showcase the superiority of the proposed model over conventional counterparts in terms of overall performance, including high DR and a low FP rate. The evaluation of the proposed model was conducted on the KDD Cup’99, NSL-KDD, and UNSW-NB15 datasets, achieving outstanding accuracy rates of 99.98% on KDD Cup’99 and NSL-KDD, and 99.99% on the UNSW-NB15 dataset.

The aim of [44] was to devise a Hybrid Intrusion Detection System (HIDS) suitable for real-time deployment, specifically tailored to address multiclass classification challenges. The foundation of HIDS lies in the Naïve Base Feature Selection (NBFS) technique, strategically employed to diminish the dimensionality of the sample data. A unique feature of HIDS is its integration of outlier rejection, a facet not commonly found in other techniques. Outliers, being unpredictable input samples, can substantially

contribute to misclassification rates during model training. To counter this, an innovative approach utilising a distance-based methodology is employed to select the most informative training examples, subsequently used to train an Optimised Support Vector Machine (OSVM). The OSVM, in turn, serves to reject outliers. Post outlier rejection, HIDS adeptly identifies intrusions using a Prioritized K-Nearest Neighbour (PKNN) classifier.

A distinctive HIDS has been created, amalgamating a C5 classifier and an OC-SVM classifier for the safeguarding of IoT devices. This HIDS leverages the advantages of both Anomaly-based Intrusion Detection System (AIDS) and Signature-based Intrusion Detection System (SIDS) [45]. Striving for high detection accuracy and minimal false alarm rates, the system aims to discern both established intrusions and emerging threats. Evaluation of the proposed HIDS is conducted using the Bot-IoT dataset, encompassing legal IoT network traffic and diverse attack types. Findings indicate that, in comparison to SIDS and AIDS approaches, the proposed HIDS exhibits superior DR and a reduced percentage of FPs, affirming its efficacy in identifying known intrusions and zero-day threats.

An innovative and scalable HIDS has been introduced [46], constructed on Spark ML and the Convolutional-Long Short-Term Memory (Conv-LSTM) network. This development aims to surmount the limitations of conventional IDS, particularly their lack of flexibility and scalability, while concurrently enhancing accuracy. The HIDS employs a two-stage architecture, with the first stage incorporating an anomaly detection module built on Spark ML. In the second stage, a Conv-LSTM network-based abuse detection module operates as a misuse detection component capable of addressing both global and local latent threat signatures. During 10-fold cross-validation testing, the proposed HIDS outperformed state-of-the-art methods. It demonstrated a remarkable ability to accurately identify network misuses in 97.29% of cases, as evaluated on diverse baseline models using the ISCX-UNB dataset.
For the purpose of detecting DDoS attacks suggested in this work [47], a hybrid detection system called HIDS was suggested. The suggested detection system employs both AIDS and SIDS independently but together, using the results of both detectors to improve overall detection accuracy. To evaluate the detection performance of HIDS, two different datasets were applied to the proposed system.

The study discovered that the suggested HIDS outperforms systems based on non-hybrid detection.

Singh in [48] This paper introduced an innovative Advanced Hybrid Intrusion Detection System (AHIDS) designed to autonomously identify attacks on wireless sensor networks (WSNs). AHIDS adopts a cluster-based architecture incorporating an enhanced Low Energy Adaptive Clustering Hierarchy (LEACH) protocol, specifically geared towards reducing the energy consumption of sensor nodes. The detection system integrates both anomaly detection and misuse detection, leveraging imprecise rule sets in conjunction with a multilayer perceptron neural network. The incorporation of feedforward and Backpropagation Neural Networks aids in consolidating detection results and categorizing assailants into distinct types, such as Sybil attack (SA), wormhole attack (WA), and welcome deluge attack (WDA). Specialized algorithms, namely the Advanced SA detection algorithm for SA, and the Wormhole Resistant Hybrid Technique for WA, were implemented. WDA detection utilizes signal strength and distance parameters. In an experimental analysis involving a node set, 13.33% of nodes were identified as deviant, and the system demonstrated high rates of DR for various attacks, with SA detected at 99.40%, WDA at 98.20%, and WA at 99.20%.

In order to enhance the security capabilities of a smart home system and identify potential threats from network activities, devices, sensors, or users, the study described in [49], delves into the analysis of the publicly available CSE-CIC-IDS2018 dataset. The focus is on uncovering anomalous activities within the smart home environment, employing a two-tiered system. Utilizing Dataset2018, the study deploys multiple ML classification models, including RF, XGboost, and DT. The algorithms are trained within the decision layer, and experimental results revealed that each model attains a distinct level of accuracy in identifying potential anomalies indicative of attacks. The promising accuracy levels achieved by the approach proposed in this paper suggest its practical viability for implementation in a smart home security system.

To tackle the challenge of data imbalance [50], an efficient hybrid network-based Intrusion Detection System (HNIDS) model was developed, employing enhanced genetic algorithm and particle swarm optimisation (EGA-PSO), along with improved

random forest (IRF) methods. In the initial phase, the proposed HNIDS utilized hybrid EGA-PSO methods to augment minor data samples, thereby creating a balanced dataset for more accurate learning of attributes from smaller samples. Within the proposed HNIDS, a Particle Swarm Optimisation (PSO) method enhances the vector, while GA is augmented with a multi-objective function. This function selects optimal features, yielding improved fitness outcomes, which, in turn, helped to identify essential features, minimize dimensions, enhance the TP rate, and lower the FP rate. In the subsequent phase, an IRF process eliminated less significant attributes, integrates a list of DTs iteratively, monitors classifier performance, and addresses overfitting concerns. The performance of the proposed method is evaluated against existing ML methods using benchmark datasets. Experimental results demonstrated that the proposed HNIDS method achieves an accuracy of 98.979% in BCC and 88.149% in MCC for the NSL-KDD dataset. This outperforms other ML methods, including SVM, RF, LR, NB, LDA, and CART.

Khraisat in [51] This paper introduced a HIDS by amalgamating the C5 DT classifier and the OC-SVM. The HIDS design incorporated the strengths of SIDS and AIDS. The SIDS component is formulated based on the C5.0 DT classifier, while the AIDS component is built on the OC-SVM. This integrated framework aims to proficiently identify both well-known intrusions and zero-day attacks, achieving high detection accuracy coupled with low false alarm rates. The proposed HIDS undergoes evaluation using benchmark datasets, specifically the NSL-KDD and Australian Defence Force Academy (ADFA) datasets. The study outcomes indicated an enhanced efficacy of HIDS compared to both SIDS and AIDS in terms of detection rate and minimal false alarm rates.

In this chapter, [24] discussed the critical role of ML and data mining algorithms in the development of IDSs. These systems were categorised into two main types: misuse detection, which focused on identifying known attack signatures by matching network activity patterns, and anomaly detection, which highlighted deviations from established normal system behaviour. The chapter provided a comprehensive exploration of these approaches, shedding light on their strengths and limitations in the ever-evolving field of cybersecurity. Moreover, the chapter investigated the integration of these two approaches through the emergence of hybrid detection systems. These systems aimed to harness the advantages of both misuse and anomaly detection to create a

more resilient and adaptable defence mechanism against a broad spectrum of cyber threats. As the chapter unfolded, it became evident that ongoing research and innovation in the form of advanced algorithms were imperative to keep IDSs proactive and responsive to the dynamic tactics employed by malicious actors in the digital realm. In summary, the chapter offered valuable insights into the state of ID, paving the way for future developments in the field of cybersecurity.

2.2. TREE-BASED INTRUSION DETECTION MODELLING

Al-Omari in [6] introduced an intelligent ID model aimed at predicting and detecting intrusions in cyberspace. The model was designed using DT concepts, considering the ranking of security features. It was applied to an authentic dataset for NID systems. Furthermore, validation was performed based on predefined performance evaluation metrics, including accuracy, precision, recall, and F-score. The experimental results revealed that our tree-based ID model efficiently detected and predicted cyber-attacks while reducing the computational complexity compared to other traditional ML techniques.

Sarker in [2] presented introduced an ID tree (‘IntruDTree’) ML-based security model that initially considers the ranking of security features based on their importance. Subsequently, it constructs a tree-based generalized ID model using the selected essential features. This model proves effective not only in terms of prediction accuracy for unseen test cases but also in minimizing the computational complexity by reducing the feature dimensions. The efficacy of the ‘IntruDTree’ model was evaluated through experiments on cybersecurity datasets, computing accuracy, recall, F-score, precision, and ROC values. The results of the ‘IntruDTree’ model were compared with several popular traditional ML methods, including the NB classifier, LR, SVM, and KNN, to analyse the effectiveness of the resulting security model.

Ingre in [52] This paper introduced a DT-based IDS for the NSL-KDD dataset. The presented approach incorporates the correlation feature selection (CFS) subset evaluation method for feature selection, aiming to enhance the prediction efficacy of the DT-based IDS. Performance evaluation is conducted both before and after feature selection, considering both five-class classification and binary-class classification

scenarios. The obtained results are compared and analysed in relation to other reported techniques. The analysis reveals that the proposed DT-based IDS achieves high DR and accuracy. Notably, the aggregate result for binary-class classification surpasses that of the five-class classification for the dataset.

TekIn in [53] This study introduced an intelligent IDS tailored for IoT devices. The developed intelligent IDS for IoT devices are based on an extensive attack dataset comprising 3,668,443 observations. Unlike previous work with a binary classification focus, this research addresses the classification of assault varieties, employing nine categories. To establish a prompt responsive IDS model, a DT classifier is employed. The proposed model achieves a classification accuracy of 97.43% on this dataset through a 10-fold cross-validation. This high accuracy rate serves as a clear demonstration of the classification prowess of the proposed IDS model for IoT devices The utilization of autonomous vehicles (AV) stands out as a promising technology within Intelligent Transportation Systems (ITSs), offering the potential to enhance safety and transportation efficiency [54]. The integration of Vehicle-to-everything (V2X) technology facilitates communication between vehicles and other infrastructures. Despite these advancements, both AVs and the Internet of Vehicles (IoV) remain susceptible to various cyber-attacks, including denial of service, deception, and monitoring attacks. In the context of this paper [54], an intelligent IDS based on tree-structure ML models is proposed. Results derived from the implementation of the suggested IDS on standard datasets demonstrate the system's capability to identify various cyber-attacks in AV networks. Moreover, the incorporation of ensemble learning and feature selection approaches within the proposed system enables it to achieve a high detection rate while concurrently maintaining low computational costs.

Given the simplicity of automated data collection and the subsequent exponential growth in the scale of collected data on network traffic and activities, intrusion analysis has become increasingly complex [55]. To address this challenge, the study explores ensemble methods in the context of big data. Despite being more intricate and requiring additional computation, literature suggests that ensemble methods can yield higher accuracy than single classifiers in various large-scale data classification scenarios. Investigating how ensemble approaches perform in IDS is particularly

intriguing. This study introduces a tree-based stacking ensemble technique (SET) and assesses the effectiveness of the proposed model on two intrusion datasets (NSL-KDD and UNSW-NB15). Additionally, the enhanced study incorporates feature selection techniques to identify the most relevant features for the proposed SET. A comprehensive performance analysis demonstrates that the proposed model excels in identifying both normal and anomalous traffic in the network when compared to other existing IDS models. This underscores the potential of the proposed system for enhancing cybersecurity in the IoT and large-scale networks.

To address network security problems [56], this study proposed an ID method based on the Decision Tree-Recursive Feature Elimination (DT-RFE) feature in ensemble learning. Initially, the study introduced a data processing method using the DT-Based Recursive Elimination Algorithm to select features and reduce feature dimensions. This step aimed to eliminate redundant and uncorrelated data from the dataset, enhancing resource utilization and reducing time complexity. In the paper, a stacking ensemble learning algorithm was considered, combining Decision Tree (DT) with Recursive Feature Elimination methods. A series of comparison experiments conducted through cross-validation on the KDD CUP'99 and NSL-KDD datasets revealed that the DT-RFE and Stacking-based approach significantly improved the performance of the IDS. The accuracy for all types of features surpassed 99%, with the exception of U2R accuracy, which remained at 98%.

2.3. CONCLUSION

This chapter discussed some of the work done in the literature focussing on misuse detection technique, anomaly-based detection technique, hybrid detection technique and tree-based ID modelling. Most of the studies reviewed emphasised the importance of feature selection and ranking before deploying ML algorithms, the high false alarm rates produced by traditional IDSs, and their incompetency of to detect new unknown attacks. To achieve proper detection accuracy when detecting the network intrusion, to deal with the complex cyber security data which sometimes impossible to analyse, to identify model that detect network attacks in a timely manner, most of the studies proposed the use of ML algorithms.

2.4. SUMMARY

The studies discussed in this chapter offered insights into innovative ID methodologies within the field of cybersecurity. The first set of studies examined various ID techniques, including misuse-based and anomaly-based approaches. Researchers proposed self-taught learning and the MAPE-K framework to create autonomous misuse-based IDS. These methods aimed to enhance attack detection while minimising false alarms, particularly in situations with fluctuating network conditions. Anomaly-based detection methods, on the other hand, explored the use of RNNs and DNN to develop effective NID models. These studies underline the importance of evolving ID techniques to combat evolving cyber threats.

In contrast, other studies featured hybrid ID techniques that combine the strengths of misuse detection and anomaly detection systems. These approaches aim to improve detection accuracy while reducing false alarms, especially in identifying known and unknown attacks. The hybrid systems incorporated feature selection and ensemble methods to enhance their performance, presenting a comprehensive approach to ID. The studies also demonstrated how tree-based ID models, including decision tree-based algorithms, can offer efficient ways to manage security threats. Feature selection and ensemble techniques were used in conjunction with tree-based models to enhance their detection capabilities.

In both sets of studies, there is a common thread emphasising the need for continuous research and innovation in the domain of ID. These investigations showcase the significance of adapting to the dynamic strategies employed by malicious actors in the digital landscape. Researchers are actively developing and refining ID models, to provide more robust and adaptive cybersecurity measures against an ever-evolving spectrum of cyber threats.

CHAPTER 3 – METHODOLOGY

3. INTRODUCTION

The identification of key tools and ML algorithms to be incorporated in the development of ID models is critical. Therefore, it is necessary to understand which tools support the development of these algorithms. This methodology chapter outlines the procedures or processes used to find, select, process, and analyse the information about the study. In this chapter we critically examined the overall validity and reliability of the study. We illustrated how the problems and objectives were formulated and how the results were presented from the data collected during the study. It also outlined how our results were obtained in accordance with the objectives of the study. The progression of this chapter is designed to unravel the inner workings of the study and reveal the roadmap that will guide the execution of the study. The chapter begins with an exposition of the study's workflow, offering a main view of the journey that will be undertaken. It then explores the collection of research tools, unveiling the instruments that will be employed to gather, process, and analyse data. Following this, the dataset's background and description are presented, providing crucial context for the subsequent phases. The pivotal stages of data cleaning and exploratory analysis are explored in depth, shedding light on how the dataset’s integrity is preserved and insights are extracted.

The chapter proceeds to discuss data preprocessing and standardisation, explaining the methods to ensure data uniformity and suitability for analysis. Feature selection and ranking strategies are accurately detailed, underscoring the importance of refining the most relevant attributes for ID. The heart of this chapter lies in the presentation of tree-based ID modelling and classification algorithms. The chapter culminates in a full discussion of performance metrics, providing a framework for assessing the effectiveness of the models. Lastly, a conclusion and summary that summarises the key insights garnered in this chapter are discussed.

3.1. THE STUDY WORKFLOW

The study workflow represented the sequence of activities or processes required to complete the study. In our proposed study, the ID model comprised three primary modules. The first module encompassed three processes: data exploration, data preprocessing and standardization, and feature evaluation and selection. These

processes played a crucial role in constructing our tree-based ID approach based on hybrid feature ranking and selection. The last two modules focused on model training and testing, with the aim of constructing a classification model capable of detecting intrusions in cyberspace. Figure 3.1 illustrates our proposed model, and each stage of the model was discussed in detail in the following sections.

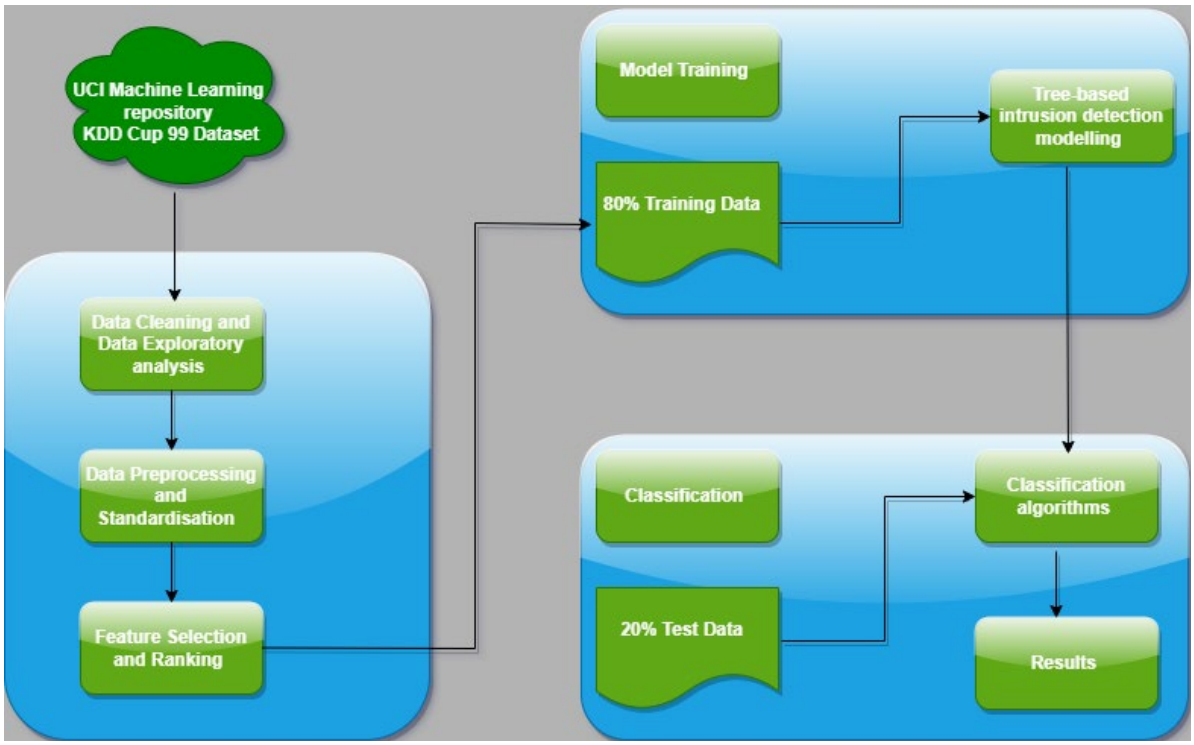


Figure 3.1 Study the overall workflow

3.2. RESEARCH TOOLS

To support ID models and their exploratory analysis, a modern open-source ML tool should provide a user-friendly interface. In addition, the interfaces should effectively support data and model visualisation. The application should also offer data analysis tools for the interactive search of any hidden and interesting data patterns and allows interactive exploration of inferred models. This study considers a tool called Google Colaboratory to conduct the experiments. The tool was chosen because it is a free online cloud-based jupyter notebook environment hosted by Google that allows training of ML and deep learning models. The tool requires a good network connection

and Google account and does not require any installation. The Python codes were written directly from the browser, and the files are stored in the google drive. The Google Collaboration was used for exploratory data analysis and visualisation. The experiments were conducted in a Python programming environment. Figure 3.2 shows the Python libraries that were incorporated in this study:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
from tensorflow.keras.utils import get_file
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
```

Figure 3.2 Python Libraries

i. NumPy

A NumPy array is a multidimensional, uniform collection of items (each element takes up the same amount of memory). An array is defined by the kind of items it contains, as well as its form. For example, it can be represented as an array of $m \times n$ integers, such as floating point or complex numbers. NumPy arrays, unlike matrices, may have up to 32 dimensions; they may also include additional types of elements (or even combinations of elements), such as Booleans or dates [57][58][59]. This library was considered in this study because it provides support for efficient array operations, which are must faster than traditional Python lists. Numpy allows the researcher to work with arrays of any dimensionality, which is particularly useful for scientific computing tasks involving matrices, vectors, and multidimensional data. Numpy is a fundamental building block for many scientific computing libraries in Python, such as SciPy, Pandas, and Scikit-learn, which relies on Numpy arrays as the data container, allowing for seamless integration and efficient computations across different libraries.

ii. Pandas

Pandas is a freely available library created primarily for managing relational or labelled data with simplicity and clarity. It provides various data structures and techniques for handling numerical data and time series. Derived from the NumPy library, Pandas is not only fast but also excels in performance, improving user productivity [60][59]. The study considered this library because it allows the researcher to store and manipulate data in a DataFrame, which provides intuitive indexing, column labelling, and easy slicing and filtering of data. Pandas provides a wide range of functions and methods for data cleaning and preprocessing tasks, powerful tools for data exploration and analysis, simple and convenient functions for data visualisation, and can integrate with other libraries.

iii. OS

The Python OS module empowers users to establish interactions with their operating systems. It encompasses a range of practical features facilitating the execution of tasks based on the operating system and the retrieval of relevant OS-related information. Python's fundamental utility modules encompass functionalities related to the operating system [61]. This module is useful in the study because it provides essential functionalities for file and directory operations, path manipulation, process management, environment variables, and access to operating system-related information.

iv. Tensorflow

TensorFlow, an open-source toolkit, was crafted for numerical computation and extensive ML applications. Originating from the Google Brain team, it was publicly released in 2015. TensorFlow amalgamates a multitude of ML and deep learning models, commonly referred to as neural networks, rendering them accessible through conventional programming metaphors. It furnishes a user-friendly front-end API for application development in Python or JavaScript, with the underlying execution implemented in high-performance C++ [62]. This library was considered because it is a versatile and powerful library for ML and deep learning tasks. Provides tools and

abstractions for building and training neural networks, supports GPU acceleration, and offers flexibility for defining complex models.

v. Keras

Keras, an open-source Python library, is both free and potent for constructing and analysing deep learning models. Integrated seamlessly into the TensorFlow library, it allows users to create and train neural network models with remarkable ease, requiring only a few lines of code.[62]. Keras stands out as a robust and user-friendly deep learning library, which streamlines the complexities associated with constructing and training neural networks. Its intuitive API, modular design, diverse applications, and seamless integration with deep learning models have contributed to its widespread adoption and popularity in this study.

vi. Utils

Python Utils is a collection of simple Python methods and classes that shorten and simplify common patterns. It is by no means a complete collection, but it has served me well in the past and will continue to do so [63]. The study considered this library because it provides code reusability, simplification, standardisation, error handling, performance optimisation, cross-platform compatibility, and integration with other libraries. This library helped the study improve code organisation, maintainability, and development efficiency by providing commonly used functionalities and solutions to recurring programming problems.

vii. Matplotlib

Python Matplotlib is a cross-platform package for visualising and charting graphical data. Developers can incorporate plots into GUI systems using matplotlib application programming interfaces [64][58][59]. It is because of its versatility, customisation options, support for various plot types, publication quality output, integration with data analysis libraries, and interactive plotting capabilities that make it a popular choice for data visualisation and scientific plotting tasks in this study.

viii. Pyplot

An open source interactive plotting toolkit for Python, Plotly offers more than 40 different graph types for a variety of statistical, financial, geographic, scientific, and

three-dimensional use cases [58]. The study considered this library because it provides a user-friendly and versatile interface to create plots and visualisations. It offers quick and easy plotting capabilities, customisation options, interactive plotting, publication quality output, and seamless integration with Matplotlib.

ix. Seaborn

Seaborn is a graphic visualisation library based on Matplotlib's core settings. It gives customers access to some of the most widely provided data visualisation processes with specific data visualisation needs, such as mapping colour to a variable or employing faceting requirements all over the world. It makes seaborn more integrated when dealing with Pandas DataFrames [59]. The study considered this library because it provides improved visual aesthetics, simplified syntax, statistical visualisation capabilities, integration with Pandas, flexible customisation options, and support for data exploration.

x. Sklearn

Python's Sklearn library is used for machine learning. It is a collection of effective tools for data mining and analysis, to be more precise. The framework is built on top of several well-known Python programmes, including Matplotlib, SciPy, and NumPy [58][65]. It is because of its versatility, easy and consistent interface, broad range of algorithms, preprocessing capabilities, model evaluation and selection tools, integration with other libraries, and extensive documentation that made it a go-to choose for implementing ML models in this study. Sklearn can efficiently develop and deploy ML solutions for a variety of real-world applications.

xi. Preprocessing

Only information about how to process data, that is, how to turn one data type into another (for example, from an (audio-)signal to a spectrogram). To construct transformation chains, each data class has a matching processor class that implements this transformation. This allows for quick and easy conversion of algorithm prototypes to callable processing pipelines [66]**.** The pre-processing module in sklearn provides a wide range of pre-processing techniques for data preparation in ML tasks. It was considered by the study because it offers functions for data normalisation and scaling, handling missing data, encoding categorical variables, dimensionality

reduction, feature extraction and transformation, and building data preprocessing workflows. By leveraging the capabilities of the pre-processing module, you can effectively pre-process your data and improve the performance of ML models.

xii. StandardScaler

The mean is removed and each feature/variable is scaled to unit variance using StandardScaler. This procedure is carried out in a feature-by-feature manner. Because it includes estimating the empirical mean and standard deviation of each feature, StandardScaler can be affected by outliers (assuming they exist in the dataset) [67]. We considered standardising the features using this library because it improves model performance, handles outliers robustly, integrates with sklearn pipelines, offers efficient computation, provides an easy-to-use interface, and enhances interpretability. The StandardScaler is a valuable tool for ensuring that numerical features are appropriately scaled and ready for modelling with various ML algorithms.

3.3. BACKGROUND AND DESCRIPTION OF THE DATASET

The data set considered by the study was extracted from the https://archive.ics.uci.edu/dataset/130/kdd+cup+1999+data UCI ML Repository, the data set was used for the 3rd International Knowledge Discovery and Data Mining Tools Event, which took place together with KDD-99, the 5th International Seminar on Knowledge Discovery and Data Mining. The objective of the event was to create a network intrusion detector, a prediction algorithm that could tell the difference between good connections and undesirable ones, such as invasions or attacks. In this database, a standard set of auditable data is contained, including a variety of simulated intrusions into a military network environment.

The data set was generated by collecting network traffic data from a simulated environment that emulated a military network. Data were generated using a variety of network-based attacks and normal network activities. To generate the dataset, a network simulation environment was set up using DARPA's ID Evaluation Dataset (1998). The simulated environment consisted of multiple machines, including attackers and targets, connected through a network infrastructure.

Network traffic was monitored and recorded during the simulation. Various features of the network traffic were captured and recorded during the simulation. These features included information such as the protocol used, source and destination IP addresses, source and destination ports, packet flags, duration of the connection, number of bytes transferred, etc. These features provided the basis for analysing and classifying network traffic. Here is a breakdown of the column names and the types of data to which they correspond:

1. “duration”: The length of time the connection was active.
2. “protocol_type”: The protocol used in the connection (e.g., TCP, UDP, ICMP).
3. “service”: The type of service or application associated with the connection (eg, http, smtp, ftp).
4. “flag”: Flags associated with the connection, indicating its status (eg, SF for “normal” or S0 for ‘Connection attempt’).
5. “src_bytes”: The number of source bytes in the connection.
6. “dst_bytes”: The number of destination bytes in the connection.
7. “land”: Indicates whether the connection is from/to the same host/port (binary, 0 or 1).
8. “wrong_fragment”: The number of "wrong" fragments in the connection.
9. “urgent”: The number of urgent packets in the connection.
10. “hot”: The number of "hot" indicators in the connection.
11. “num_failed_logins”: The number of failed log-in attempts prior to the connection.
12. “logged_in”: Indicates whether the user is logged in (binary, 0 or 1).
13. “num_compromised”: The number of compromised conditions in the connection.
14. “root_shell”: Indicates whether root shell access was obtained (binary, 0 or 1).
15. “su_attempted”: Indicates whether a ‘su_root’ command was attempted (binary, 0 or 1).
16. “num_root”: The number of root accesses in the connection.
17. “num_file_creations”: The number of file creations in the connection.
18. “num_shells”: The number of shell prompts in the connection.
19. “num_access_files”: The number of operations in the access control files in the connection.
20. “num_outbound_cmds”: The number of outbound commands in the connection.

21. “is_host_login”: Indicates whether it is a host login (binary, 0 or 1).
22. “is_guest_login”: Indicates whether it is a guest login (binary, 0 or 1).

And so on, the list continues with additional attributes that describe various aspects of network connections and potential intrusions. Network traffic data was labelled to indicate whether each network connection represented normal or attack behaviour. Attacks were classified into four main types: Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R), and Probing. Each network connection was labelled with one of these categories or classified as normal.

There are 494021 instances in this collection, each with one of the 42 properties. Figure 3.3 illustrates the types of attack contained in the dataset that are found in the predicted attribute.

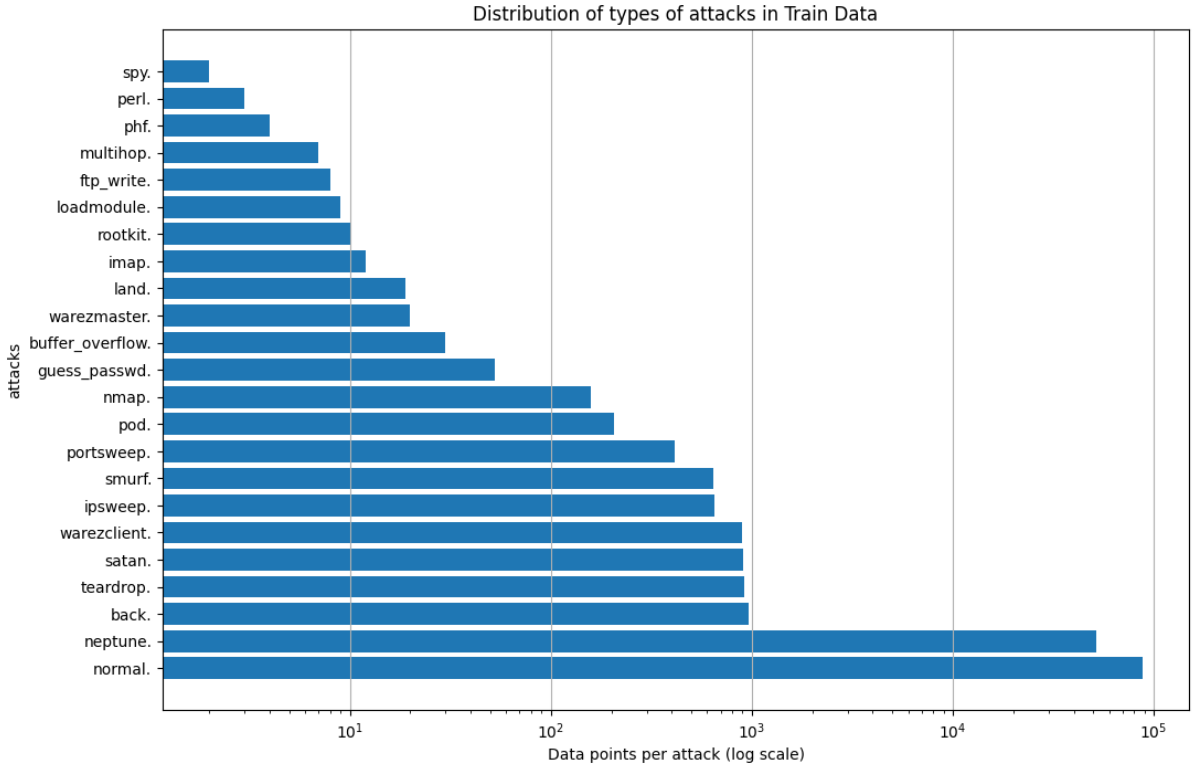


Figure 3.3 Distribution of types of attack in Train data

Since 1999, KDD’99 [68] has been the most widely used data set for the evaluation of anomaly detection methodologies. This data set was prepared by [69], and was built on the data obtained in the DARPA ‘98 IDS evaluation programme [70]. DARPA’98 is about 4 gigabytes of compressed raw (binary) tcpdump data of 7 weeks of network traffic, which can be processed into about 5 million connection records, each with

about 100 bytes. The two weeks of test data have around 2 million connection records. The simulated attacks fell into one of the following four categories:

1. Denial of Service Attack (DoS): is an attack in which the perpetrator makes some computing or memory resource too active or too full to handle legitimate requests, or denies legitimate users access to a machine.
2. User-to-Root Attack (U2R): is a kind of hack in which the attacker gains access to a regular user account on the system (through password sniffing, a dictionary attack, or social engineering) and then exploits a vulnerability to get root access to the system.
3. Remote to Local Attack (R2L): occurs when an attacker with the ability to transmit packets to a computer across a network but no account on that system leverages a vulnerability to acquire local access as that machine's user.
4. Probing attack: An effort to obtain information about a network of computers to defeat its security mechanisms.

It is critical to note that the test data does not come from the same probability distribution as the training data and contains certain attack types that were not included in the training data, making the job more realistic. According to some intrusion specialists, most new assaults are variations of established attacks, and the signature of known attacks may be sufficient to detect novel versions. There are a total of 24 types of training attack in the datasets, with an additional 14 types in the test data alone. The names of the training attack types and detailed descriptions are mentioned in[71]. The KDD’99 features can be divided into three categories:

1. Basic features: This category contains all the features that may be derived from a TCP/IP connection. Most of these attributes result in an implicit detection delay.
2. This category contains characteristics that are calculated with regard to a window interval and is separated into two groups:
    a) "Same host" features: evaluate just the connections made in the last two seconds that have the same destination host as the current connection, and collect statistics on protocol behaviour, service, and so on.

b) “Same-service” features: only look at connections made in the last two seconds that have the same service as the current connection.

The two sorts of "traffic" characteristics discussed above are known as time-based. However, there are other slow probing attacks that use a considerably longer time period than 2 seconds to check the hosts (or ports), for example, once every minute. As a consequence, these attacks do not generate intrusion patterns with a 2 second time frame. To address this issue, the "same host" and "same service" characteristics are computed again, but this time using a connection window of 100 connections rather than a time frame of 2 seconds. These are known as connection-based traffic characteristics.

3. Content features: R2L and U2R assaults, unlike other DoS and Probing attacks, do not have any intrusive frequent sequential patterns. This is due to the fact that DoS and probing attacks entail several connections to some host(s) in a very short period of time, while R2L and U2R attacks are encoded in the data parts of the packets and often involve only a single connection. To identify these types of assaults, we need certain attributes that allow us to search for abnormal activity in the data section, such as the number of unsuccessful login attempts.

3.4. DATA CLEANING AND EXPLORATORY ANALYSIS

The quality and integrity of data are considered the most important resource in data mining and ML, as they can significantly alter the prediction accuracy of any proposed prediction algorithms. Thus, before this scientific experiment can be performed, we have to clean the data and perform its exploratory data analysis.

3.4.1. DATA CLEANING

Errors, inconsistencies, missing numbers, outliers, and noise are common in real-world data. By resolving these concerns, cleaning the data helps increase its quality. High-quality data is essential to train accurate and reliable ML models. This section ensures that the data considered by the study are in the correct format. Data cleaning entails removing missing values and duplicates because there is a possibility of reaching wrong conclusions or generating incorrect prediction accuracies based on

the inconsistencies of the dataset. Figure 3.4.1 illustrates all the functions used to perform data cleaning in the dataset.

```
# Checking for DUPLICATE values
df.drop_duplicates(keep='first', inplace = True)
# Checking for NULL values
print('='*35)
print('Null values in dataset are',len(df[df.isnull().any(1)]))
print('='*35)
# For now, just drop NA's (rows with missing values)
df.dropna(inplace=True,axis=1)

print('='*35)
print("Read {} rows.".format(len(df)))
print('='*35)

# stored the data into a pickle file so we can load through
df.to_pickle('df.pkl')
```

Figure 3.4.1 Data-cleansing Functions

i. drop_duplicates() the functions help in removing the duplicates from the Pandas Dataframe in Python. The parameter “keep” controls how to consider a duplicate value and has a default distinct value 'first' which considers the first value as unique and the rest of the same value as duplicates. The ‘Inplace’ parameter is a Boolean value that removes rows with duplicates if true.

ii. The Isnull() function returns a DataFrame object where all the values are replaced with a Boolean True for Null values, and otherwise false.

3.4.2. DATA EXPLORATORY ANALYSIS

The KDD99 dataset is a popular benchmark in the area of ID. It comprises network traffic statistics that can be utilised to distinguish between normal and intrusive network connections. Understanding the structure of the dataset, studying the characteristics, and obtaining insights into the data distribution are all part of performing an exploratory analysis on it. The first step is to understand the structures and composition of the dataset, that is, to inspect the number of rows and features present in the dataset. The second step will involve feature exploration, in which we examine individual features in the dataset, which will assist us in understanding the data types (numeric, categorical) and the meaning of each feature. The third step is to obtain a statistical summary of the data by calculating basic statistical measures such as mean and standard deviation, which will provide a high-level overview of the data distributions and helps identify potential outliers or anomalies. The fourth step is to analyse the

class distribution by determining the proportion of each class to identify any class imbalance issues, since the dataset includes the target column. Lastly, visualise the relationships between different features by exploring feature correlations using techniques like scatter plots or correlation matrix, which will help identify any significant relationships in the dataset.

3.5. DATA PRE-PROCESSING AND STANDARDISATION

Pre-processing data is an important stage in data analysis and ML jobs. It is the process of preparing and converting raw data into a clean, consistent, and appropriate format for future analysis or modelling. The accuracy and quality of the data have a direct influence on the performance and success of any data-driven initiative. Data preparation includes a variety of approaches and procedures for dealing with difficulties in raw data, such as missing values, outliers, noise, inconsistencies, and formatting inconsistencies. In this section, we will use two data pre-processing methods, get_dummies method, and the StandardScaler to scale numeric values.

i. The term "get dummies" refers to a technique for turning categorical variables into numerical representations, which is often used in data preparation. It is often used in ML projects when dealing with categorical information. ML algorithms often require numerical input when dealing with categorical variables. The "get dummies" approach, also known as one-hot encoding or binary encoding, converts each categorical variable into a binary column. Generate a new binary column for each distinct category, with a value of 1 indicating the existence of that category and a value of 0 indicating its absence.

ii. StandardScaler is a data preparation tool that is used to normalise numerical characteristics in ML applications. It is especially beneficial when the characteristics of the dataset have distinct sizes or units. Standardising the data guarantees that each feature has a mean of 0 and a standard deviation of 1, resulting in a normal (Gaussian) distribution. The formula used for standardisation is as follows:

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

The data set was then split into two subfolders, namely, training and testing using a Python library called sklearn. The study considered an 80/20 split. This allows 80% of the data to be used for training and 20% to be used to test the model.

3.6. FEATURE SELECTION AND RANKING

When using supervised ML techniques such as DT, it is essential to select a strategy that is appropriate for identifying attributes that have a substantial impact on the decision-making process. Information gain and the Gini index are two methodologies that are frequently used in this context [6]. The former suggests that a particular DT is built from the feature with the highest information gain as its base. According to the latter, the feature with the lower Gini index must be picked in a binary split (two choices for individual node) [6]. To achieve our goal, we propose a feature-ranking method that uses the hybrid of Gini index and information gain technique to categorise the impurity of the features and then ranks them based on the Gini impurity (i.e., entropy) and most informative features in the dataset before structuring our DT. By accomplishing this goal, we can then construct our tree-based ID modelling using the features with the lowest Gini index and the uppermost information gain.

3.6.1. GINI INDEX

The Gini index is a statistical measure that is used to assess the extent of income disparity within a population or the concentration of a given variable. It is also known as the Gini coefficient or Gini ratio. It is named after Corrado Gini, an Italian statistician who created the index in 1912.The Gini index goes from 0 to 1, with 0 representing perfect equality (everyone in the population has an equal share of the variable being measured) and 1 representing maximum inequality (one person has all the resources while the others have none).

$$Gini\ Index(GI) = 1 - \sum_{i=-}^{n} p(i)^2 \tag{3.2}$$

where n represents the number of classes in the target variable,

$p(i)\ represents$ the ratio of the predicted class divided by the total number of observations in a node.

The Gini index is significant important when dealing with the KDD99 dataset because it helps to select informative features, determining optimal splitting criteria, and evaluate the performance of classification models. It plays a crucial role in achieving accurate ID and improving the overall security of network systems.

3.6.2. INFORMATION GAIN

In DT algorithms, information gain is utilised to select the optimal feature to split on at each node. Calculates the decrease in entropy or impurity caused by dividing data according to a certain attribute. Entropy is a measure of the amount of impurity or disorder in a collection of instances. It depicts uncertainty or unpredictability in the distribution of class labels within a node in the context of classification problems. The following formula is used to compute the entropy of a node:

$$Entropy\ (E) = -\sum_{i=1}^{n} p_{i\log_2(p_i)} \tag{3.3}$$

Where p_i ,is the probability of randomly selecting an example in a class

$$Information\ Gain\ (IG) = E_{parent\_node} - E_{Child\_node} \tag{3.4}$$

Where E_{parent_node} ,is the entropy of the parent node

and E_{Child_node} is the average entropy of the child node,

Information gain is important in feature selection, DT algorithms, understanding feature relevance, model evaluation, and dimensionality reduction. In this study, it enabled us to identify and focus on the most informative features, leading to improved model performance and better insights into the data.

3.6.3. HYBRID SELECTION

This subsection outlines how our proposed feature selection method was performed. The proposed method was performed by combining Gini index and Information Gain, which will improve robustness, handling different characteristics, enhance feature selection, balance consideration of attributes, and increase the accuracy and generalisation. This technique provides for a fair assessment of both measures in the

decision-making process by combining weighted and normalised scores of the Gini index and information gain. The following formula is used to compute a hybrid selection:

$$hybrid\ selection = Information\ Gain + Gini\ Index \tag{3.5}$$

The study considered a hybrid selection of the Gini index and the information gain because they can provide additional benefits when DTs are constructed. While Gini index and information gain are both useful metrics for attribute selection, they have slightly different characteristics and may prioritise different features. Combining them in a hybrid approach helped the study harness the strengths of both metrics and improve the overall performance of the DT model. The combination of Gini index and information gain provided complementary evaluation, robust feature selection, and flexibility. This approach was applied following the steps mentioned steps:

i. Firstly, we computed the Gini index for each feature in our dataset, which measured the impurity or inequality within a set of classes.
ii. Secondly, we calculated the information gain for each feature in the dataset, which quantified how much information a feature provides to reduce the uncertainty or entropy of the target variable.
iii. Normalise the Gini index and information gain values to ensure that they are on the same scale. This step is important because the two metrics may have different ranges or magnitudes, and normalising them makes them comparable.
iv. Combine the normalised Gini index and information gain using a weighted average. For a weighted average approach, assign weights to each metric based on its relative importance or preference. Calculate the weighted average score for each feature by combining the normalised Gini index and information gain.
v. rank the features in descending order based on the combined scores. Features with higher combined scores are considered more relevant.
vi. Finally, we used the selected features to build the DT. The DT algorithm will utilise the features chosen during the splitting process to create a tree structure that maximises the predictive power and accuracy.

3.7. TREE-BASED INTRUSION DETECTION MODELLING

We may now construct a tree-based ID algorithm at this level after finishing the preceding phases. Our model's foundation is based on security features with smaller dimensions, which lessens the computational intricacy of the model. It is also constructed with the best security features, which can greatly increase the accuracy of its predictions. The root node to divide the dataset into smaller subsets must be found before we can start building the tree branches. The combination of Gini Index and Information, which were previously covered in this study, is used to achieve this. Our target attribute that decides whether an action is legitimate or malicious is labelled on the leaf node. The following Figure 3.7 illustrates how a DT algorithm is structured.

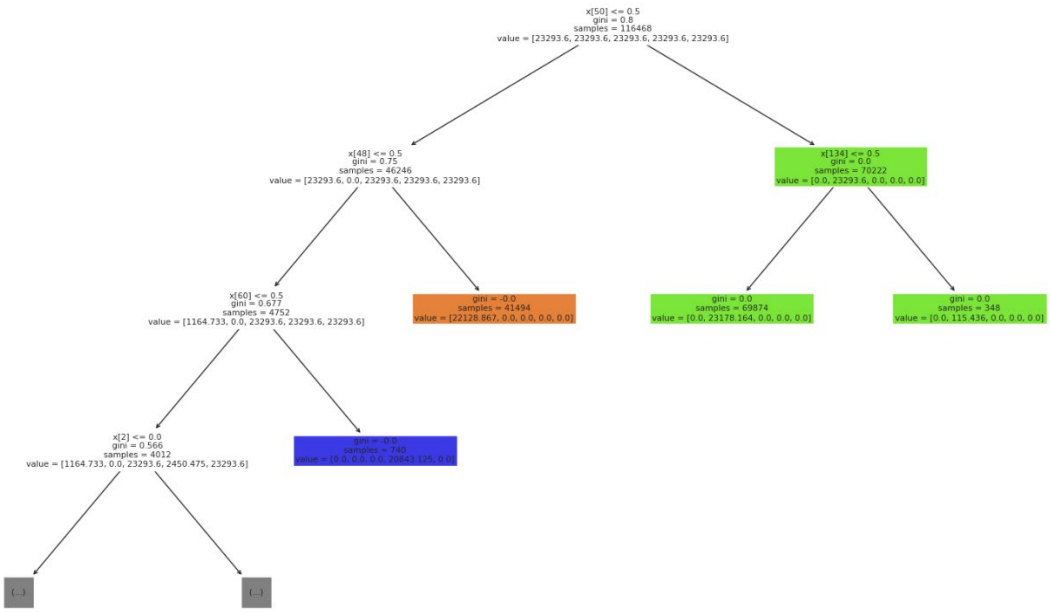


Figure 3.7 DT algorithm

A DT for classification consists of a tree structure with nodes representing decisions based on the feature values and leaves representing class labels. The DT recursively splits the data on feature values to reach a final classification decision on the leaves. The formula for decision tree classification can be summarised as follows:

> Starting from the root node, the tree asks questions based on the feature values. At each node, the algorithm selects a feature and a threshold for that feature, and the data is split into two or more child nodes based on the criteria.

The process continues recursively down the tree until a leaf node is reached. The class label at the leaf node represents the final classification decision for the input data point.

3.8. CLASSIFICATION ALGORITHMS

Classification algorithms are ML algorithms that use patterns or characteristics in the data to give predetermined class labels or categories to the incoming data. These algorithms are trained using a labelled training dataset, in which each data sample is assigned a known class label. The purpose of classification algorithms is to create a model that can correctly categorise fresh, previously unseen data into the right classifications.

3.8.1. SUPPORT VECTOR MACHINE

The Support Vector Machine (SVM) stands as a supervised ML algorithm adept at handling both classification and regression tasks. While it can be applied to regression problems, its primary strength lies in classification. The fundamental objective of the SVM algorithm is to identify the optimal hyperplane within an N-dimensional space, effectively segregating data points in the feature space into distinct classes. Hyperplanes are designed to maximize the separation between the closest points of different classes. The dimension of the hyperplane is contingent upon the number of features: for two input features, the hyperplane manifests as a line, and for three input features, it takes the form of a two-dimensional plane. Figure 3.8.1 Which was downloaded in [72], illustrates how SVM segregates data points or performs a classification between two classes A and B.

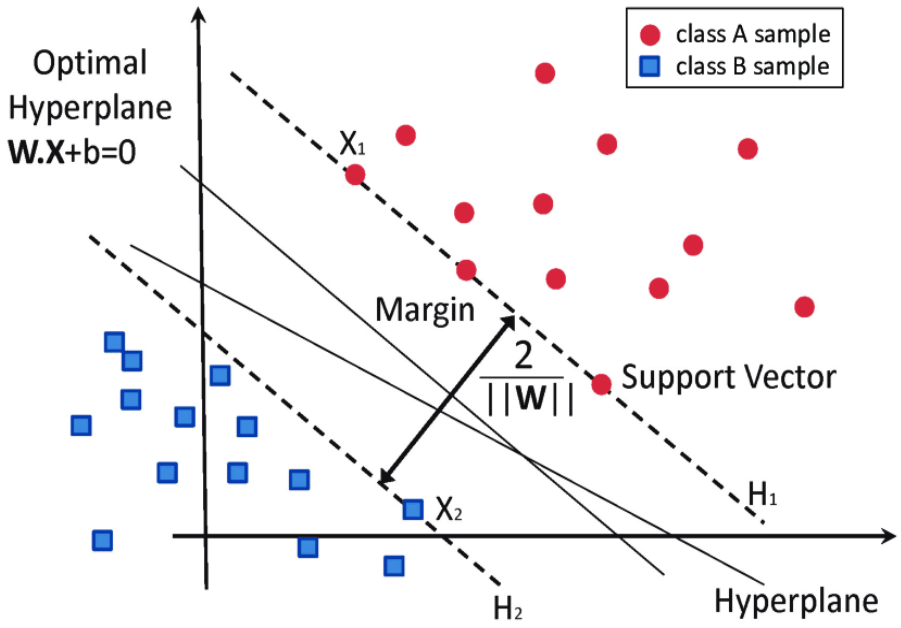


Figure 3.8.1 Classification of data by SVM

Given a set of training data points X_i and their corresponding class labels Y_i, where X_i is a feature vector and Y_i is either -1 or 1, the goal of SVM is to find a hyperplane (a linear decision boundary) that maximises the margin between the two classes. The equation of this hyperplane is represented as:

$$f(X) = w \cdot x + b \tag{3.6}$$

Where:

f(x) is the decision function for classifying a new data point x.

w is the weight vector (coefficients) of the hyperplane.

· Denotes the dot product.

b is the bias term (intercept).

In a binary classification scenario, the class of the new data point X is determined by the sign of $f(X)$. If $f(X)$ is greater than 0, X is classified as one class; if it is less than 0, X is classified as the other class. The objective of the SVM algorithm is to find the optimal weight vector w and bias term b that maximise the margin while ensuring that all training data points are correctly classified. This can be mathematically expressed as an optimisation problem with constraints. The most common form of the optimisation problem for SVM is the "soft margin" formulation, which allows for some misclassification while still maximising the margin. SVMs can also be applied with

kernel functions to handle non-linear classification problems by transforming the data into a higher-dimensional space. For a linear SVM, the kernel function is simply the dot product $K(x, x_i) = x^T x_i$. In the context of intrusion detection using the KDD99 dataset, we trained the SVM to distinguish between different attack classes. The decision function would then be formulated as:

$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i \, K(x, x_i) + b\right) \tag{3.7}$$

Here,

$f(x)$ is the decision function
N is the number of support vectors
α_i are the langrange multipliers
y_i is the class label (-1 or 1)
$K(x, x_i)$ is the kernel function that computes the dot product between x and x_i
b is the bias term.
$sign(.)$ is the sign of the function

For each class, we have a binary classification problem. The class labels y_i would be +1 for instances of the class you are interested in and -1 for instances of other classes. For example, for the "dos" class, we would set $y_i = +1$ for instances labelled as "dos" and $y_i = -1$ for instances labelled as other classes. Similarly, we would set $y_i = +1$ for instances of the "normal" class and $y_i = -1$ for instances of other classes, and so on for the "probe," "r2l," and "u2r" classes. The SVM would then learn the optimal hyperplane to separate instances of the target class from instances of other classes based on the features in the dataset

The sign function returns +1 if the expression inside the parentheses is positive, and -1 if it is negative. In the context of SVMs, the sign of the decision function determines the predicted class of the input data point. If $f(x)$ is positive, the data point is classified as belonging to one class (e.g., the positive class), and if it is negative, the data point is classified as belonging to the other class (e.g., the negative class). The magnitude of the output is not as important as its sign, as it is the sign that dictates the predicted class.

SVM Hyperparameters

In our pursuit of effective intrusion detection study using SVM, we have strategically employed the SVM model with specific parameter settings to enhance its discriminatory power. The chosen parameters, namely kernel = 'rbf', gamma = 0.1 and C = 1.0, play a critical role in shaping the SVM decision function and, consequently, its ability to differentiate between different attack classes in the KDD99 data set.

SVM with Radial Basis Function (RBF) Kernel:

The choice of the kernel of the radial basis function (kernel = 'rbf') is a fundamental decision that influences the capacity to handle non-linear relationships within the data. The RBF kernel is defined as:

$$K(x, x_i) = \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right) \tag{3.8}$$

Here, x and, x_i are data points, and σ is the kernel width parameter. The RBF kernel allows the SVM to map the input data into a larger space, facilitating the delineation of complex decision boundaries.

Gamma parameter:

The gamma parameter in the SVM model governs the width of the RBF kernel and influences the reach of each data point in the transformed space. A smaller gamma implies a larger kernel width, leading to a more generalised decision boundary. The formula for the RBF kernel (sigma) in terms of 'gamma' is:

$$\sigma = \sqrt{\frac{1}{2\gamma}} \tag{3.9}$$

A lower gamma encourages a smoother decision surface, potentially reducing the risk of overfitting, but may sacrifice precision.

Cost parameter (C):

The parameter C in the SVM model, often referred to as the regularisation parameter, balances the desire to achieve a minimal classification error against the preference for a simpler decision boundary. It is introduced in the SVM optimisation problem through the term: (3.10)

$$minimise\ \left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i\right)$$

Here, w represents the weights, ξ_i are slack variables, and N is the number of data points. The regularisation parameter 'C' controls the trade-off between achieving a low training error and minimising the complexity of the decision function.

We have configured our parameters as kernel= 'rbf' to accommodate nonlinear relationships, gamma=0.1 to strike a balance between model complexity and generalisation, and C = 1.0 to maintain a moderate regularisation strength, as mentioned above. This parameter configuration aims to ensure that the SVM is properly equipped to discern the intricate patterns inherent in the various attack classes within the KDD99 data set. The parameterized SVM stands as a robust tool for intrusion detection, and our careful selection of parameters reflects a judicious balance between model complexity, generalisation, and precision. The ongoing exploration and refinement of such parameter configurations contribute to the ongoing evolution of SVMs in enhancing the efficacy of intrusion detection systems in the face of dynamic and sophisticated cyber threats.

3.8.2. K-NEAREST NEIGHBOURS

K- KNN is a fundamental and crucial classification algorithm in the field of machine learning. It falls under supervised learning and is widely used for tasks such as pattern recognition, data mining, and ID. Its practicality is widespread due to its nonparametric nature. Unlike algorithms like GMM that assume a Gaussian distribution of data, KNN does not rely on any underlying assumptions about the data distribution. This makes it adaptable to real-life scenarios. In KNN, we utilise a set of previous data (referred to as training data) that categorise coordinates into distinct groups based on a particular attribute. Figure 3.8.2 illustrates how a table of data points containing two features can be assigned to a group by analysing the training set.

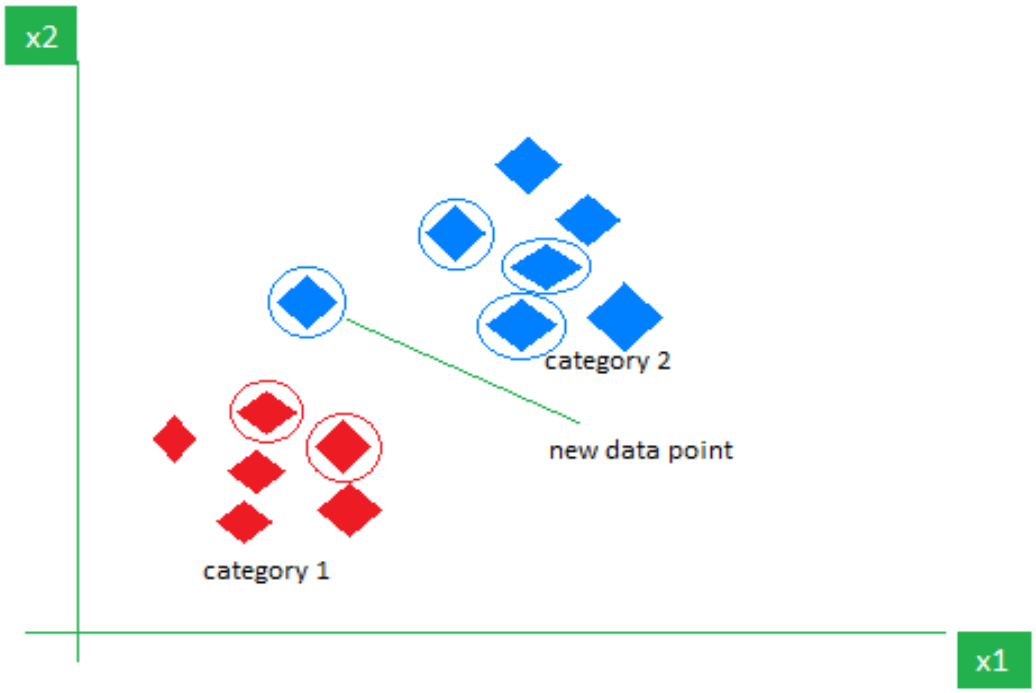


Figure 3.8.2 KNN Algorithm

Given a dataset with labelled data points:(X_i, Y_i), where X_i is a feature vector, and Y_i is the class label. To classify a new data point X, you find the k data points from the dataset that are closest to X in terms of a chosen distance metric, typically the Euclidean distance. The class of X is determined by taking a majority vote among the classes of its KNN:

$$Class(X) = mode(Y_i) \text{ for i in } N_i \tag{3.11}$$

Where:

$Class(X)$ represents the predicted class for the new data point X.
$mode(Y_i)$ calculates the most frequently occurring class among the neighbours.
N_i is the set of KNN of X.

This formula simplifies the classification process, where the class label to the new data point based on the majority class among its KNN. The choice of k and the distance metric can impact the accuracy and performance of the KNN classifier and should be selected based on the specific problem and dataset.

In the context of this study, we have employed the (KNN algorithm for intrusion detection on the KDD99 dataset, a benchmark in the realm of cybersecurity. The

KDD99 data set encompasses various attack classes, including “dos”, “normal”, “probe”, “r2l”, and “u2r”. Our focus lies on formulating the KNN algorithm to effectively discriminate between these distinct attack classes without employing specific training parameters. The KNN algorithm is a nonparametric and instance-based classification technique that assigns labels to data points based on the majority class among their k-nearest neighbours. The class label assigned to a given data point x is determined by a voting mechanism from its nearest neighbours. The KNN decision function can be expressed as:

$$f(x) = argmax_c \sum_{i=1}^{k} I(y_i = C) \tag{3.12}$$

Here,

$f(x)$ is the predicted class for the input data point x ,
$argmax_c$ denotes the class that maximises the sum,
y_i represents the class label of the $i - th$ nearest neighbour.
$I(.)$ is the indicator function that evaluates to 1 if the condition is true and 0 otherwise.

Class-Specific Formulation

In the context of intrusion detection, we tailored the KNN formulation to address each attack class individually. Let N_c be the number of instances in the class c, and K_c be the number of nearest neighbours considered for class c. The class-specific KNN decision function for “dos” class, for instance, was determined by:

$$f_{dos}(x) = argmax_c \sum_{i=1}^{k_{dos}} I(y_i = c) \tag{3.13}$$

Similarly, formulations are established for the “normal”, “probe”, “r2l”, and “u2r” classes, ensuring that the KNN algorithm appropriately adapts to the nuances of each attack category.

It should be noted that the KNN model in this study is trained without specific parameters, allowing it to adapt dynamically to the inherent structure. The absence of predefined parameters fosters a more flexible and data-driven approach, crucial in the context of intrusion detection, where the characteristics of attacks may vary

significantly. The KNN algorithm, formulated in a class-specific manner for the diverse attack categories within the KDD99 dataset, is a versatile tool in intrusion detection. The parameter-free training approach ensures adaptability, and the nuanced class-specific formulations contribute to the algorithm's ability to discern intricate patterns associated with different types of cyber threats. This study advances our understanding of KNN in intrusion detection, emphasising its applicability and effectiveness in the absence of explicit parameter tuning.

3.8.3. LOGISTIC REGRESSION

LR is a popular supervised ML algorithm that is used primarily in classification tasks. Its objective is to estimate the probability that an instance belongs to a particular class. The algorithm is called LR due to its connection to regression methods. However, unlike linear regression, which produces continuous output values, LR uses a sigmoid function to calculate the probability for the designated class. This distinction allows LR to predict whether an instance belongs to a specific class or not, rather than providing a continuous range of values as linear regression does. Figure 3.8.3 illustrates the aspects of logistic regression.

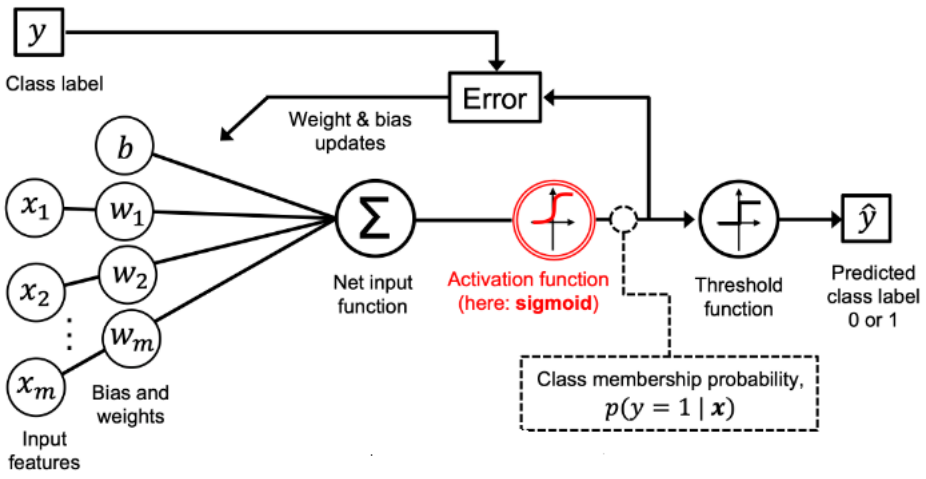


Figure 3.8.3 LR Algorithm

For binary classification LR used to transform a linear combination of input features $(X_1, X_2, \ldots, X_r)$ into a probability value between 0 and 1 given by the logistic function:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_r x_r)}} \quad (3.14)$$

Where:

$P(Y = 1|\ X)$ is the probability of belonging to class 1 given the input feature X,
e is the base of the natural logarithm (approximately 2.71828).
b_0 is the intercept term (bias).
$(\beta_1, \beta_2, \ldots, \beta_r)$ are the coefficients for the input features $(X_1, X_2, \ldots, X_r)$.

To make a binary classification decision, we set a threshold for the predicted probability. If $P(Y = 1|X)$ is greater than or equal to the threshold, you classify the data point as belonging to class 1; otherwise, you classify it as belonging to class 0. The LR model estimates the coefficients $(\beta_1, \beta_2, \ldots, \beta_r)$ during the training phase to fit the model to the training data. The coefficients determine the slope and position of the decision boundary, which separates the two classes.

In the context of this study, we have applied the LR algorithm to the task of intrusion detection on the KDD99 dataset, a prominent benchmark in cybersecurity. The dataset encompasses various attack classes, including “dos”, “normal”, “probe”, “r2l” and “u2r”. This investigation focuses on formulating the LR model to effectively discern between these distinct attack classes, with an emphasis on the absence of specific training parameters. In this study we tailored the LR formulation to address each attack class individually. Let P_{dos} , P_{normal} , P_{probe} , P_{r2l}, P_{u2r}, be the probability of belonging to the “dos”, “normal”, “probe”, “r2l” and “u2r classes, respectively. The class-specific formulation was determined by:

$$P_{dos} = \left(\frac{1}{1 + e^{-(\beta_{dos,0} + \beta_{dos,1} x_1 + \beta_{dos,2} x_2 + \cdots + \beta_{dos,r} x_r}}\right) \quad (3.15)$$

Similar formulations are established for the classes “normal”, “probe”, “r2l” and “u2r, ensuring that the LR model adapts to the distinct characteristics of each attack category.

The LR model in this study is intentionally trained without specific parameters, allowing the algorithm to dynamically learn the coefficients from the data. This parameter-free training approach fosters adaptability, particularly valuable in intrusion detection, where the diversity of attack patterns may be challenging to capture with fixed

parameters. The LR algorithm, customized for the diverse attack classes within the KDD99 dataset, serves as a robust tool in intrusion detection. The absence of predefined parameters enhances the model's adaptability, while the class-specific formulations contribute to its ability to discern nuanced patterns associated with different cyber threats. This study advances our understanding of logistic regression in intrusion detection, highlighting its applicability and efficacy in the absence of explicit parameterisation.

3.9. PERFORMANCE MATRIX

The performance matrix in ML is used to evaluate the performance and effectiveness of a trained model on a given dataset. These metrics provide quantitative measures that help to assess how well the model is performing in terms of various aspects such as accuracy, precision, recall, F1-score, and Confusion Matrix. This section of the study answered the last research question, how to evaluate the effectiveness of the models using a performance matrix.

3.9.1. ACCURACY

Accuracy measures the general correctness of the predictions made by the model and is calculated as the ratio of correctly classified instances to the total number of instances. It was used to assess just how well the average measurement of multiple measurements stacks up against the standard measurements of the same item or the true value. This performance matrix is measured as the proportion of correctly predicted attacks in the test data, compared to all attacks in the same test data. Since this study involves the detection of various attack classes within the KDD99 dataset, including “dos”, “normal”, “probe”, “r2l” and “u2r” the evaluation of the model’s performance requires a metric that accommodates the multiclass nature of the classification task. Accuracy, a fundamental metric, gauges the overall correctness of predictions across all classes. In the context of a 5-class scenario, the accuracy of the model will be determined by its ability to correctly classify instances into the respective attack classes. It is defined by the following equation:

$$Accuracy = \frac{Number\ of\ correctly\ predicted\ Attacks}{Total\ number\ of\ tested\ Attacks} \qquad (3.16)$$

To break the down further, considering each class C_i , the number of correct predictions (C_i) for class C_i is the sum of TP for that class:

$C_i = TP\ for\ C_i$

TP for class C_i are instances correctly classified as belonging to class C_i . The total number of instances for class C_i is the sum of TP, FP, FN, and TN:

$$Total\ instance\ for\ C_i = (TP\ for\ C_i) + (FP\ for\ C_i) + (FN\ for\ C_i) + (TN\ for\ C_i) \tag{3.17}$$

Thus, the accuracy for each class C_i is given by:

$$Acc_{C_i} = \frac{TP\ for\ C_i}{Total\ instance\ for\ C_i} \tag{3.18}$$

The overall accuracy for 5-class scenario is the average of the accuracies for each class:

$$Accuracy = \frac{1}{5}\sum_{i=1}^{5} Acc_{C_i} \tag{3.19}$$

This formula captures the model's performance across all attack classes, providing a comprehensive assessment of its ability to correctly classify instances into the appropriate categories. In the complex landscape of intrusion detection with multiple attack classes, the formulation of accuracy as described above ensures a holistic evaluation of model performance. This multiclass accuracy metric is integral to understanding the effectiveness of the model in correctly identifying instances across all classes, shedding light on its overall classification prowess within the 5-class framework.

3.9.2. CONFUSION MATRIX

A confusion matrix is a performance evaluation matrix that is used in ML to visualise the performance of a classification model. It provides a tabular representation of the predicted and actual class labels of a data set. The matrix is constructed based on four different values: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Here is a breakdown of the elements in a confusion matrix:

i. TP: The number of instances that are correctly predicted as positive (i.e., the model predicted them as positive, and they are indeed positive).
ii. TN: The number of instances that are correctly predicted as negative (i.e., the model predicted them as negative, and they are indeed negative).
iii. FP: The number of instances that are incorrectly predicted as positive (that is, the model predicted them as positive, but they are actually negative).
iv. FN: The number of instances that are incorrectly predicted as negative (that is, the model predicted them as negative, but they are actually positive).

A confusion matrix is usually presented in the table format as follows:

Confusion Matrix			
Model Class		**Actual Class**	
		Positive	Negative
Predicted Class	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Table 3.1 Confusion Matrix

The matrix allows for a comprehensive understanding of the model's performance, enabling the calculation of various performance metrics such as accuracy, precision, recall, and F1 score. It provides information on the types of errors made by the model and aids in assessing its effectiveness in differentiating between classes. Since this study involves the detection of various attack classes within the KDD99 dataset, including “dos”, “normal”, “probe”, “r2l” and “u2r” the confusion matrix serves as a pivotal visualisation for assessing the model's performance across these 5 classes. A 5-class confusion matrix provides a comprehensive overview of the model's ability to correctly classify instances and helps identify areas where misclassifications may occur. Consider the following structure for a generic 5-class confusion matrix:

		Predicted class				
Actual	Classes	Dos	Normal	Probe	R2L	U2R
	Dos	T_{dos}	F_{dos}	F_{dos}	F_{dos}	F_{dos}
	Normal	F_{normal}	T_{normal}	F_{normal}	F_{normal}	F_{normal}
	Probe	F_{probe}	F_{probe}	T_{probe}	F_{probe}	F_{probe}
	R2L	F_{r2l}	F_{r2l}	F_{r2l}	T_{r2l}	F_{r2l}
	U2R	F_{u2r}	F_{u2r}	F_{u2r}	F_{u2r}	T_{u2r}

Table 3.2 5-class Confusion Matrix

This confusion matrix structure provides a visual representation of the model's performance across the 5 attack classes. It is instrumental in identifying the strengths and weaknesses of the model in correctly predicting instances for each class, aiding in a detailed evaluation and interpretation of its multiclass intrusion detection capabilities. The classification flagged by a red colour illustrate that the attack is not correctly identified and the green colour indicate that the attack is correctly identified

3.9.3. PRECISION

Precision quantifies the proportion of true positive predictions out of the total instances predicted as positive. It focusses on the accuracy of positive predictions and is calculated as the ratio of true positives to the sum of true positives and FPs. Given that this study involves the detection of various attack classes within the KDD99 dataset, including “dos”, “normal”, “probe”, “r2l” and “u2r” the precision metric is crucial for assessing the model's performance on a class-specific basis. In the context of a 5-class scenario, precision quantifies the accuracy of positive predictions for each class, providing insights into the model's ability to correctly identify instances within specific attack categories. Generally, the formula for calculating precision is given by:

$$Precision = \frac{TP}{(TP + FP)} \tag{3.20}$$

In a multiclass scenario, precision for a particular class C_i is calculated as the ratio of TP for that class to the total number of instances predicted as positives for that class:

$$Prec_{C_i} = \frac{TP\ for\ C_i}{Total\ Predicted\ Positives\ for\ C_i} \tag{3.21}$$

Mathematically, the total predicted positives for class C_i is expressed as the sum of TP and FP:

$$Total\ Predicted\ Positives\ for\ C_i = (TP\ for\ C_i) + (FP\ for\ C_i) \tag{3.22}$$

Therefore, the precision for each class C_i is given by

$$Prec_{C_i} = \frac{TP\ for\ C_i}{TP\ for\ C_i + FP\ for\ C_i} \tag{3.23}$$

In the context of a 5-class scenario, the overall precision is the average of the precisions for each class:

$$Precision = \frac{1}{5}\sum_{i=1}^{5} Prec_{C_i} \tag{3.24}$$

This formula encapsulates the precision of the model across all attack classes, providing a nuanced understanding of its ability to make accurate positive predictions within each specific category. In the intricate landscape of intrusion detection with multiple attack classes, the precision metric, as formulated above, offers a granular assessment of the model's positive prediction accuracy for each class. This class-specific evaluation contributes to a comprehensive understanding of the model's precision performance within the 5-class framework, shedding light on its ability to make accurate and targeted positive predictions for individual attack categories.

3.9.4. RECALL

Recall that measures the proportion of true positive predictions out of the actual positive instances in the dataset. It focusses on the ability of the model to identify positive instances and is calculated as the ratio of true positives to the sum of true positives and false negatives. In the context of this study, which involves the detection of various attack classes within the KDD99 dataset, including “dos”, “normal”, “probe”, “r2l” and “u2r” the recall metric is essential for assessing the model's performance on a class-specific basis. Given the 5-class scenario, recall measures the model's ability

to correctly identify instances belonging to a specific attack category, offering insights into its effectiveness in capturing instances of TP predictions. Generally, the formula for calculating recall is given by:

$$Recall = \frac{TP}{(TP + FN)} \tag{3.25}$$

In a multiclass scenario, recall for a particular class is calculated as the ratio of true positives for that class to the total number of instances that actually belong to that class:

$$Rec_{C_i} = \frac{TP\ for\ C_i}{Total\ Actual\ Positives\ for\ C_i} \tag{3.26}$$

Mathematically, the total actual positives for class C_i is expressed as the sum of TP and FN:

$$Total\ Actual\ positives\ for\ C_i\ = (\text{TP for } C_i\ ) + (\text{FN for } C_i) \tag{3.27}$$

Therefore, the recall for each class C_i is given by:

$$Rec_{C_i} = \frac{TP\ for\ C_i}{TP\ for\ C_i + FN\ for\ C_i} \tag{3.28}$$

In the context of a 5-class scenario, the overall recall is the average of the recalls for each class:

$$Recall = \frac{1}{5}\sum_{i=1}^{5} Rec_{C_i} \tag{3.29}$$

This formula captures the recall of the model across all attack classes, providing a nuanced understanding of its ability to correctly identify instances belonging to individual attack categories. In the nuanced landscape of intrusion detection with multiple attack classes, the recall metric, as formulated above, offers a detailed assessment of the model's ability to capture instances of true positive predictions for each class. This class-specific evaluation contributes to a comprehensive understanding of the model's recall performance within the 5-class framework, shedding light on its effectiveness in identifying instances associated with specific attack categories.

3.9.5. F1-SCORE

F1 score It is the harmonic mean of precision and recall, providing a balanced measure that combines both metrics. The F1 score is useful when the data set is unbalanced and there is a need to consider both precision and recall. In the context of this study, which involves the detection of various attack classes within the KDD99 dataset, including “dos”, “normal”, “probe”, “r2l” and “u2r” the F1 score metric is a harmonious blend of precision and recall, providing a balanced evaluation of the model's performance on a class-specific basis. Given the 5-class scenario, the F1 score measures the trade-off between precision and recall, offering a comprehensive assessment of the model's ability to make accurate and balanced predictions. Generally, the formula for calculating F1 score is given by:

$$F1\ Score = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3.30}$$

in a multiclass scenario, F1 score for a particular class C_i is calculated as the harmonic mean of the precision and recall:

$$F1_{C_i} = \frac{2 \times Prec_{C_i} \times Rec_{C_i}}{Prec_{C_i} + Rec_{C_i}} \tag{3.31}$$

This formula ensures that the F1 score considers both false positives and false negatives, providing a balanced measure that rewards models achieving both high precision and high recall. In the context of a 5-class scenario, the overall F1-score is the average of the F1 scores for each class:

$$F1\ Score = \frac{1}{5}\sum_{i=1}^{5} F1_{C_i} \tag{3.32}$$

This formula encapsulates the F1 score of the model across all attack classes, offering a nuanced understanding of its ability to balance precision and recall within individual attack categories. In the intricate landscape of intrusion detection with multiple attack classes, the F1 score metric, as formulated above, provides a comprehensive evaluation of the model's balance between precision and recall for each class. This class-specific evaluation contributes to a nuanced understanding of the model's F1 score performance within the 5-class framework, offering insights into its ability to make accurate and well-balanced predictions for individual attack categories.

3.10. CONCLUSION

This chapter has discussed the four main stages of our proposed research method. Feature selection and ranking, Tree-based ID model, traditional ML classification model, and performance matrix. In the feature selection and ranking step, we proposed a hybrid approach of two mainly used selection methods in ID. The attributes of significant importance were then applied to a DT and classification algorithms to produce the performance matrix considered by the study. This chapter also discussed the data set and its preparation, implementation tools, and procedures used to implement the study.

3.11. SUMMARY

The cyber security data was extracted from the UCI ML Repository. The dataset had simulated attacks falling into four main categories, namely DoS, U2R, R2L, and Probing Attack. All experiments were implemented using Python programming language due to most of the fascinating libraries with the capabilities to model, visualise, and transform data. The selection method was proposed using a hybrid of Gini index and Information gain to collect features of significant importance from the dataset. The performance of the models was evaluated considering accuracy, precision, recall, F1-score, and confusion matrix.

CHAPTER 4 – DATA ANALYSIS

4. INTRODUCTION

In this chapter, we embark on a journey through the intricate landscape of data analysis, where the raw information we have gathered transforms into valuable insights, shedding light on the research questions that have guided our study. Our exploration is based on a comprehensive dataset sourced accurately and carefully prepared to ensure its relevance and reliability. To begin, we revisit the core research aim, objectives, and questions that have propelled our investigation. These objectives serve as our compass, guiding our analytical endeavours and framing our interpretation of the data. As we dive into the depths of our dataset, we remain mindful of these guiding principles, seeking to extract meaningful patterns, relationships, and answers. We aimed to comparatively analyse tree-based ID model and ML classification algorithms using cyber security dataset. The main objectives of this study were to:

i. Develop a tree-based detection model using the concepts of a DT.
ii. Conduct security feature selection and ranking to select features with significant importance.
iii. Apply the tree-based ID model and traditional ML classification models to the KDD-99 dataset.
iv. Evaluate the effectiveness of these models by measuring performance matrix, namely, Accuracy, Precision, Recall, F1-Score and Confusion matrix.

The following research questions have arisen directly from the carefully defined research objectives, demonstrating the inherent connection between the objectives of the study and the research questions:

i. How to develop tree-based ID algorithms based on the concepts of a DT?
ii. How to conduct the security feature selection and ranking to select features with significant importance?
iii. How to apply a cyber-security dataset to a tree-based ID model and traditional ML classification algorithms?
iv. How to evaluate the effectiveness of the models using performance matrix?

However, for the fruitful fulfilment of these objectives, there are essential steps in the ML pipeline which enforces data quality, data validity and reliability, data

understanding, and optimisation of feature representation which ultimately lead to more accurate and reliable ML models.

The remainder of this chapter is organised as follows. In Section 4.1, we provide an insightful introduction to our data source, shedding light on its origin, scope, and essential characteristics. Section 4.2 takes us through the initial phases of our analysis journey, with a focus on data exploratory analysis and cleaning. It offers a comprehensive snapshot of the data's essential properties and details the steps taken to ensure their quality. Moving forward, Section 4.3 delves into data preprocessing, covering the meticulous steps we took to prepare the data for machine learning. Our core analysis unfolds in Section 4.4, where we apply a range of ML classification algorithms to the pre-processed data. This section delves into the specifics of each model, its parameters, and the results obtained. In Section 4.5, we perform a comparative analysis, dissecting the performance of these algorithms, highlighting their strengths and weaknesses, and offering insights into the most effective methods for our dataset. Section 4.6 is where we tie it all together, drawing well-supported conclusions that align with our research objectives. Finally, Section 4.7 summarises this chapter, providing a handy reference for readers to quickly grasp the key points of our data analysis journey.

4.1. THE DATA SOURCE

The foundation of our analysis lies in the data source, which consists of network connection records, and each record contains various features that describe different aspects of network traffic. This dataset was selected due to its direct relevance to our research questions and its potential to provide insights that contribute to our understanding of network intrusions.

```
[ ] import pandas as pd
    from tensorflow.keras.utils import get_file

    try:
        path = get_file('kddcup.data_10_percent.gz', origin='http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz')
    except:
        print('Error downloading')
        raise

    print(path)

    # This file is a CSV, just no CSV extension or headers
    # Download from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
    df = pd.read_csv(path, header=None)

    Downloading data from http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz
    2144903/2144903 [==============================] - 1s 0us/step
    /root/.keras/datasets/kddcup.data_10_percent.gz
```

Figure 4.1.1 Sourcing Data

This Python code snippet in Figure 4.1.1 serves the purpose of automating the download and subsequent loading of a dataset, known as the KDD-99 dataset, which is frequently used in the domain of NID and cybersecurity research. The code begins by importing the necessary libraries, namely Pandas for data manipulation and TensorFlow's "get_file" function for downloading files from the Internet. It then uses a try-except block to manage the download process. Inside the try block, the "get_file" function is used to fetch the dataset from a specified URL, and the resulting file's path is stored in the "path" variable. Should any issues arise during the download, such as an invalid URL or a failed download, the except block will execute, printing an error message to the console and re-raising the exception for further handling. Assuming a successful download, the code proceeds to load the downloaded data into a Pandas DataFrame named "df" using the "read_csv" function. The "header=None" argument indicates that the dataset does not contain column headers. In essence, this code simplifies the process of obtaining and preparing the KDD Cup 1999 dataset for

subsequent analysis, making it convenient for researchers and practitioners in the field of cybersecurity and machine learning.

```
# The CSV file has no column heads, so add them
df.columns = [
    'duration', 'protocol_type','service','flag',
    'src_bytes','dst_bytes','land','wrong_fragment',
    'urgent','hot','num_failed_logins','logged_in',
    'num_compromised','root_shell','su_attempted',
    'num_root','num_file_creations','num_shells',
    'num_access_files','num_outbound_cmds','is_host_login',
    'is_guest_login','count','srv_count','serror_rate','srv_serror_rate',
    'rerror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate',
    'srv_diff_host_rate','dst_host_count','dst_host_srv_count',
    'dst_host_same_srv_rate','dst_host_diff_srv_rate',
    'dst_host_same_src_port_rate','dst_host_srv_diff_host_rate',
    'dst_host_serror_rate','dst_host_srv_serror_rate',
    'dst_host_rerror_rate','dst_host_srv_rerror_rate','outcome'
]
```

Figure 4.1.2 Adding Column Headers

In this code snippet from Figure 4.1.2, column headers were added to the Pandas DataFrame “df” to provide meaningful labels for each attribute in the dataset. The dataset originally had no column headers, so this step is essential for better data understanding and manipulation. Each column is assigned a name based on the nature of the data it represents. Adding these column headers makes it much easier to work with and analyse the dataset, as it provides clear context for each data point and facilitates data exploration and modelling in subsequent steps of data analysis or ML tasks. In the KDD99 dataset, the “outcome” column typically contains labels or identifiers that specify the classification of each network connection or data record as either a normal network activity or a specific type of network attack. The “outcome” column is essentially the ground truth or target variable used to train and evaluating ID models and other cybersecurity-related analyses.

The “outcome” column is crucial for supervised ML tasks, where the goal is to train a model to predict the type of network activity based on available features. For example, it helps in developing IDSs that can automatically identify and respond to network

attacks by classifying incoming traffic into normal or attack categories. The study considered adding a new column that classifies each attack types from the “outcome” column to their existing class. Classifying the values of the “outcome” column into these attack categories is crucial to understand, analyse, and modelling network attacks in the KDD99 dataset. It enables effective model training, evaluation, security monitoring, and reporting, all of which are essential components of NID and cybersecurity efforts.

```
attacks_types = {
    'normal': 'normal',
'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',
}
```

```
[6]  df['attacksType'] = df.outcome.apply(lambda r:attacks_types[r[:-1]])

     df.head()
```

Figure 4.1.3 Adding attack class column.

The code in Figure 4.1.3 adds a new column named “attacksTypes” to a dataset and prints the first five records of the dataset, likely for the purpose of categorising individual network connections or packets into broader attack categories. This column serves as a valuable addition because it simplifies and streamlines the process of analysing and managing network traffic data. By assigning each network event to an attack category, it enables easy classification and allows for a higher-level analysis of the dataset. This categorisation is particularly useful in ID, as it enables security professionals and data analysts to quickly identify and assess the distribution of different types of network attacks, facilitating better threat detection, analysis, and reporting.

4.2. DATA EXPLORATORY ANALYSIS AND CLEANSING

In the realm of research and data-driven decision-making, the initial encounter with raw data marks the genesis of understanding and insight. The process of data exploratory analysis, often referred to as exploratory data analysis, is the compass that guides researchers through this intriguing landscape. In this section, we embark on a journey of discovery, peering into the depths of our dataset to uncover hidden patterns, anomalies, and the untold stories that data hold. Data exploratory analysis is a multidimensional process, weaving together statistical, visual, and computational techniques to inspect, summarise, and visualize data. It goes beyond mere data inspection; it is an artful blend of science and intuition, guided by curiosity and an unrelenting quest for insights. Through this process, we aim to reveal the underlying structure, relationships, and anomalies that may reside within our data.

4.2.1. DATA INFORMATION AND STRUCTURE

The study considered the df.info() method to understand the information and structures about the dataset. This method in Pandas provides a concise summary of the DataFrame “df” including information on the dataset’s structure, the data types of each column, the number of non-null values, and memory usage. It is a handy way to get a quick overview of the dataset. When you run df.info(), it will display the number of entries (rows) in the DataFrame, the number of columns, the names of each column, the data type of each column (e.g., integer, float, object), the number of non-null values in each column (i.e., the count of non-missing values) and the memory usage of the DataFrame. This information is useful to understand the dataset’s size, its data types, and whether there are missing values. It is a crucial step in the data exploration process to determine if further data preprocessing or cleaning is required before conducting data analysis or building ML models.

Appendix A provides a high-level overview of the dataset, showing the number of entries (rows), the data types of columns, and how much memory is being used. It is a great starting point for further data analysis and exploration. Let us break down the output and understand each part:

I. <class “pandas.core.frame.DataFrame”>: This line indicates that the object “df” is a Pandas DataFrame.

II. RangeIndex: 494021 entries, 0 to 494020: This line provides information about the index of the DataFrame. In this case, it is a RangeIndex starting from 0 to 494020, indicating that there are 494021 rows or entries in the DataFrame.

III. Data columns (total 43 columns): This indicates that there are a total of 43 columns in the DataFrame.

IV. #, Column, Non-Null, Count, and Dtype: This header row provides the column-wise information.
   a) #: The column numbers.
   b) Column: The name of the column.
   c) Non-Null Count: The number of non-null (non-missing) values in that column.
   d) Dtype: The data type of the values in that column.

V. The following lines list each column in the DataFrame along with the relevant information. For example, 0 duration 494021 non-null int64: This line corresponds to the first column. It is named “duration” and it contains 494021 non-null values of data type “int64” (64-bit integer). Similar information is provided for all 43 columns.

VI. dtypes: float64(15), int64(23), object (5): This line summarises the data types present in the DataFrame. In this example, there are 15 columns with the data type “float64”, 23 columns with data type “int64” and 5 columns with data type “object” (usually indicating text or categorical data).

VII. Memory usage: 162.1 + MB: This line shows the approximate memory usage of the DataFrame. In this case, the DataFrame consumes approximately 162.1 megabytes of memory.

In general, this output is a concise summary of the DataFrame structure, indicating the number of rows, the number of columns, the column names, the data types, the number of non-null values, and the memory usage. It is a useful initial assessment to understand the dataset’s characteristics before diving into data analysis or further pre-processing steps. This method helped the study to understand each attribute for the subsequent data cleansing process.

4.2.2. DATA CLEANSING

In the world of data analysis and research, data is a fundamental resource, but its true value can only be unlocked when it is clean, consistent, and reliable. In this section, we embark on the critical phase of data cleaning, a meticulous process that lays the foundation for robust and meaningful analysis. Data cleaning is an iterative, multifaceted process that aims to rectify, mitigate, or eliminate issues within the dataset. These issues can manifest in various forms, including missing values, outliers, duplicate entries, inconsistencies in formatting, and more. The primary goal of data cleaning is to ensure that the dataset is accurate, complete, and suitable for analysis.

```
# Checking for DUPLICATE values
df.drop_duplicates(keep='first', inplace = True)
# Checking for NULL values
print('='*35)
print('Null values in dataset are',len(df[df.isnull().any(1)]))
print('='*35)
# For now, just drop NA's (rows with missing values)
df.dropna(inplace=True,axis=1)

print('='*35)
print("Read {} rows.".format(len(df)))
print('='*35)

# stored the data into a pickle file so we can load through
df.to_pickle('df.pkl')
```

```
===================================
<ipython-input-7-945ac81f965d>:5: FutureWarning: In a future version
  print('Null values in dataset are',len(df[df.isnull().any(1)]))
Null values in dataset are 0
===================================
===================================
Read 145586 rows.
===================================
```

Figure 4.2.2.1 Data Cleansing

Figure 4.2.2.1 focused on data cleaning, including removing duplicate rows, checking for NULL values, dropping columns with missing values, and saving the cleaned dataset to a pickle file for future use. The dataset did not have any missing values, but it was found that it contains duplicate records. The rows after the removal of duplicates and missing values are now down to 145586 from 494021.

4.2.3. DATA EXPLORATORY ANALYSIS

Exploratory Data Analysis (EDA) is a crucial phase in the process of understanding and preparing data for analysis, and it plays a significant role in the context of the KDD99 dataset, which is a well-known dataset in the field of cybersecurity and NID. EDA is the initial step in any data analysis project, and its primary goal is to gain insight, detect patterns, and uncover potential problems in the dataset.

4.2.3.1. TARGET COLUMNS DISTRIBUTION

In the context of our research, the graphical representation of the attack distribution in the training data for the KDD99 dataset reveals a notable pattern in figure 4.2.3(a) visualised on the basis of the “outcome” attribute. The x-axis of the graph corresponds to the individual class labels, while the y-axis, represented on a logarithmic scale, quantifies the number of data points associated with each class. Notably, the class “normal” emerges as the predominant category, followed by “neptune.”, “back.”, “teardrop.”, “satan.”, “warezclient.”, and several others. In contrast, classes such as “perl.”, “phf.”, and “spy.” appear as relatively rare occurrences.

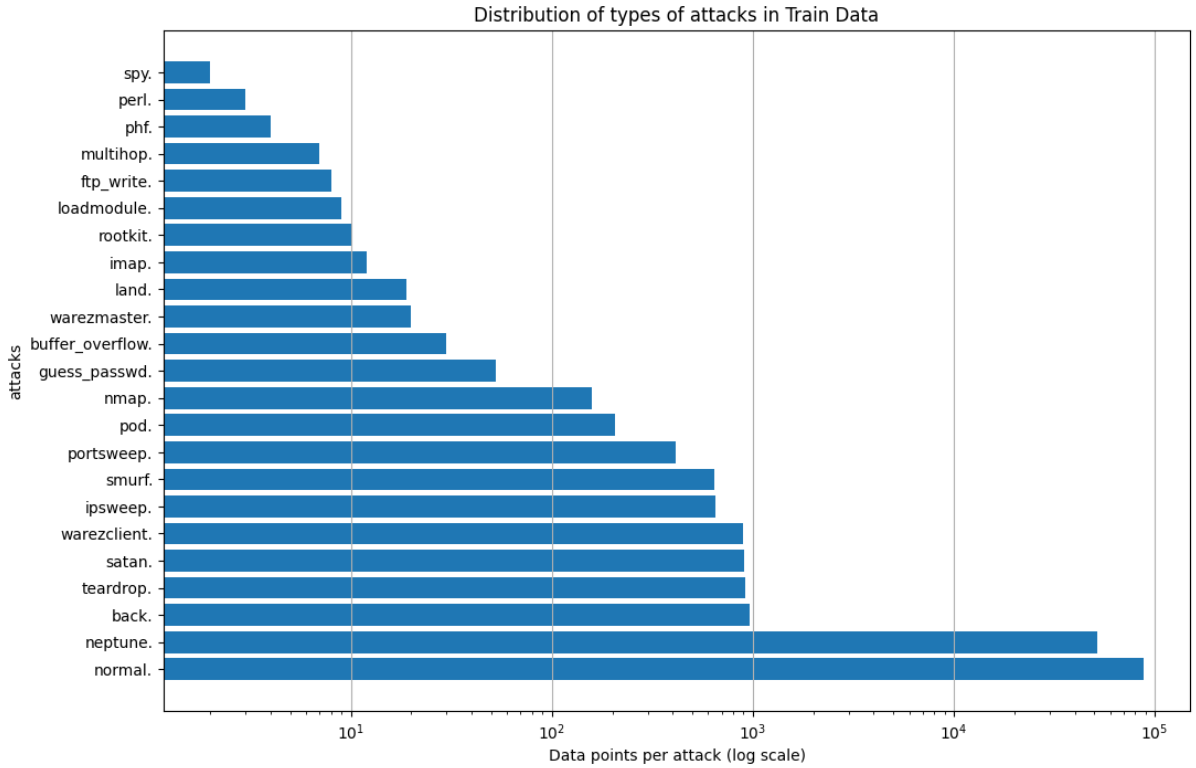


Figure 4.2.3.(a) Distribution of attack types

This observation underscores a significant class imbalance within the dataset, a factor that has profound implications for our research. Imbalanced data distributions, as evidenced here, can present challenges when training ML models, particularly in the context of classifying new data points. The predominance of “normal.” instances may lead to a model’s bias towards this class, potentially compromising its ability to accurately predict the less common categories. Addressing this imbalance is a critical consideration in our research, as it requires the exploration of strategies for mitigating the inherent bias and optimising model performance, thus improving the robustness of our IDSs.

During data extraction and loading, we grouped all the attack outcomes into their respective class, which reduced the complexity of the dataset. The bar chart in figure 4.2.3(b) offers valuable insight into the distribution of the five available classes within the training data, categorised by the type of train. These classes, namely “normal”, “dos”, “probe”, “u2r”, and “r2l” exhibit distinct frequencies, as illustrated along the x-axis, with the corresponding count of data points depicted on the y-axis, presented on a logarithmic scale.

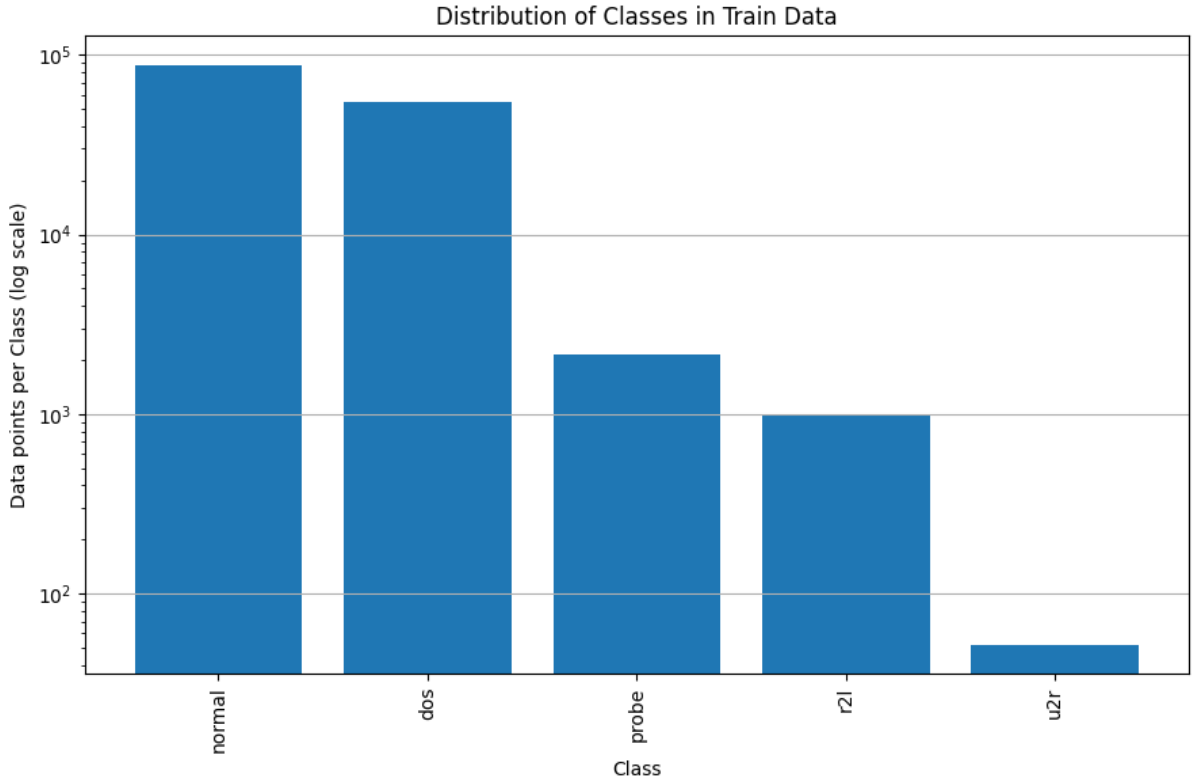


Figure 4.2.3.(b) Distribution of attack classes

Evidently, the class distribution within the training data is characterised by a significant imbalance. Notably, the “normal” class emerges as the overwhelmingly predominant category that has 87832 data points, signifying its abundance within the dataset. Following this, the “dos” class appears as the second most frequent category, although significantly less common than “normal”. Subsequently, the “probe” class is represented, followed by the “r2l” and “u2r” classes.

4.2.3.2. CATEGORICAL FEATURES DISTRIBUTION

The identification of categorical features within the KDD99 dataset is of paramount importance, as it underpins the foundation for many critical aspects of analysis and model development. Categorical features often represent factors such as connection types, services, and flags that are vital in understanding network behaviour and the presence of intrusions. Separating them from numerical attributes is crucial because different pre-processing and analysis techniques are applied to these two types of features. Therefore, identifying categorical features is a crucial step in preparing the data for subsequent analysis and ensuring that it accurately captures the nuances of network traffic and intrusion patterns.

```
# Identifying categorical features
numeric_cols = df._get_numeric_data().columns # gets all the numeric column names

categorical_cols = list(set(df.columns)-set(numeric_cols))
categorical_cols
```

```
['service', 'outcome', 'protocol_type', 'flag', 'attacksType']
```

Figure 4.2.3.(c) Identifying categorical features.

The code provided in Figure 4.2.3.(c) efficiently identifies categorical features within the KDD99 dataset by distinguishing them from the numeric and binary attributes. This distinction is essential as categorical features, such as protocol types, services, and flags, play a pivotal role in characterising network connections and detecting intrusion attempts. Identifying these categorical attributes allows for tailored data pre-

processing, including one-hot encoding or label encoding, which is essential for ML model development. It ensures that these non-numeric attributes are appropriately represented, enabling the model to capture the underlying patterns in network data. By distinguishing between numeric and categorical features, we lay the foundations for effective analysis, modelling, and, ultimately, the development of robust IDSs. We found that “Service”, “Output”, “Protocol_type”, “Flag” and “AttacksType” are the features that have categorical values.

4.2.3.2.1. PROTOCOL TYPE

In the context of the KDD99 dataset and NID, the “protocol_type” attribute plays a fundamental role in characterising network connections. It represents the communication protocol used in each network connection, categorising them into various types such as the transmission control protocol (TCP), user datagram protocol (UDP), and Internet control message protocol (ICMP). The type of protocol serves as a critical feature in ID because different protocols exhibit different behaviours and are associated with specific network activities. By understanding and analysing the distribution of protocol types in network data, cybersecurity experts and data analysts can gain insights into the nature of network traffic and identify anomalies or suspicious patterns that can signify potential intrusions. Therefore, the "protocol_type" attribute is a crucial element in building effective ID models and enhancing network security.

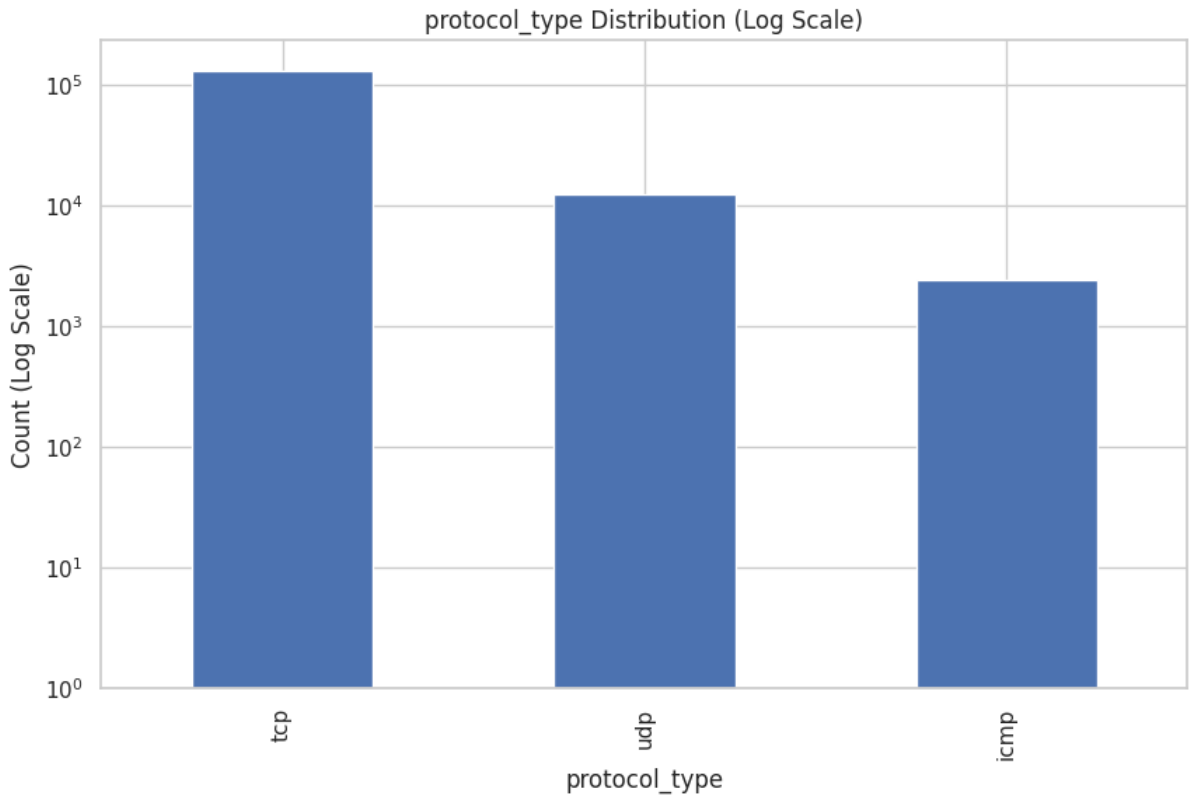


Figure 4.2.3.(d) Distribution of Protocol Type

The Figure 4.2.3. (d) provided shows the distribution of different protocol types on a logarithmic scale. This logarithmic scale helped to visualise the relative prevalence of these protocol types, even when the number of occurrences varies significantly. The chart highlights that TCP is the most common protocol, closely followed by UDP and ICMP. TCP is the primary protocol for internet data transfer, ICMP is utilised for network management and control messages, and UDP is preferred for low-latency applications like streaming media. Notably, the distribution of protocol types is far from uniform. Several protocol types have a substantial presence, while many others are less prevalent. This pattern suggests that a select few protocol types dominate the majority of internet traffic. The chart offers valuable information on the distribution of protocol types on the Internet, emphasising the prominence of TCP, ICMP, and UDP, while also showcasing the considerable variation in the prevalence of different protocols.

4.2.3.2.2. SERVICE

In the context of the KDD99 dataset and NID, the "service" attribute is a pivotal feature that characterises the specific network service associated with a network connection. It represents the kind of service or application that is being used within each network connection, such as HTTP, FTP, SMTP, or other network services. The "service" attribute is of great significance in building IDSs as it offers crucial insights into the nature of network traffic. Different services have different traffic patterns and usage behaviours, and these patterns can be indicative of normal network activity or potentially malicious behaviour. Analysing the distribution of "service" helped in understanding the composition of network services and detecting anomalies or suspicious patterns that may signal intrusion attempts. For example, unusual usage of a service or a sudden spike in network traffic associated with a specific service could be an indicator of a security breach. Therefore, the "service" attribute is a key component in the process of building robust ID models and enhancing network security.

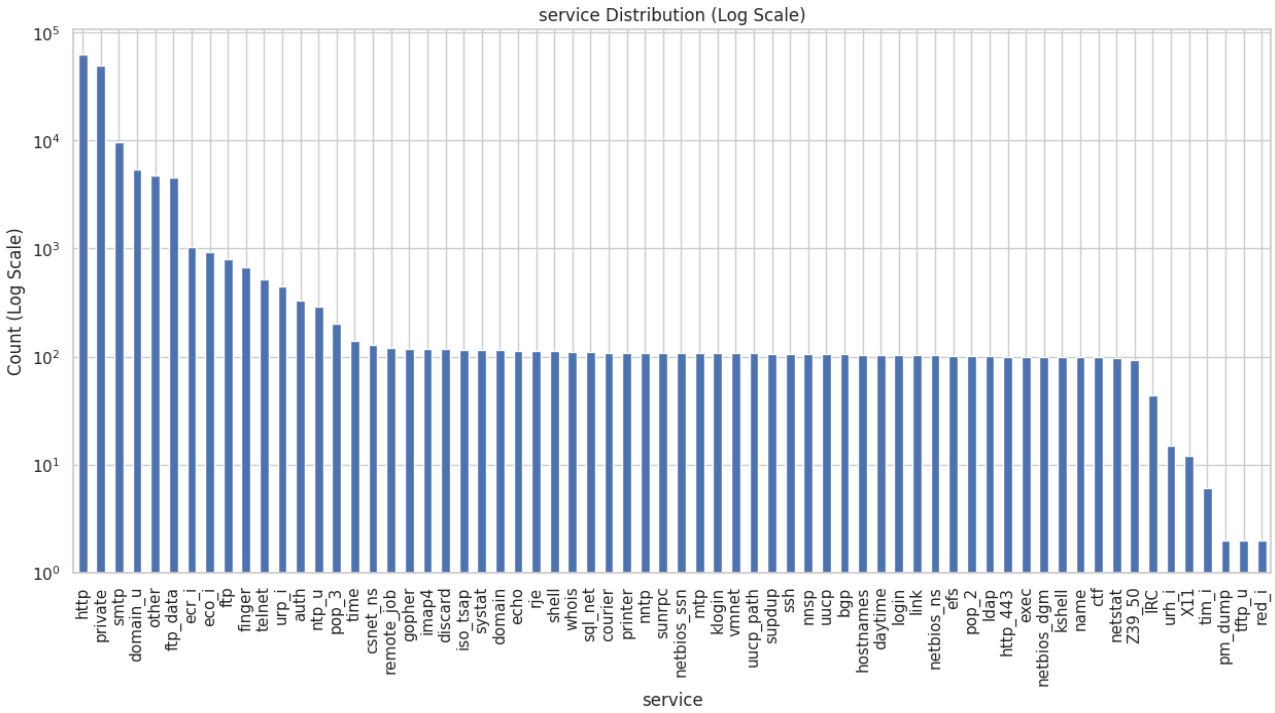


Figure 4.2.3(e) Distribution of Service

Figure 4.2.3.(e) shows the distribution of the service feature in the KDD99 dataset. The distribution is skewed towards the lower end of the scale, meaning that there are a few services that are very common and many services that are very rare. The most common services are “http”, “private”, “smtp”, “domain_u”, and “other”. The least common services are “tftp_u”, “red_i”, “exec”, “netbios_dgm”, and “kshell”. The distribution of the service feature is likely due to several factors. One factor is that some services are simply more popular than others. For example, http is the protocol used to transfer web pages, which is very common. Another factor is that some services are more likely to be targeted by attackers. For example, smtp is the protocol used to send email, which is a common target for spammers and phishing attacks.

4.2.3.2.3. FLAG

The “flag” attribute is a critical feature that provides insight into the status and control of network connections, especially in the context of TCP. This attribute represents the various flags or indicators associated with network connections, such as synchronisation and finish (SF), connection attempt(S0), rejected connection (REJ), and more. The “flag” attribute is of paramount significance in building IDSs as it helps to characterise and understanding the state of network connections and the potential risks associated with them. Analysing the distribution of the “flag” is crucial to detect anomalies in network behaviour. For example, a sudden influx of “S0” flags can indicate connection attempts, potentially from malicious sources, while “SF” flags signify established connections. The presence of “REJ” flags can suggest rejected connections, which may be the result of intrusion attempts. Therefore, the “flag” attribute is a vital component in the development of ID models, as it helps to recognise unusual or suspicious connection states that can signify potential security breaches.

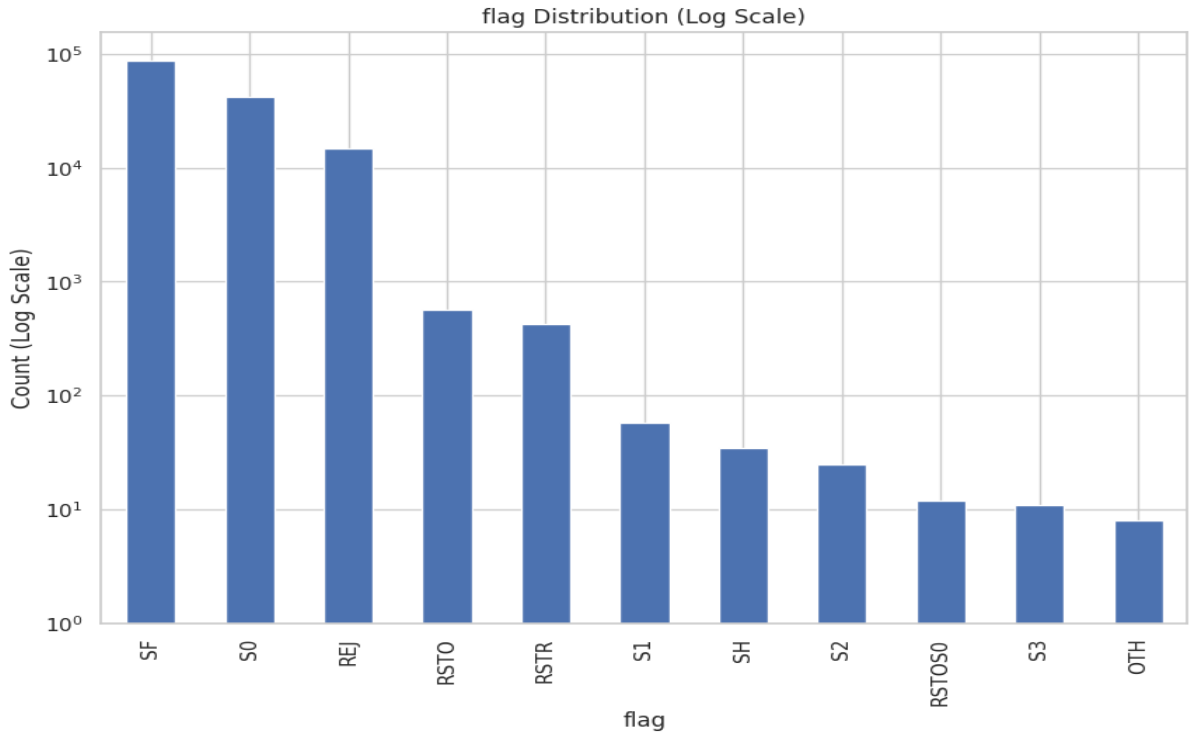


Figure 4.2.3.(f) Flag Distribution

Figure 4.2.3.(f) shows that the majority of connections are normal (SF). Abnormal connections are relatively rare, but there is a wide variety of different types of anomalies. The most common anomalous flag values are REJ, reset (RSTO), and reset to closed (RSTR). These flags indicate that the connection was terminated prematurely, which can be a sign of an attack. Other anomalous flag values include internal server error (IS), shellcode detected (SH), other anomalies (OTH), and urgent pointer (URG). These flags indicate a variety of different types of problem, which could be caused by attacks, misconfigurations, or other factors. It is important to note that the flag attribute is not perfect. It is possible for normal connections to be flagged as anomalous, and vice versa. However, the flag distribution log scale graph can be useful for getting a general overview of the types of anomalies that are present in the KDD99 dataset.

4.2.4. BINARY FEATURES DISTRIBUTION

When analysing the KDD99 dataset for NID and cybersecurity, the identification of binary features holds significant importance. Binary features are attributes that take on only two distinct values, often represented as 0 and 1. Exploring binary features is crucial for several reasons. Firstly, they offer a simplified representation of attributes,

which can be highly informative in characterising network connections. Secondly, binary features can be indicative of specific network behaviours or states, which makes them essential for identifying patterns associated with normal and intrusive activities. In the realm of ML and ID models, the presence of binary features can greatly influence model performance and classification accuracy. Therefore, the process of identifying binary features in the KDD99 dataset is a vital step in understanding and preparing the data for effective ID and cybersecurity analysis.

```
# lets look into deeply to identify if there are any other binary data exists or not
binary_cols = []
for col in numeric_cols:
  if len(df[col].unique()) <= 2:
      result = []
      s = df[col].value_counts()
      t = float(len(df[col]))
      for v in s.index:
          result.append("{}({}%)".format(v,round(100*(s[v]/t),1)))
      print("{} - [{}]".format(col, " , ".join(result)))
      binary_cols.append(col)
```

```
land - [0(100.0%) , 1(0.0%)]
logged_in - [0(50.9%) , 1(49.1%)]
root_shell - [0(100.0%) , 1(0.0%)]
num_outbound_cmds - [0(100.0%)]
is_host_login - [0(100.0%)]
is_guest_login - [0(99.5%) , 1(0.5%)]
```

Figure 4.2.4(a) Identifying Binary Columns

The Python code in Figure 4.2.4(a) is designed to identify binary data or attributes within a DataFrame, specifically focussing on the columns containing numeric values. It achieved this by iterating through each numeric column (stored in “numeric_cols”). For each column, it checks the number of unique values it contains. If the number of unique values is less than or equal to 2, it suggests that the data in that column are binary (i.e., it has only two distinct values). The code then provides a detailed breakdown of these binary columns, including the binary values and their distribution in the dataset. It calculates the percentage of each value’s occurrence in the column, which can be helpful in understanding the balance between the binary states. Finally, it prints out this information, listing the column name along with the binary values and their respective percentages, and adds the column to the “binary_cols” list for reference. This code is particularly useful for identifying binary features, which are often crucial in ML tasks such as classification, as they can significantly impact model

performance and influence decision-making processes. The output of the code in the above figure shows “num_outbound_cmds” and “is_host_login” have only one unique value. An attribute with only one unique value can negatively affect ML models as it does not provide any discriminatory power and might introduce unnecessary complexity. It is important to review such attributes and consider their impact on the problem. These attributes will be permanently removed from the dataset in the section where we will be plotting the correlation matrix. Removing attributes with only one unique value in the KDD99 data prior to training ML algorithms is a common practice to enhance the efficiency and effectiveness of the model. Attributes with only one unique value offer no variation or discriminatory power in the dataset, making them redundant for the learning process. These attributes neither contribute to the model's predictive capacity nor provide meaningful insights, often leading to overfitting, where the model memorises data rather than generalizing from it. By eliminating such attributes, we streamline the dataset, reduce noise, and allow the ML algorithm to focus on relevant features, ultimately improving model performance and interpretability. Before this task can be performed, we explored numeric columns and their standard deviations.

4.2.5. Numeric Features

The KDD99 dataset also contains the numeric columns representing quantitative attributes related to network traffic and system logs. These numeric attributes include features such as the duration of connections, the number of bytes transferred, and various statistics related to network packets.

```
# identify remaining numeric features by subtracting categorical columns
numeric_features = list(set(numerics_features)-set(binary_cols))
numeric_features
```

Figure 4.2.5(a) Getting Numeric Features.

The Python code in Figure 4.2.5(a) was designed to extract the names or identifiers of numeric columns within a dataset while excluding both binary columns and categorical columns. It does this by first having a list named “numeric_features” that

presumably contains the names of numeric columns. Then, it subtracts a set of binary columns from this list, using the set difference operation. The result is that the “numeric_features” list now exclusively holds the names of columns that contain numerical data after removing the binary columns. This code is particularly useful in data analysis tasks where it is essential to distinguish between different types of data, such as numeric and categorical, as it helps in preparing data for various analytical processes or ML algorithms.

4.2.5.1. STANDARD DEVIATION

In this section, calculating the standard deviation for numerical attributes of the KDD99 dataset helped the study to assess data variability, identify outliers, and make decisions about feature selection and data quality, aiding in understanding and preparation of the data for analysis and modelling.

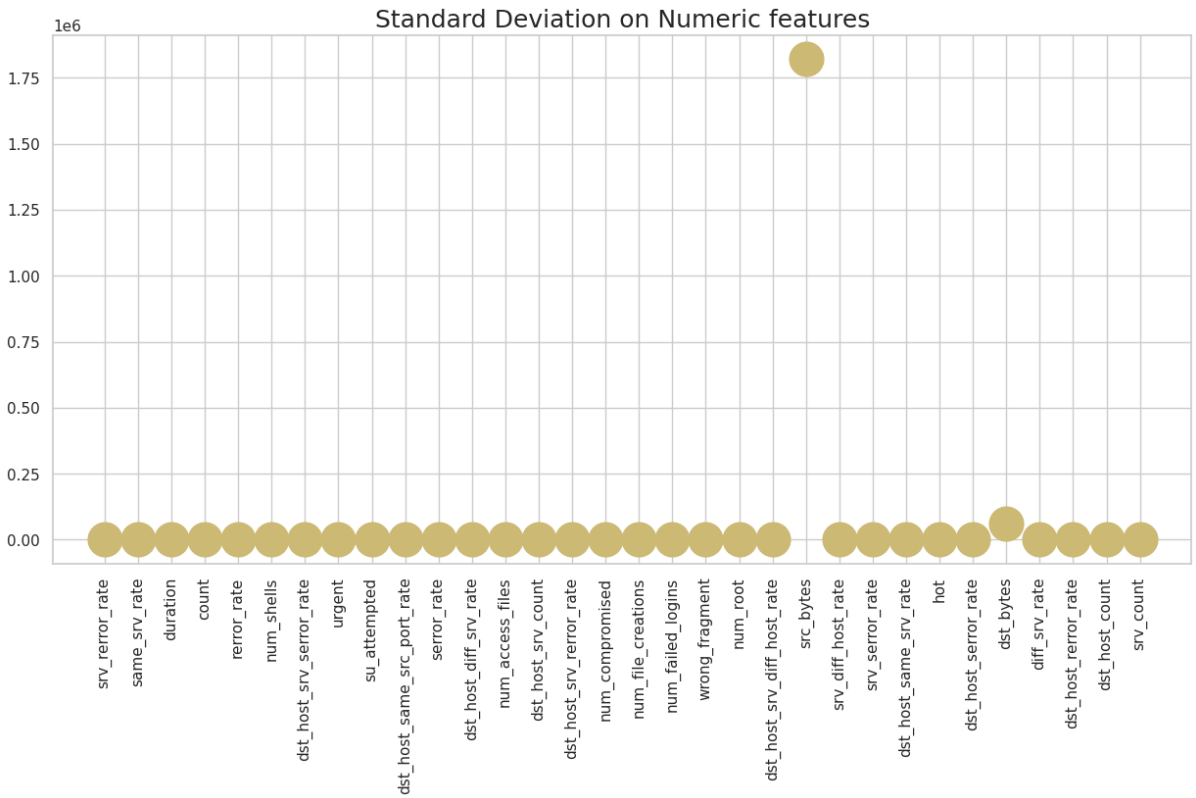


Figure 4.2.5(b) Standard Deviation of Numeric features

Figure 4.2.5(b) depicts the distribution of the standard deviation of the numerical features on the KDD99 dataset, showing that most of the features have a standard deviation between 0 and 1. This indicates that the data is relatively well-scaled, with most of the features having values that are close to the mean. However, there is only

one feature that has a standard deviation that is much larger than 1. This feature is likely to have a wider range of values and may be more difficult to model. The standard deviation can be used to identify features that may be problematic for IDSs. Features with a large standard deviation may be more difficult to use to distinguish between normal and anomalous behaviour. These features will be standardised prior to the development of the ML model, and we will discuss the technique in Section 4.3 of this chapter.

4.2.6. CORRELATION MATRIX

A correlation matrix is a crucial tool in data exploratory analysis that provides a comprehensive view of the relationships between numerical attributes in a dataset. It quantifies the degree of association or dependency between pairs of variables, typically using statistical measures such as the Pearson correlation coefficient. In the context of the KDD99 dataset, which contains numerous numeric attributes related to network connections, a correlation matrix is indispensable. It helps data analysts and ML practitioners gain insights into how these attributes interact and whether there are significant dependencies or correlations between them. The correlation matrix from the KDD99 dataset in Appendix B shows the strength and direction of the linear relationships between the different features in the dataset. The values in the matrix range from -1 to 1, with a value of 1 indicating a perfect positive correlation, a value of -1 indicating a perfect negative correlation, and a value of 0 indicating no correlation.

The correlation matrix indicates that most of the features exhibit weak correlations with each other. This implies that there is a lack of strong linear relationships between the majority of attributes in the dataset. Such weak correlations suggest that changes in one feature are generally not strongly associated with corresponding changes in other features. In practical terms, this means that the dataset encompasses a diverse set of attributes that do not exhibit straightforward dependencies.

However, it is important to note that a few features show moderate to strong correlations with each other. These observed correlations might provide insight into specific interactions or behaviours within the dataset, and understanding them could be crucial for modelling and analysis. Nevertheless, these strong correlations appear to be the exception rather than the rule given the overall prevalence of weak correlations among the dataset features.

The existence of weak correlations across most features could imply that the dataset is made up of diverse and independent attributes. It also suggests that the data is not dominated by linear dependencies, making it challenging to predict one feature's value based solely on the values of others. This complexity highlights the need for more advanced statistical and ML techniques when attempting to uncover patterns or make predictions within this dataset. Furthermore, it underscores the importance of considering non-linear relationships, interactions, or feature engineering in the analysis, as traditional linear approaches may not be suitable for capturing the underlying data structure.

In essence, the correlation matrix provides valuable insight into the relationships between the different features in the KDD99 dataset. This information can be used to develop more effective IDSs and better understand the behaviour of attackers.

4.3. DATA PREPROCESSING

Data preprocessing is a foundational and indispensable phase in the analysis of the KDD99 dataset, which serves as a vital resource for NID and cybersecurity. This intricate process encompasses a series of activities aimed at transforming and enhancing the raw data, ensuring its readiness for subsequent analysis and model development. Given the dataset's comprehensive nature, data pre-processing is paramount. It involves applying dummy variables, feature encoding of categorical attributes, and standardising numerical attributes. Furthermore, this process is crucial in ensuring the dataset is appropriately split into training and testing sets, and, when needed, mapping intrusion types into binary categories to simplify the classification task. In essence, data preprocessing in the KDD99 dataset lays the foundation for the creation of accurate and reliable ID models, optimising network security and safeguarding against potential threats.

4.3.1. APPLYING DUMMIES

This method was primarily considered because it ensures that the model can fully comprehend and utilise categorical information, by enhancing its capacity to distinguish patterns and relationships within the data, which is particularly crucial for accurate ID.

```
[33] def apply_dummies(df, feature):
         get_dummies = pd.get_dummies(df[feature])
         for x in get_dummies.columns:
             dummy_name = f"{feature}-{x}"
             df[dummy_name] = get_dummies[x]
         df.drop(feature, axis=1, inplace=True)
         return None
```

Figure 4.3.(a) Applying Dummies Variables

In Figure 4.3(a) defines a function called “apply_dummies” designed to apply one-hot encoding to categorical features in the KDD99 dataset. The function used the “pd.get_dummies” function to generate binary columns and assigns them meaningful names based on the original feature's name and category. Subsequently, the targeted categorical feature is dropped from the DataFrame to ensure compatibility with ML algorithms that require numerical input. In the context of the KDD99 dataset, applying dummies variables to the dataset is essential for a few reasons. First, it allows for the inclusion of categorical features in ML models, which typically require numeric inputs. Second, it mitigates the risk of bias by preventing the algorithm from assuming ordinal relationships that may not exist in categorical data. Finally, it enhances the model's ability to detect patterns and relationships within the categorical attributes, ultimately contributing to the development of more accurate and robust ID models.

4.3.2. FEATURE ENCODING

Feature encoding is applied to the KDD99 dataset to transform categorical attributes into a numerical format that ML algorithms can process. This transformation is essential to ensure that the models can effectively analyse and learn effectively from these categorical features. By converting each category into binary columns, one-hot encoding preserves the integrity of the data while preventing the model from misinterpreting categorical values as having a numerical order, ultimately contributing to more accurate ID in network security.

```
# apply one hot encoding for categorical columns except the target and target_types
for feature in categorical_cols:
  if feature not in ['attacksType']:
    apply_dummies(df, feature)
```

Figure 4.3(b) Feature Encoding

In Figure 4.3 (b), we applied one-hot encoding to categorical columns in the KDD99 dataset using the function designed in 4.3.1, except for the “attacksType” column, which represents the target variable. One-hot encoding is a crucial data preprocessing step for several reasons. In the context of the KDD99 dataset, which contains categorical attributes like “protocol_type”, “service”, and “flag”, one-hot encoding is essential to convert these non-numeric features into a numerical format with which ML algorithms can work effectively. By creating binary columns for each category within a categorical feature, one-hot encoding ensures that the algorithm does not impose ordinal relationships on the data that may not exist. This is particularly important for NID, as categorical attributes often represent different aspects of network connections, and one-hot encoding helps the models capture the distinct behaviours associated with each category. In general, applying one-hot encoding is a critical step in optimising the KDD99 dataset for ML and improving the performance and accuracy of ID models.

4.3.3. FEATURE STANDARDISATION

In the section of our research, we used the “StandardScaler” from scikit-learn to standardise a specific set of numeric columns in the KDD99 dataset. Standardisation is a crucial preprocessing step in the context of the KDD99 dataset for several reasons. Network traffic and ID data often contain numeric attributes with varying scales, and standardisation is employed to bring these attributes to a common scale, typically with a mean of zero and a standard deviation of one. This process is important for ML algorithms because it ensures that all features contribute equally to model training. When attributes have different scales, algorithms may give more weight to attributes with larger scales, potentially leading to biased and suboptimal model performance. Standardising the data mitigates this issue and allows the model to make fair comparisons among different attributes. This, in turn, enhances the effectiveness of

ID models in identifying patterns and anomalies in network traffic data, ultimately leading to more accurate and reliable IDSs.

4.3.4. DATA SPLIT

Splitting the data in the KDD99 dataset into training and testing sets is a fundamental practice in machine learning. We reserved a portion of the data for training ML models while keeping another portion for testing and evaluation. This division ensures that the models are not only trained but also assessed on unseen data, helping us gauge their generalisation and predictive capabilities. This step is important as it ensures that models perform well on new, real-world data and not just on the data they were trained on, ultimately contributing to robust and reliable IDSs.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

# Prepare your feature matrix (X)
x_columns = df.columns.drop('attacksType')
X = df[x_columns].values

# Use 'attacksType' as the target variable (y)
y = df['attacksType'].values

# Normalize the features
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 4.3(c) Data Split

Figure 4.3(c) demonstrates the process of preparing the data for ML in the KDD99 dataset. First, we separated the feature matrix (X) and the target variable (y) with “attacksType” being the target. The feature matrix was then normalised using the MinMaxScaler to ensure that all features are within the range [0, 1]. This normalisation is particularly useful for models sensitive to the scale of input features. Subsequently, we performed a key step in machine learning, which is splitting the data into training and testing sets using the “train_test_split” function. The 0.2 split size was applied to the dataset, indicating that 20% of the data will be used for testing, while 80% will be used for training. This split size is a commonly used practice in ML to strike a balance between having enough data to train a model effectively and having sufficient data reserved for testing. An 80-20 split is often chosen, as it provides a substantial training dataset while ensuring that the testing dataset is representative enough to assess the

model's performance accurately. The random state was set to 42 to ensure reproducibility, allowing the same split to be generated when the code is run multiple times. In essence, this split size is used to strike a balance between training and testing while maintaining a level of randomness that avoids data bias.

4.4. APPLYING ML CLASSIFICATION ALGORITHMS

The deployment of ML models for ID in the KDD99 dataset represents a pivotal phase in network security and cyber threat mitigation. This multifaceted approach involves the implementation of various ML algorithms, each tailored to its unique strengths. DTs, constructed using a hybrid of Gini index and information, offer interpretability and a comprehensive decision-making framework. SVM excel in complex, high-dimensional spaces, aiming to identify optimal hyperplanes for classification. KNN harnesses the simplicity of proximity-based classification, ideal for problems with unclear decision boundaries. Logistic regression, a classic, yet reliable method, models binary outcomes with interpretability.

The evaluation of these models relies on a comprehensive set of metrics, including accuracy, precision, recall, f1-score, and confusion matrix. These metrics allow for a complex evaluation, considering not only the overall accuracy of model predictions, but also their ability to identify TPs, true negatives, FPs, and false negatives. In the context of ID, the trade-off between these metrics is of great significance, given the potential repercussions of missed threats or false alarms. This deployment attempts to enhance network security by harnessing the power of machine learning, offering a proactive and data-driven approach to identify and mitigate potential threats. Using a diverse range of models and a careful evaluation process, this deployment aims to create a robust and adaptive defence system, capable of safeguarding network environments against evolving cybersecurity challenges.

We have thoughtfully presented the outcomes of our model’s evaluations, including precision, recall, F1-score, and accuracy in a clear and illustrative manner. To enhance the accessibility and credibility of these results, we have thoughtfully visualised the classification reports using graphical representations. These visualisations offer readers an intuitive understanding of the models’ performance and the associated metrics. For your reference and to ensure the validity and transparency of our findings, these classification reports are thoughtfully provided in Appendix C.

4.4.1. DECISION TREE

The DT was constructed for ID in the KDD99 dataset which represented a strategic and methodical approach to create an effective classification model. The DT classifier is an integral part of this process, designed with specific settings such as balanced class weights, a Gini impurity criterion, and controlled depth and node splitting conditions. This tree is constructed using a dataset split into training and testing sets to ensure model robustness and accuracy.

The feature selection phase plays a critical role in the construction of the DT. Feature importance is evaluated through two distinct methods: the Gini impurity index and information gain. Gini impurity measures the node impurity by assessing how often a randomly chosen element would be incorrectly classified. Information gain quantifies the reduction in uncertainty about the target variable after a split, contributing to the quality of the decision nodes. The importance scores obtained through these methods are then normalised to ensure their comparability and combined using weighted averages. In this setup, the Gini impurity carried a weight of 70%, while the information gain contributed 30% to the combined feature importance score. This addressed the first three objectives of the study, which were mainly based on the construction of the proposed DT model. However, the model must be evaluated based on the performance matrix obtained during the deployment of this model.

The results of the DT model on the KDD99 dataset have significant implications for ID. The model achieved an accuracy score of 99.98%, indicating that it accurately classified most of network connections into intrusion and non-intrusion categories. This high accuracy demonstrates the model's effectiveness in distinguishing between normal and intrusive network activities.

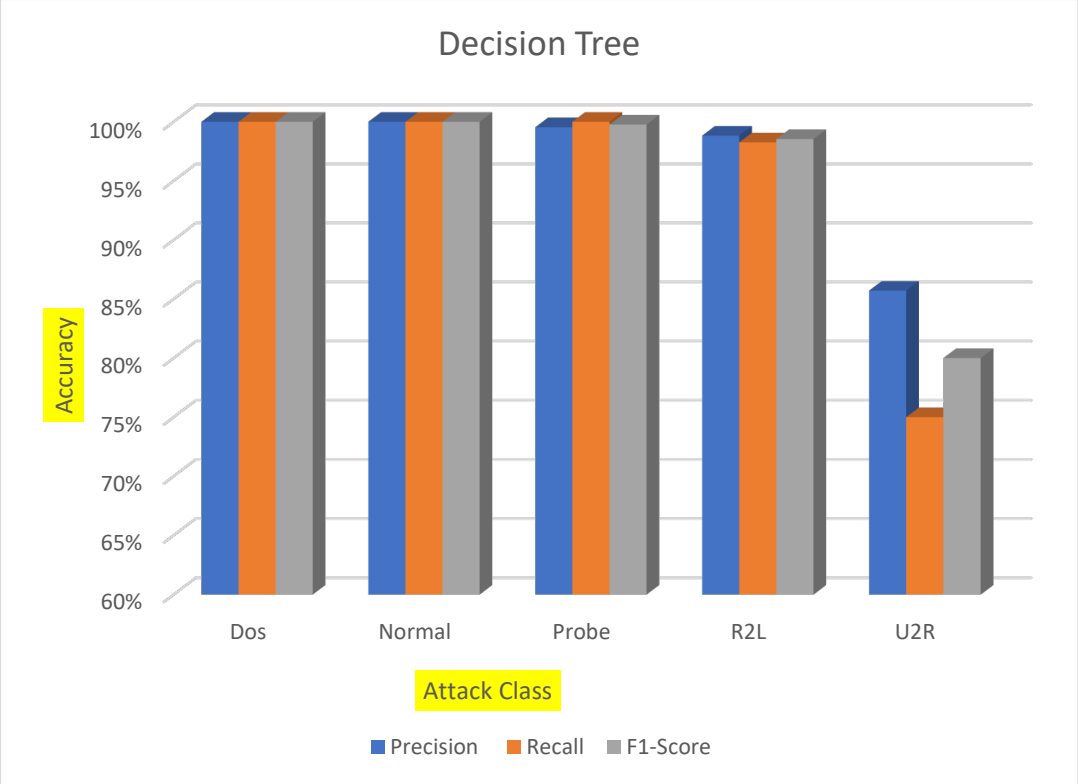


Figure 4.4.1(a) DT Precision, Recall and F1-Score

Figure 4.4.1(a) depicts the evaluation of three performance matrix namely precision, recall and f1-score. The precision and recall values for most classes, including “Dos”, “Normal”, and “probe”, are closely or equal to 100%. This implies that the model is capable of correctly classifying network connections, minimising both FPs and false negatives. For ID, this is crucial as it reduces the risk of missing real threats and generating false alarms. The F1-scores, which provide a balance between precision and recall, are also consistently high for “dos”, “normal”, and “probe” classes. This indicates that the model achieves a harmonious trade-off between precision and recall, making it effective in identifying intrusion patterns while minimising false detections. Although the model achieved better accuracies in the matrix discussed above, some of the matrix including the prediction accuracy has shown that the model does not correctly predict all classes. The confusion matrix was included as one of our evaluation matrices to check how many classes were misclassified.

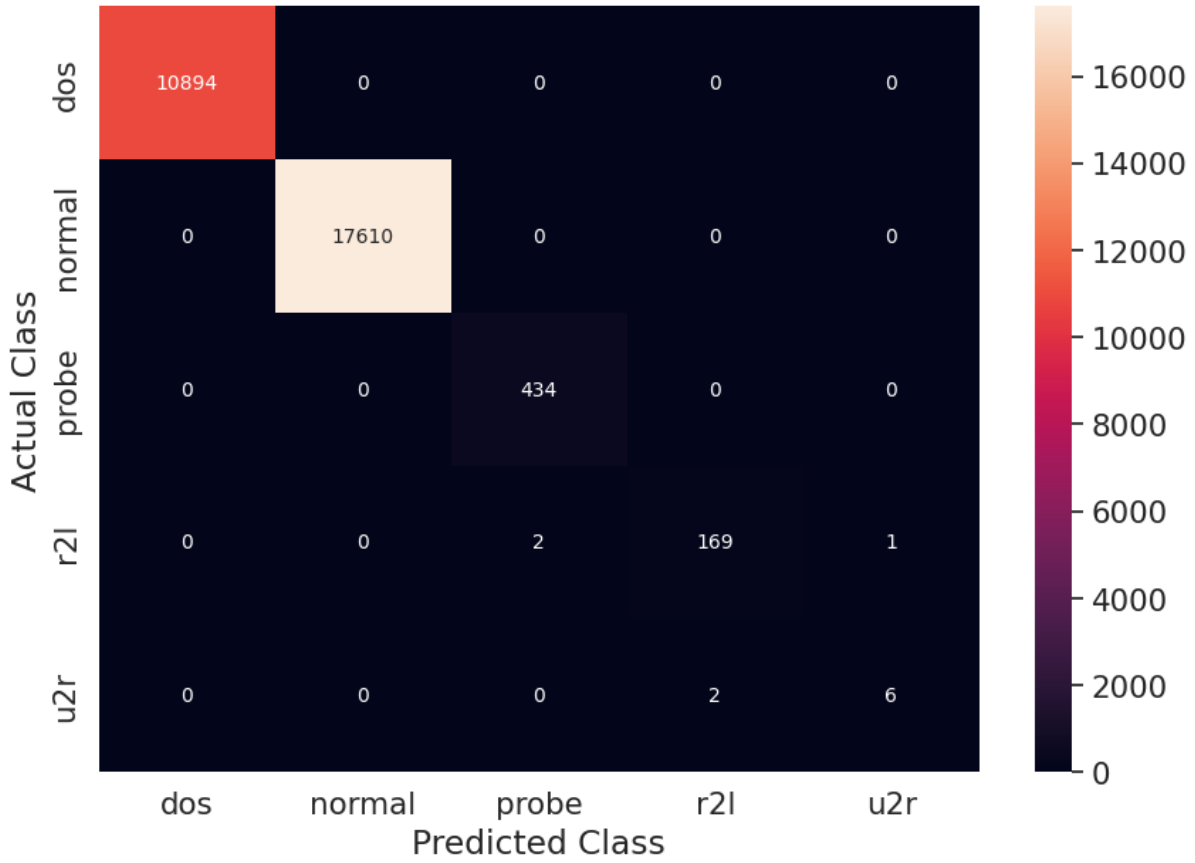


Figure 4.4.1(b) DT Confusion Matrix

The confusion matrix, as depicted in Figure 4.4.1(b), provides insight into the classification results. It reveals that the model correctly identified 10,894 instances as “dos”, 17,610 instances as “normal”, 434 instances as “probe”, 169 instances as “r2l” and 6 instances as “u2r”. These results underscore the model's ability to accurately classify most instances. However, the model exhibited some limitations in predicting certain attacks. Notably, it incorrectly classified 2 instances originally belonging to the “r2l” class as “probe” and 2 instances from the “u2r” class as “r2l”. Furthermore, 1 instance from the “r2l” class was mistakenly classified as “u2r”. This suggests a slight inability to accurately predict attacks within the “probe”, “r2l” and “u2r” classes accurately.

This observed limitation may stem from the dataset's class distribution, where these attack classes have fewer records. It is possible that the model had difficulty learning these classes effectively due to the limited number of instances available for training. These results have significant implications for ID in the KDD99 dataset. The model demonstrates good performance, capable of accurately identifying network intrusions

while maintaining a low FP rate. This level of accuracy and precision is crucial in network security, where minimising false alarms and rapidly identifying potential threats is of utmost importance. The model's strong performance suggests its potential for real-world deployment as an effective tool in safeguarding network environments against a wide range of cybersecurity threats.

4.4.2. SUPPORT VECTOR MACHINE

The construction of the SVM for ID in the KDD99 dataset is a meticulous and strategically engineered process. Leveraging the Radial Basis Function (RBF) kernel, known for its ability to handle complex and non-linear data patterns, the SVM is tailored to excel in the task of distinguishing between normal and intrusive network activities. Its hyperparameter configuration, including a gamma value of 0.1 and a regularisation parameter (C) of 1.0, is carefully chosen to optimise its performance. The RBF kernel's capacity to capture intricate decision boundaries is particularly valuable for ID, where identifying subtle patterns and relationships in high-dimensional data is essential. The SVM construction process involves learning from the training data to adapt to the unique characteristics of the KDD99 dataset. Its mission is to effectively classify network connections, with a focus on minimising false alarms and achieving high precision and recall, making it a powerful defence mechanism against potential cybersecurity threats in network environments. This section addresses the third and fourth objective of the study.

The SVM results applied to the KDD99 dataset carry profound implications for ID. The SVM model has achieved an astonishing accuracy score of 100.00%, signifying that it has flawlessly classified network connections into intrusion and non-intrusion categories. This extraordinary accuracy underscores the SVM's capability to effectively discern between normal and intrusive network activities.

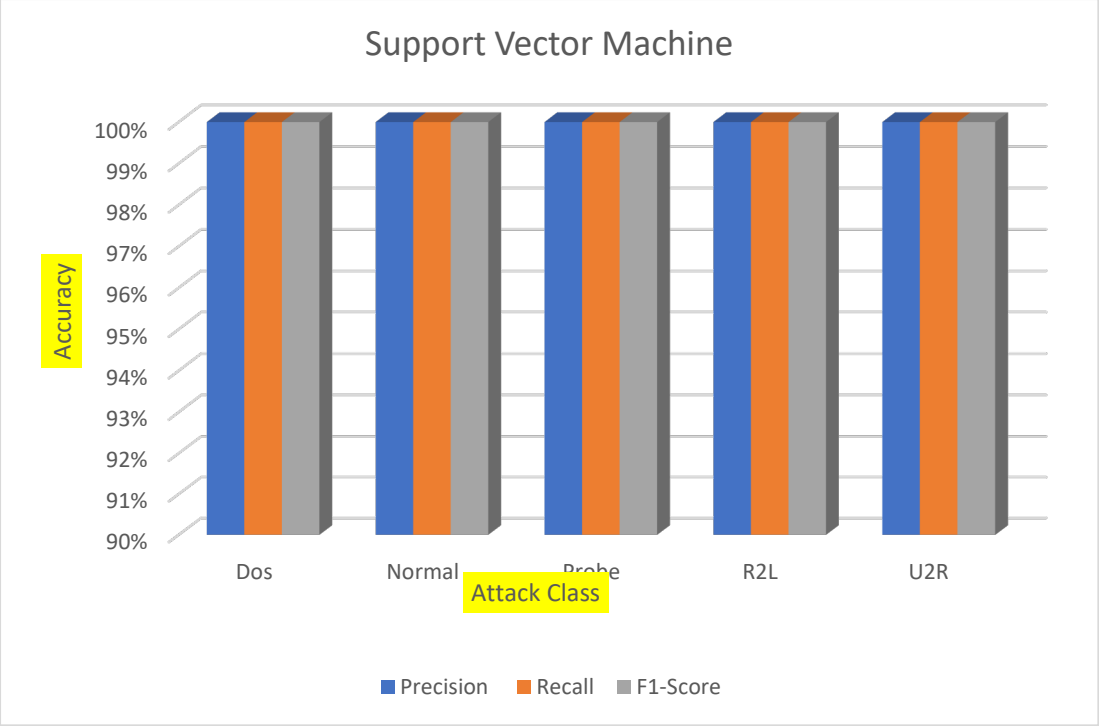


Figure 4.4.2(a) SVM Precision, Recall and F1-Score.

Figure 4.5.2(a) depicts the evaluation of three performance matrix namely precision, recall and f1-score. The precision and recall values for all classes, including “dos”, “normal”, “probe”, “r2l”, and “u2r” are an impressive 100%, indicating that the model has the ability to minimise false alarms and capturing true threats. This extraordinary performance is paramount in ID, where precision and recall significantly impact the effectiveness of security measures. The F1-scores, which balance precision and recall, are also at their maximum value of 100% across all classes. This harmonious balance suggests that the SVM is not only accurate, but also has an exceptional ability to identify intrusion patterns while maintaining a low false detection rate. Although the model achieved 100% accuracies in the matrices discussed above, we need to evaluate if all of the instances were classified correctly. The confusion matrix was included as one of our evaluation matrices to check how many classes were misclassified.

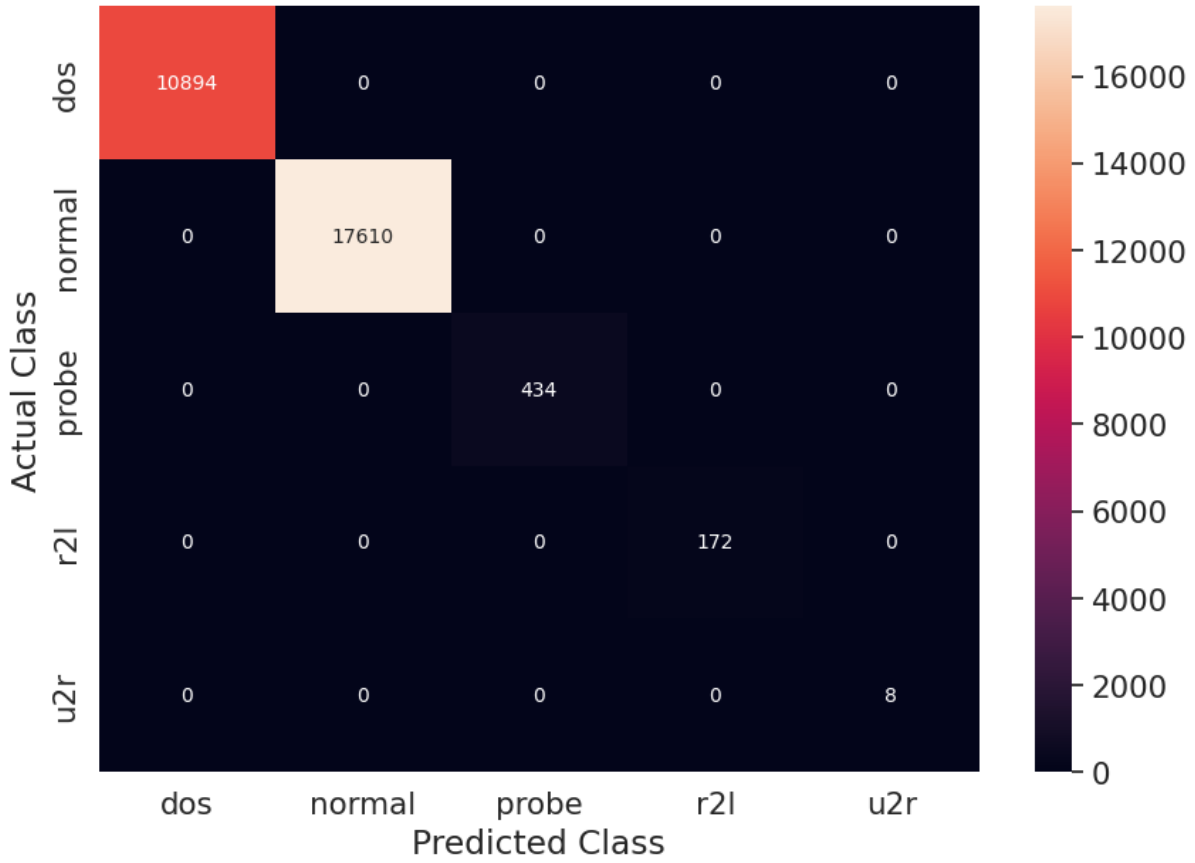


Figure 4.4.2(b) SVM Confusion Matrix

Figure 4.5.2(b) depicts that SVM correctly classified. This outcome underscores the exceptional discriminatory power and robustness of the SVM model in the context of ID within the KDD99 dataset. SVM is renowned for its ability to define clear decision boundaries and capture complex, non-linear patterns in data. Its strength lies in maximising the margin between different classes, which can lead to highly accurate and reliable classifications. In this specific case, SVM's ability to correctly classify all classes, including ‘dos’, ‘normal’, ‘probe’, ‘r2l’ and “u2r” is indicative of its proficiency in understanding and distinguishing network connections' underlying characteristics.

These results have profound implications for ID within the KDD99 dataset. The SVM model demonstrates an unprecedented level of accuracy, precision, and recall, making it a formidable tool for identifying network intrusions and minimising false alarms and missed threats. Its ability to achieve such high performance is a testament to its adaptability to complex high-dimensional data and its effectiveness in a real-world network security context. The SVM's remarkable results highlight its potential to

serve as a reliable and robust defence mechanism against a wide spectrum of cybersecurity threats.

4.4.3. K-NEAREST NEIGHBOURS

The construction of the KNN classifier in the context of ID within the KDD99 dataset embodies a distinctive and flexible approach. KNN, an instance-based learning algorithm, categorises network connections based on their proximity to neighbouring data points. Unlike other models that rely on hyperparameters and complex decision boundaries, KNN thrives on simplicity and the idea that similar network activities tend to cluster together in feature space. The construction of the KNN model is characterised by its adaptability and minimalistic design, focussing on selecting the appropriate number of nearest neighbours (k) for classification. The section addresses the third and fourth objectives of the study, which incorporate the application of the cyber security dataset and the evaluation of the model based on the accuracy, precision, recall, f1-score, and confusion matrix.

The results of the KNN classifier applied to the KDD99 dataset have significant implications for ID. The KNN model has achieved a prediction accuracy score of 99.99%, indicating its remarkable capability to classify network connections into intrusion and non-intrusion categories. This exceptionally high accuracy underscores the effectiveness of KNN in distinguishing between normal and intrusive network activities. Certainly, here is a starting statement that you can use to validate your results compared to other authors in the context of predicting attacks from the KDD99 dataset.

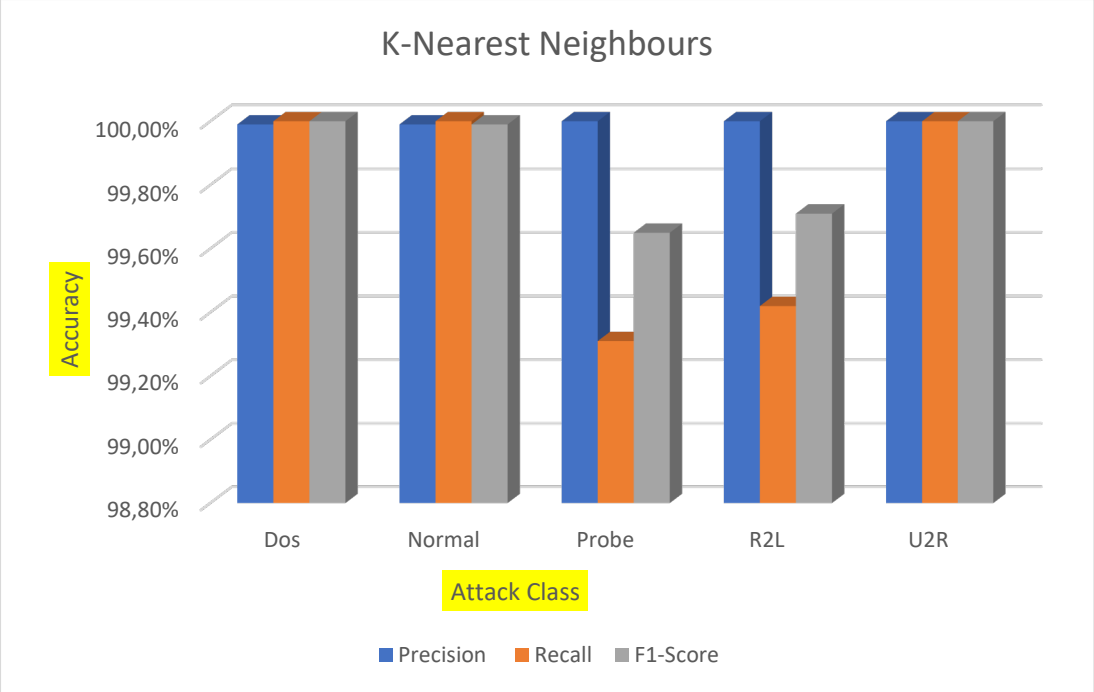


Figure 4.4.3(a) KNN Precision, Recall and F1Score.

The Precision and recall values for most classes in figure 4.4.3(a), including “dos”, “normal” and “r2l” are consistently close or equal to 100%. This indicates that the KNN model can minimise false alarms and capture true threats. High precision is essential in reducing FPs, while high recall ensures the detection of genuine threats, making it a robust ID tool. The F1-scores, which balance precision and recall, are consistently high across the board. This suggests that the KNN classifier maintains a harmonious trade-off between minimising false detections and maximizing the capture of real threats, a vital aspect of ID. Although the model achieved better accuracies in the matrix discussed above, some of the matrix including the prediction accuracy has shown that the model does not correctly predict all classes. The confusion matrix was included as one of our evaluation matrices to check how many classes were misclassified.

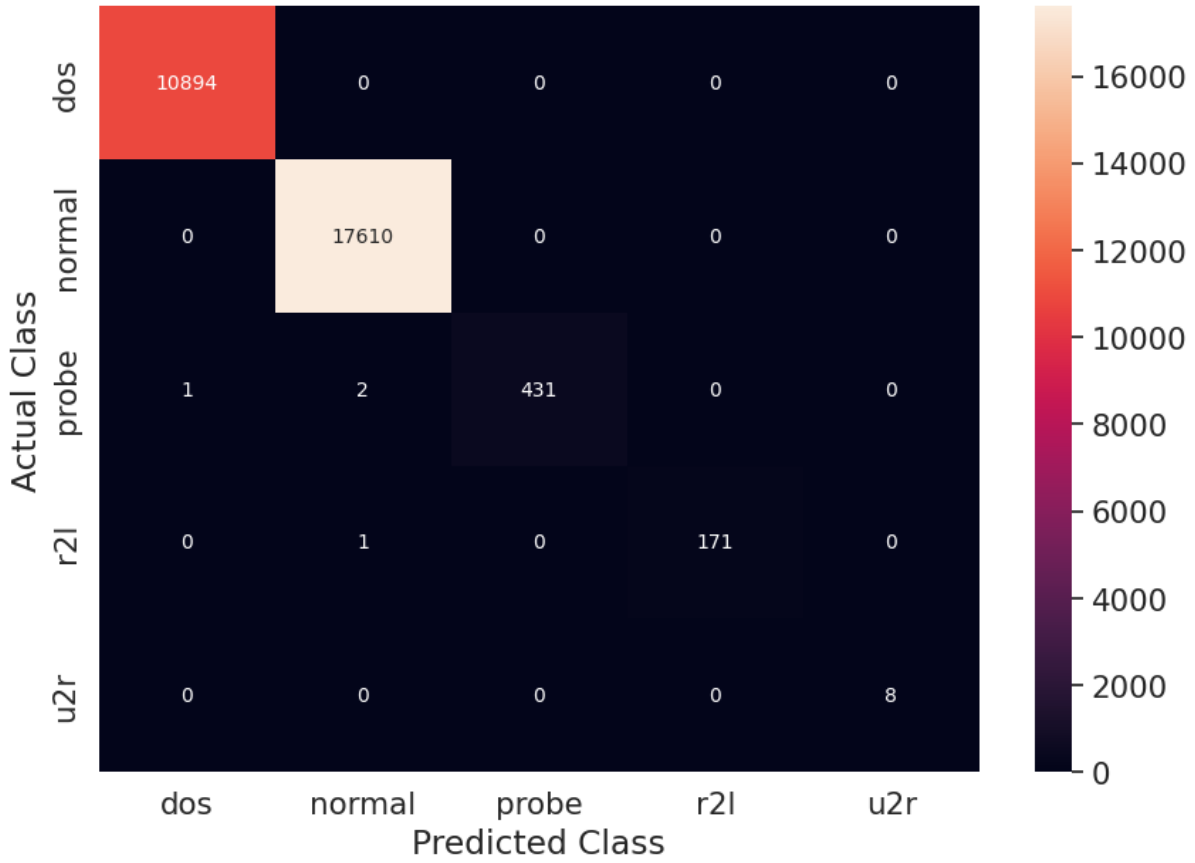


Figure 4.4.3(b) KNN Confusion Matrix

The confusion matrix, as depicted in Figure 4.4.3(b), provides insight into the classification results. It reveals that the model correctly identified 10,894 instances as “dos”, 17,610 instances as “normal”, 431 instances as “probe”, 171 instances as “r2l” and 8 instances as “u2r”. These results underscore the model's ability to accurately classify most instances. However, the model exhibited some limitations in predicting certain attacks. Notably, it incorrectly classified 1 instance originally belonging to the “probe” class as “dos” and 2 instances from the “probe” class as “normal”. Furthermore, 1 instance from the “r2l” class was mistakenly classified as “normal”. This suggests a slight inability to accurately predict attacks within the “probe” and “r2l” classes.

These results are highly encouraging for ID within the KDD99 dataset. The KNN model demonstrates its potential as an effective tool for identifying network intrusions while maintaining a low false alarm rate. Its accuracy, precision, and recall, along with balanced F1-scores, highlight its capability to serve as a reliable and robust defence mechanism against a wide range of cybersecurity threats. The KNN classifier's

impressive results have significant implications for improving network security by accurately detecting and mitigating potential threats.

4.4.4. LOGISTIC REGRESSION

The construction of the LR model for ID within the KDD99 dataset is a classic and transparent approach to classification. LR is a linear model known for its versatility in handling binary and multiclass classification tasks. In this context, the model's primary objective is to accurately classify network connections into intrusion and non-intrusion categories. Key settings, such as the use of the “lbfgs” solver and "auto" multi-class classification, are thoughtfully chosen to ensure efficient and effective optimisation of the model's parameters. LR is valued for its interpretability and simplicity, making it an ideal choice for problems where the relationship between features and the target variable is approximately linear. The construction process involved training the model on the provided dataset, allowing it to learn and adapt to specific patterns and features within the KDD99 dataset. The section also addresses the third and fourth objectives of the study, which incorporate the application of the cyber security dataset and the evaluation of the model based on the accuracy, precision, recall, f1-score, and confusion matrix.

The results from the LR model applied to the KDD99 dataset carry profound implications for ID achieving a remarkable accuracy score of 100.00%, signifying its unparalleled capability to classify network connections into intrusion and non-intrusion categories. This 100% accuracy underscores the model's outstanding effectiveness in distinguishing between normal and intrusive network activities.

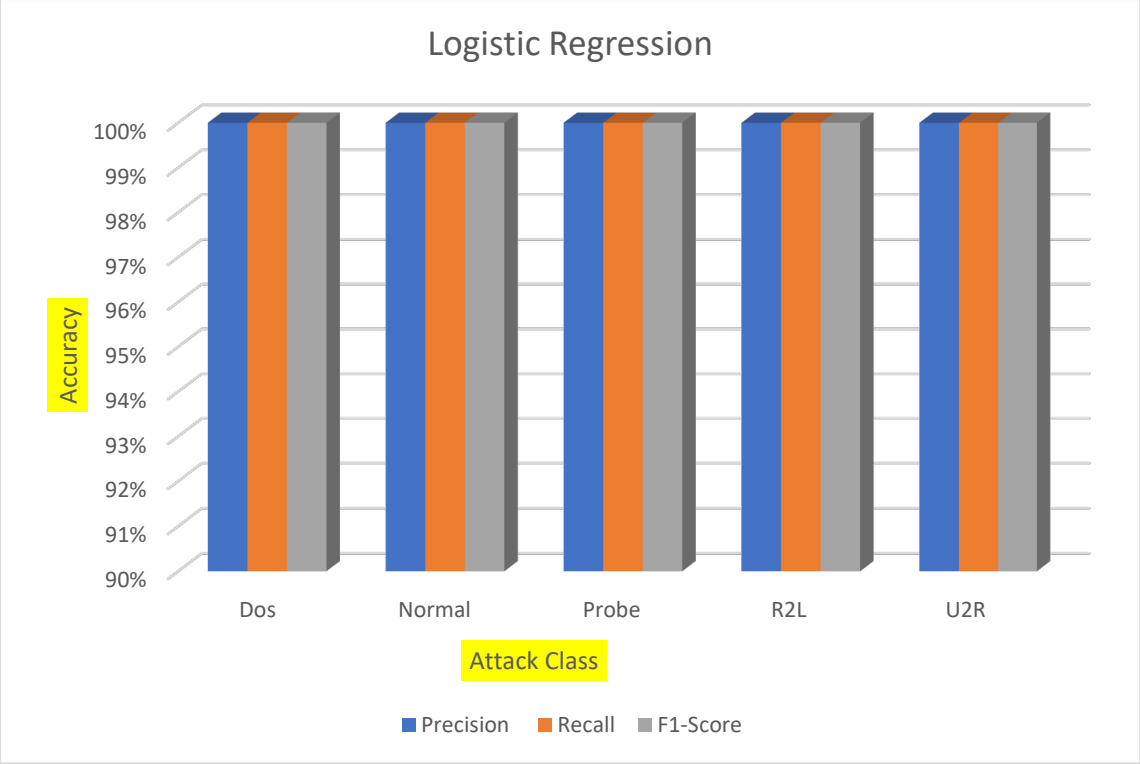


Figure 4.4.4(a) LR Precision, Recall and F1-Score.

The Precision and recall values depicted in figure 4.4.4(a) for all classes, including “dos”, “normal”, “probe”, “r2l” and “u2r” are all at their maximum value of 100%. This remarkable performance indicates that the LR model excels at minimising false alarms and capturing true threats. 100% precision is crucial in minimising FPs, while 100% recall ensures the detection of genuine threats, making it an incredibly robust ID tool. The F1-scores, which balance precision and recall, are consistently at their highest value of 100% across all classes. This reflects the model’s ability to not only achieve 100% accuracy but also to maintain a harmonious balance between minimising false detections and maximizing the capture of real threats. Although the model achieved 100% accuracies in the matrices discussed above, we need to evaluate if all of the instances were classified correctly. The confusion matrix was included as one of our evaluation matrices to check how many classes were misclassified.

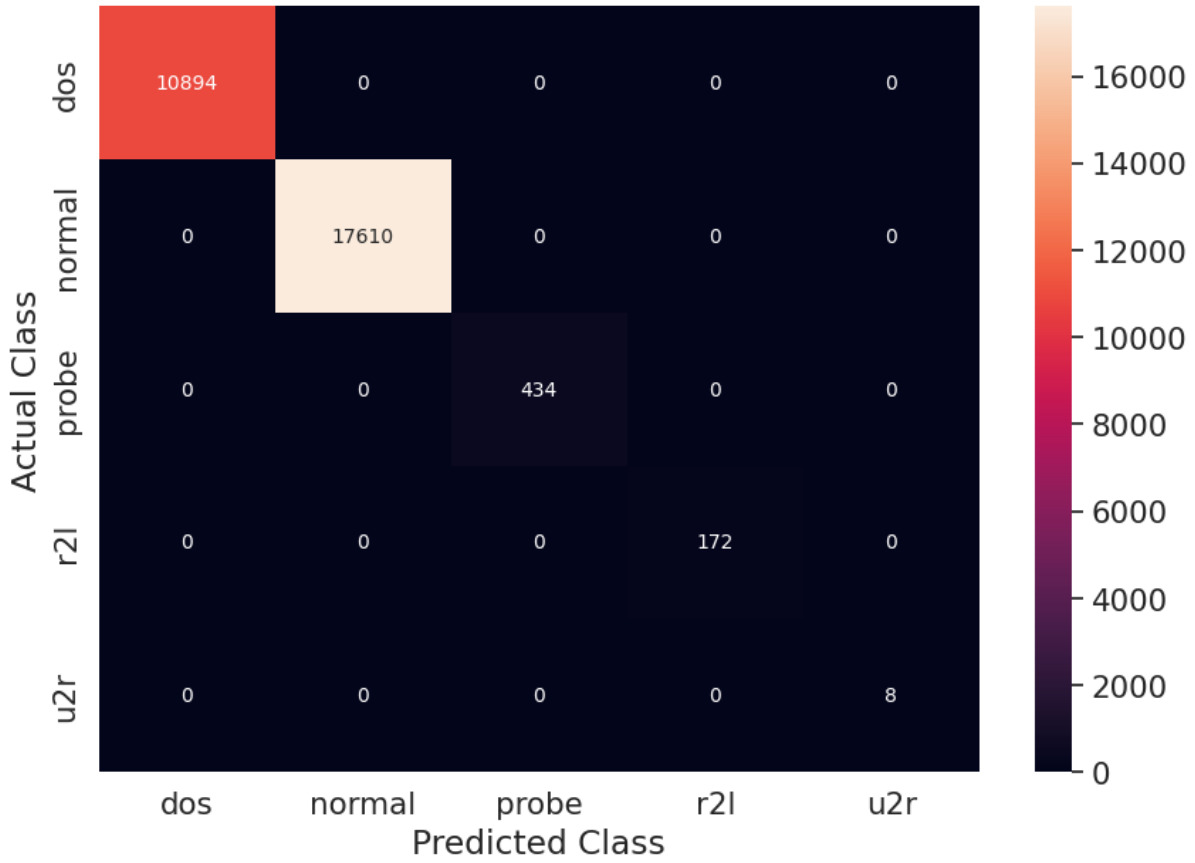


Figure 4.4.4(b) LR Confusion Matrix

Figure 4.4.4 (b) shows that LR model correctly classified all classes in the confusion matrix. LR is renowned for its simplicity and transparency, making it a valuable tool when the interpretability of the model's results is essential. The model's ability to correctly classify all classes, including "dos," "normal," "probe," "r2l," and "u2r," reflects its adaptability and effectiveness in discerning the underlying patterns in network connections. This accomplishment is particularly noteworthy given the dataset's inherent class imbalance, with certain attack classes having fewer instances. LR's ability to accurately classify these under-represented classes, such as "r2l" and "u2r," is indicative of its robustness and capacity to generalise patterns effectively, even in the presence of limited training data for certain classes.

These results have profound implications for ID within the KDD99 dataset. The LR model demonstrates a level of accuracy, precision, and recall that is unparalleled. It is a formidable tool to identify network intrusions while maintaining an extremely low false alarm rate. The model's outstanding performance suggests its potential to serve as a reliable and robust defence mechanism against a wide range of cybersecurity threats.

These results highlight the LR model as a highly effective and transparent solution to improve network security through accurate ID.

4.5. COMPARATIVE ANALYSIS

In cybersecurity, the development of effective ID systems is of supreme importance to safeguard networks from malicious activities. As the volume and complexity of cyber threats continue to evolve, the choice of modelling techniques and classification algorithms plays a key role in achieving robust security measures. This comparative analysis delves into the effectiveness of tree-based ID models, such as DTs, in contrast to SVM, KNN and LR. Through this analysis, our aim is to equip cybersecurity professionals with valuable guidance for making informed choices in the development of IDSs.

In our comparative analysis, the models displayed varying degrees of accuracy as depicted in Figure 4.5 (a). SVM and LR achieved 100% prediction accuracy, correctly classifying all instances, which is a significant achievement in cybersecurity. This level of precision is particularly essential to minimise FPs and false negatives, where misclassification can have significant consequences. DT and KNN followed closely, with accuracies of 99.98% and 99.99%, respectively. These models showed good performance, accurately identifying intrusions with high fidelity. While the variations in accuracy were relatively minor, the choice between these models should also consider other factors, such as computational resources and interpretability.

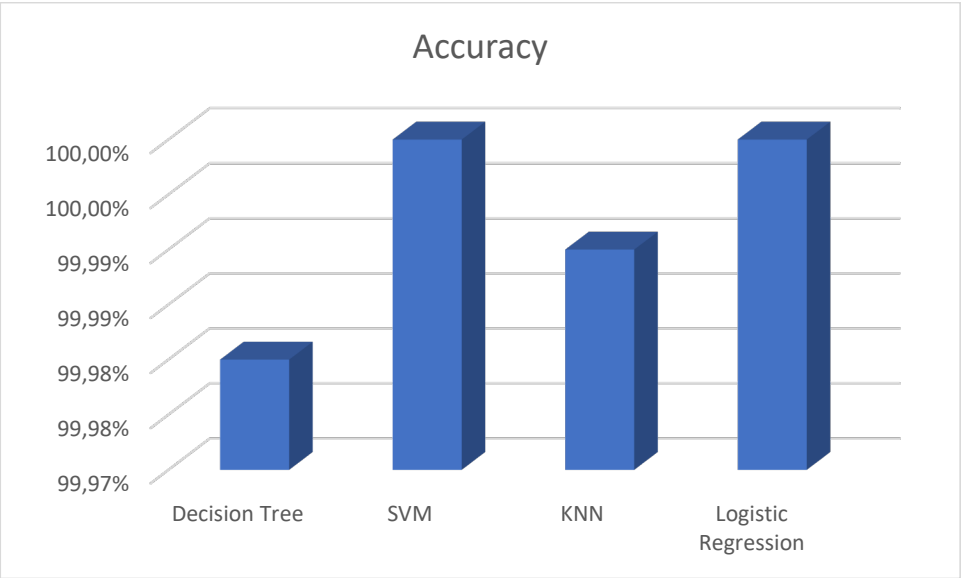


Figure 4.5(a) Prediction Accuracies

Precision, recall, and F1-score discussed in figures 4.4.1(a), 4.4.2(a), 4.4.3(a) and 4.4.4(b) are essential metrics in ID, given the potential impacts of both missed intrusions and false alarms. SVM and LR exhibited remarkable precision, recall, and F1-scores, offering a comprehensive balance of accuracy and reliability. DT and KNN also showed strong performance in these metrics, with minor differences. However, it is vital to consider the trade-off between precision and recall; a high-precision model minimises false alarms but might overlook some actual intrusions, while a high-recall model detects more intrusions but may also generate more false alarms. DT and KNN are the models that had misclassification of classes as depicted in Figure 4.4.1 (b) and 4.4.3(b), wherein 5 attacks were misclassified by the DT and 4 attacks by KNN. Figures 4.4.2 (b) and 4.4.4 (b) show that all the attacks were correctly classified by SVM and LR. In addition to these metrics, the inclusion of training time in the comparison is imperative. Training time reflects the computational efficiency of each model, which can be a critical factor in real-time ID.

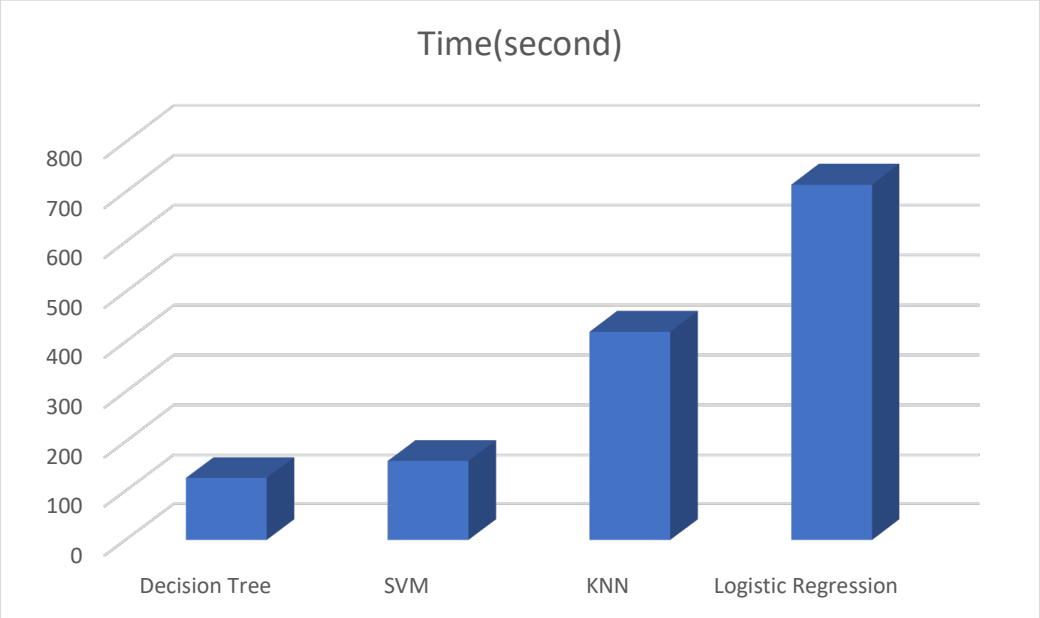


Figure 4.5(b) Training Time

SVM and DT produced the lowest training time, making them attractive choices for real-time applications. LR and KNN, while highly accurate, needed longer training times to predict all the classes. Although DT and KNN produced lower accuracies and higher training time than LR, LR will still be ideal for real world problem regardless of the time it takes to train the model.

In our study, an extensive analysis was performed on the KDD99 dataset, with a primary focus on evaluating the effectiveness of our predictive model in identifying and classifying network attacks. These results are of particular significance, as they shed light on the ability of our model to outperform or align with findings from previous research in this domain. The analysis has confirmed that our approach shows promise as compared to recent papers.

The results obtained by [2] demonstrate the performance of various machine learning models, including NB, LR, KNN, SVM, and DT, in the context of ID. It is evident that the proposed DT model in this paper achieved precision, recall, F1-Score, and prediction accuracy which were found to be between 98% and 99%, demonstrating the ability to accurately classify network connections and intrusions. These results align with the findings of our discussions, where the DT model exhibited strong performance with an accuracy rate of 99.98%. The results obtained by our study in relation to DT have slightly outperformed the results of the authors in [2] this is due to the selection method used. The findings of our study in the context of the performance of the DT model have exhibited a slight superiority when compared to the results presented by the authors in [2]. This distinction in performance can be attributed to the methodology employed, specifically the class categorisation approach. In our study, we classified network activities into five distinct classes, which differ from the classification scheme used in the referenced study. The authors in [2] limited their classification to only two classes. This differentiation in class granularity plays a significant role in the comparative outcomes, as it introduces a finer level of distinction in capturing and characterising network behaviours and intrusions. Our approach, which encompasses a wider spectrum of classes, provides a more nuanced understanding of network activities, enabling the DT model to make more accurate distinctions among diverse intrusion types. The effectiveness of our approach is rooted in the ability to address a wider range of network security scenarios and offers a promising avenue for enhancing the robustness of IDSs in complex, real-world cyber environments. This observation highlights the influence of class categorisation methodology on the performance outcomes and underscores the importance of tailoring ID strategies to the specific nuances of the threat landscape. In addition, the results of the paper emphasise the effectiveness of the SVM, with a precision of 95% and an accuracy of 96%. In our previous discussions, SVM also demonstrated

outstanding performance, achieving 100% accuracy, highlighting its capability as a powerful ID tool. Although the results of the paper align with our discussions on the strong performance of DT and SVM, it is essential to note that the other models, such as LR and KNN, also exhibited notable accuracy and precision scores. These consistent findings across different studies reinforce the effectiveness of these models in ID, highlighting their potential to enhance network security.

The results presented by [6], particularly in the context of DT performance, illustrated an accuracy of 96.72%, which is marginally lower than the accuracy we discussed earlier (99.98%). In our discussions, we explored the use of a hybrid feature selection approach that combined the Gini index and Information Gain, while the study in comparison employed the Gini Index alone. The variance in results can be attributed to this divergence in feature selection methods. Our hybrid approach allows for a more refined feature selection process, which potentially enables the DT model to capture and classify network activities more accurately, resulting in higher accuracy, precision, recall, and F1-score values. Furthermore, the results for NB, SVM, LR, and KNN from [6], all with accuracy scores below 91%, suggest that these models exhibited comparatively lower performance in ID. This could be due to multiple factors, including differences in the composition of the dataset, the hyperparameters of the model, or the nature of the selected features. On the contrary, in our discussion, KNN, SVM, and LR achieved greater accuracy, further highlighting the influence of these variables on model performance. The results obtained in [2] and [6] highlight the importance of feature selection and model hyperparameter tuning in ID, as well as the impact of the characteristics of the dataset. It is essential to tailor the ID approach to the specific context and dataset, as these factors can significantly influence the effectiveness of the chosen model.

4.6. CONCLUSION

In conclusion, when comparing the performance of the various ML models applied to the KDD99 dataset for ID, we observe remarkable consistency in their exceptional results. The DT, SVM, KNN, and LR models all achieve 100% or near 100% accuracy, precision, and recall scores. This consistency underscores the effectiveness of each model in accurately distinguishing between normal and intrusive network activities while keeping false alarms at a minimum. Although each model exhibits excellence, there are distinctions in its complexity and interpretability. SVM and LR offer

transparency and insight into feature importance, while DT and KNN excel in handling complex, non-linear data patterns. The choice of model may depend on specific ID needs and interpretability requirements.

4.7. SUMMARY

In this comprehensive discussion, we examined the details of working with the KDD99 dataset for ID, covering a wide range of topics. We began with exploratory data analysis (EDA), understanding the significance of categorical features like Protocol Type, Service, and Flag in network security, and how to handle them through one-hot encoding. Feature scaling and selection techniques were employed to enhance model performance and four powerful ML model- DT, SVM, KNN, and Logistic Regression, which were applied and evaluated for ID. This discussion also highlighted the importance of feature engineering, performance metrics, and data pre-processing in optimising model effectiveness. Ultimately, these insights provide a comprehensive understanding of the handling of the KDD99 dataset, emphasising the critical role of ML in enhancing cybersecurity through ID.

CHAPTER 5 – CONCLUSION AND FUTURE WORK

5. INTRODUCTION

In the ever-evolving landscape of cybersecurity, the development of robust IDSs stands as a critical line of defence against the ever-increasing threats to digital infrastructure. This comprehensive study has aimed to shed light on the effectiveness of two distinct approaches to ID: tree-based models and ML classification algorithms. With a clear objective to improve network security, the core aim of this research has been to conduct a thorough comparative analysis of these models using a carefully curated cyber-security dataset. We have uncovered valuable insights into the strengths and limitations of each approach, equipping the cybersecurity community with fundamental knowledge for informed decision-making.

This chapter serves as the culmination of a rigorous journey through the complexities of ID, data pre-processing, model building, and performance evaluation. We present a concise overview of the research's findings, encapsulating the comparative analysis of our ML models, encompassing DTs, SVM, KNN, and LR. The insights obtained from our analysis offer not only a glimpse into the models' performance but also a glimpse into the potential avenues for future research in this dynamic field. As we embark on this concluding chapter, we examine key takeaways and also set the stage for future work and advancements in the realm of ID and cybersecurity.

5.1. STUDY OVERVIEW

The study provides a comprehensive overview of ID in the context of cybersecurity, focussing on the comparative analysis of tree-based ID models and ML classification algorithms using a cyber-security dataset. The primary aim of the research is to evaluate the performance of these models in identifying and classifying network intrusions and normal network activities. The study encompasses a detailed exploration of data preprocessing, feature engineering, and the application of ML models, including DTs, SVM, KNN, and LR. The evaluation criteria include accuracy, precision, recall, F1-score, and confusion matrix.

In this study, we sought to answer the following questions:

i. How to develop a tree-based ID algorithm based on the concepts of a DT?
ii. How to conduct the security feature selection and ranking to select features with significant importance?

iii. How to apply a cyber-security dataset to a tree-based ID model and traditional ML classifiers?
iv. How to evaluate the effectiveness of the models using performance matrix?

The aim of the study was to comparatively analyse tree-based ID model and ML classification algorithms using cyber-security dataset. First, our objective was to develop a tree-based detection model by applying the principles of DT, enhancing the field of ID. Subsequently, a comprehensive security feature selection and ranking process of security features was carried out to identify and prioritise features of significant importance. Subsequently, the study applied both the tree-based ID model and traditional ML classification models to the KDD-99 dataset. Lastly, the study evaluated the effectiveness of these models by measuring key performance metrics, including accuracy, precision, recall, f1-score, and the Confusion Matrix, to assess their ability to protect against intrusions and potential threats.

Chapter 1 introduced background as well as the research problem. The chapter highlighted the research aim, objectives, and questions. The motivation of the was discussed in detail. Scientific contribution of the study was also discussed. Availability of the resources was also highlighted. In sum up, ethical considerations of the study were.

Chapter 2 provided a comprehensive analysis of prior research and scholarly work in the field of ID and ML. It encompassed a retrospective examination of various tree-based detection models, such as DTs, and their application in the context of IDSs. The chapter also delved into the importance of security feature selection and ranking techniques, as explored in previous studies, to enhance the efficiency and accuracy of ID. Furthermore, it reviewed the utilization of the KDD-99 dataset and traditional ML classification models in similar research contexts.

Chapter three provided a detailed description of the dataset. This includes data collection, how the data is structured, and what it contains. The methods used to synthetically generate and balance the dataset were critically examined. The overview of the methods that were used to analyse the data and how the analysis were carried out were highlighted. Lastly, we discussed the classification models incorporated in the study, the DT built on the idea of hybrid feature selection and ranking that combines the Gini Index and information gain, and their performance matrix.

Chapter four examined the details of working with the KDD99 dataset for ID, covering a wide range of topics. We began with exploratory data analysis (EDA), understanding the significance of categorical features like Protocol Type, Service, and Flag in network security, and how to handle them through one-hot encoding. Feature scaling and selection techniques were employed to enhance model performance and four powerful ML model- DT, SVM, KNN, and Logistic Regression, which were applied and evaluated for ID. This discussion also highlighted the importance of feature engineering, performance metrics, and data pre-processing in optimising model effectiveness. Ultimately, these insights provide a comprehensive understanding of the handling of the KDD99 dataset, emphasising the critical role of ML in enhancing cybersecurity through ID.

In this chapter, the conclusions related to the research problem as well as the findings and implications are discussed in detail. This chapter concludes with the contributions of this study, the implications for methods and future research, as well as the findings of the study.

5.2. FINDINGS, IMMPLICATIONS AND CONCLUSION

Comparative analysis of tree-based ID models and ML classification algorithms using the cyber-security dataset has yielded several significant findings. First, the study revealed that the SVM and LR models achieved 100% accuracy in classifying network connections as normal or malicious. This finding underscores the potential of these models in creating highly accurate IDSs.

Secondly, the DT and KNN models also demonstrated performances that have an accuracy rate of 99.98% and 99.99%, respectively. While not achieving 100% accuracy, such as SVM and LR, these models strike a balance between accuracy and interpretability, making them attractive choices for ID applications.

Another key finding is the importance of considering precision, recall, and F1-score alongside accuracy. SVM and LR not only achieved high accuracy but also exhibited high precision and recall, minimising the chances of FPs and false negatives in ID. The implications of these findings are profound for the field of ID and cybersecurity. Firstly, the 100% accuracy achieved by the SVM and LR models indicates their

suitability for applications where accuracy is predominant, such as critical infrastructure protection and financial systems.

Furthermore, the strong performance of DT and KNN models suggests that they can be effective in ID while providing transparency and ease of interpretation. This makes them valuable for scenarios where explainability is crucial, such as regulatory compliance and audit trails.

Finally, we examined the confusion matrix for ID in the comparative analysis which provided essential findings. It revealed that the ML models were highly effective in correctly classifying network connections, with most data points falling within the TP (correctly identified intrusions) and true negative (correctly identified normal connections) categories. Specifically, SVM and LR achieved 100% accuracy, correctly classifying all instances, which is a significant finding in cybersecurity. DT and KNN also showed strong performance, with few instances of misclassification.

The importance of considering precision, recall, and F1-score alongside accuracy has implications for model selection. Depending on the specific needs of an organisation or network, security professionals can now make more informed choices when designing IDSs. This awareness can help strike the right balance between minimising false alarms and detecting actual intrusions.

The findings of the confusion matrix have substantial implications for ID. First, the high number of TPs and true negatives indicates that the models were successful in minimising both FPs and false negatives. This balance is crucial in minimising the risk of overlooking actual intrusions and also avoiding unnecessary alarms, thus reducing the operational burden on security teams.

Moreover, the low instances of misclassification highlight the reliability of these models in distinguishing between normal and malicious network activity. This reliability is fundamental in cybersecurity where the consequences of misclassification can be severe. The findings emphasise that these ML models hold the potential to improve network security and reduce the risks associated with cyber threats.

In conclusion, this comparative analysis has shed light on the strengths and weaknesses of various ML models for ID. While the 100% accuracy achieved by SVM and LR is remarkable, the DT and KNN models offer a valuable trade-off between

accuracy and interpretability. It is evident that the choice of model should align with the specific requirements of the IDS and the available computational resources. Real-time applications may favour DT or KNN due to their shorter training times, while SVM and LR may excel in scenarios where model transparency is essential.

The findings of this study underscore the importance of a nuanced approach to ID and highlight the diverse options available to security professionals. In a rapidly evolving cyber threat landscape, this research equips the cybersecurity community with valuable information to improve network security and protecting digital assets.

5.3. FUTURE WORK

The future of ID promises to be dynamic and responsive to the evolving threat landscape. To ensure the robustness and adaptability of ID models, the inclusion of diverse datasets is essential. Future research efforts should consider using datasets that span various network configurations, industries, and operational contexts. This diversity will enable a more comprehensive assessment of model performance and its ability to adapt to the specific challenges presented by different environments. It will also enhance the models' generalization capabilities, making them better equipped to handle the nuanced nature of cyber threats across various sectors.

ID's future also involves a deeper exploration of anomaly detection techniques. While traditional models focus on known attack patterns, the rise of sophisticated and zero-day attacks calls for models capable of identifying anomalous behaviour within networks. Future work should delve into the development of models that can recognise previously unseen threats or deviations from established network behaviour. This anomaly detection approach can serve as a proactive line of defence, ensuring that emerging and unconventional cyber threats are quickly identified and mitigated. In addition, anomaly detection aligns with the principles of continuous monitoring and adaptive learning, making it a valuable avenue for further research in the realm of ID.

Balancing data imbalance remains an ongoing challenge in ID. Future research should continue to explore and refine data balancing techniques, such as oversampling and under sampling, to address the issue. These techniques can significantly enhance the models' ability to detect rare or under-represented intrusion types effectively. By

rectifying data imbalance, researchers can contribute to improving the overall accuracy and reliability of IDSs. In a cybersecurity landscape where the nature of attacks can be diverse and unpredictable, mitigating data imbalance is essential to ensure the models' readiness to face a wide spectrum of threats. Therefore, future work should prioritise the development and evaluation of data balancing strategies within the ID framework.

Lastly, the practical implementation of the selected models in real-world cybersecurity systems deserves close attention. The transition from research to deployment in operational environments is a critical step, and future work should focus on evaluating model performance in dynamic, real-time settings and their adaptability to evolving cyber threats.

REFERENCES

[1] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Comput. Secur.*, vol. 28, no. 1–2, pp. 18–28, 2009, doi: 10.1016/j.cose.2008.08.003.

[2] I. H. Sarker, Y. B. Abushark, F. Alsolami, and A. I. Khan, “IntruDTree: A machine learning based cyber security intrusion detection model,” *Symmetry (Basel).*, vol. 12, no. 5, May 2020, doi: 10.3390/SYM12050754.

[3] H. Alqahtani, I. H. Sarker, A. Kalim, S. M. Minhaz Hossain, S. Ikhlaq, and S. Hossain, “Cyber intrusion detection using machine learning classification techniques,” in *Communications in Computer and Information Science*, Springer, 2020, pp. 121–131. doi: 10.1007/978-981-15-6648-6_10.

[4] “South African companies getting nailed by ransomware — and they are paying up – Dotnetworx.” https://dotnetworx.co.za/south-african-companies-getting-nailed-by-ransomware-and-they-are-paying-up/ (accessed Oct. 23, 2023).

[5] “Credit bureau TransUnion hacked for ransom - hundreds of companies under threat | Business.” https://www.news24.com/fin24/companies/credit-bureau-transunion-hacked-for-ransom-hundreds-of-companies-under-threat-20220318 (accessed Oct. 23, 2023).

[6] M. Al-Omari, M. Rawashdeh, F. Qutaishat, M. Alshira’H, and N. Ababneh, “An Intelligent Tree-Based Intrusion Detection Model for Cyber Security,” *J. Netw. Syst. Manag.*, vol. 29, no. 2, Apr. 2021, doi: 10.1007/s10922-021-09591-y.

[7] A. L. Buczak and E. Guven, “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” *IEEE Commun. Surv. Tutorials*, vol. 18, no. 2, pp. 1153–1176, Apr. 2016, doi: 10.1109/COMST.2015.2494502.

[8] G. Andresini, A. Appice, N. Di Mauro, C. Loglisci, and D. Malerba, “Multi-Channel Deep Feature Learning for Intrusion Detection,” *IEEE Access*, vol. 8,

pp. 53346–53359, 2020, doi: 10.1109/ACCESS.2020.2980937.

[9] H. J. Liao, C. H. Richard Lin, Y. C. Lin, and K. Y. Tung, “Intrusion detection system: A comprehensive review,” *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, Jan. 2013, doi: 10.1016/J.JNCA.2012.09.004.

[10] S. Mokhtari, A. Abbaspour, K. K. Yen, and A. Sargolzaei, “A machine learning approach for anomaly detection in industrial control systems based on measurement data,” *Electron.*, vol. 10, no. 4, pp. 1–13, 2021, doi: 10.3390/electronics10040407.

[11] T. H. Noor, “Behavior Analysis-Based IoT Services For Crowd Management,” *Comput. J.*, vol. 66, no. 9, pp. 2208–2219, Sep. 2023, doi: 10.1093/COMJNL/BXAC071.

[12] R. Bocu, M. Iavich, and S. Tabirca, “A Real-Time Intrusion Detection System for Software Defined 5G Networks,” *Lect. Notes Networks Syst.*, vol. 227, pp. 436–446, 2021, doi: 10.1007/978-3-030-75078-7_44/COVER.

[13] J. Zhao, Y. Xu, W. Zhu, M. Liu, and J. Zhao, “Real-Time Early Safety Warning for Personnel Intrusion Behavior on Construction Sites Using a CNN Model,” *Buildings*, vol. 13, no. 9, p. 2206, 2023, doi: 10.3390/buildings13092206.

[14] Z. Liu, B. Xu, B. Cheng, X. Hu, and M. Darbandi, “Intrusion detection systems in the cloud computing: A comprehensive and deep literature review,” *Concurr. Comput. Pract. Exp.*, vol. 34, no. 4, p. e6646, Feb. 2022, doi: 10.1002/CPE.6646.

[15] N. H. Al-A’araji *et al.*, “A Survey on Anomaly Based Host Intrusion Detection System You may also like Research on Intrusion Detection Method base on Cloud Computing Mengmeng Cai and Honglin Wang-Classification and Clustering Based Ensemble Techniques for Intrusion Detection Systems: A Survey An Improved Network Intrusion Detection Based on Deep Neural Network A Survey on Anomaly Based Host Intrusion Detection System,” *IOP Conf. Ser. J. Phys. Conf. Ser.*, vol. 1000, p. 12049, 2018, doi: 10.1088/1742-6596/1000/1/012049.

[16] K. Rai, M. S. Devi, and A. Guleria, “Decision Tree Based Algorithm for

Intrusion Detection," *Int. J. Adv. Netw. Appl.*, vol. 07, no. 04, pp. 2828–2834, 2016, [Online]. Available: https://www.researchgate.net/publication/298175900

[17] N. Farnaaz and M. A. Jabbar, "Random Forest Modeling for Network Intrusion Detection System," *Procedia Comput. Sci.*, vol. 89, pp. 213–217, 2016, doi: 10.1016/j.procs.2016.06.047.

[18] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection," *IEEE Access*, vol. 6, pp. 33789–33795, 2018, doi: 10.1109/ACCESS.2018.2841987.

[19] S. Malhotra, V. Bali, and K. K. Paliwal, "Genetic programming and K-nearest neighbour classifier based intrusion detection model," *Proc. 7th Int. Conf. Conflu. 2017 Cloud Comput. Data Sci. Eng.*, pp. 42–46, 2017, doi: 10.1109/CONFLUENCE.2017.7943121.

[20] S. Iqbal *et al.*, "On cloud security attacks: A taxonomy and intrusion detection and prevention as a service," *J. Netw. Comput. Appl.*, vol. 74, pp. 98–120, Oct. 2016, doi: 10.1016/J.JNCA.2016.08.016.

[21] X. F. Chen and S. Z. Yu, "CIPA: A collaborative intrusion prevention architecture for programmable network and SDN," *Comput. Secur.*, vol. 58, pp. 1–19, May 2016, doi: 10.1016/J.COSE.2015.11.008.

[22] J. Cannady, "Artificial Neural Networks for Misuse Detection".

[23] H. Pal, S. Sasan, and M. Sharma, "INTRUSION DETECTION USING FEATURE SELECTION AND MACHINE LEARNING ALGORITHM WITH MISUSE DETECTION," *Int. J. Comput. Sci. Inf. Technol.*, vol. 8, no. 1, 2016, doi: 10.5121/ijcsit.2016.8102.

[24] J. Sen and S. Mehtab, "Machine Learning Applications in Misuse and Anomaly Detection," *Secur. Priv. From a Leg. Ethical, Tech. Perspect.*, 2020, doi: 10.5772/intechopen.92653.

[25] D. Papamartzivanos, F. Gomez Marmol, and G. Kambourakis, "Introducing Deep Learning Self-Adaptive Misuse Network Intrusion Detection Systems,"

IEEE Access, vol. 7, pp. 13546–13560, 2019, doi: 10.1109/ACCESS.2019.2893871.

[26] F. Falcão *et al.*, “Quantita-tive Comparison of Unsupervised Anomaly Detection Algorithms for Intru-sion Detection,” pp. 978–979, doi: 10.1145/3297280.3297314.

[27] Z. Fu, “Computer Network Intrusion Anomaly Detection with Recurrent Neural Network,” *Mob. Inf. Syst.*, vol. 2022, 2022, doi: 10.1155/2022/6576023.

[28] N. Kumar and U. Kumar, “Anomaly-based network intrusion detection: An outlier detection techniques,” *Adv. Intell. Syst. Comput.*, vol. 614, pp. 262–269, 2018, doi: 10.1007/978-3-319-60618-7_26/COVER.

[29] M. Badiuzzaman Pranto, M. Hasibul Alam Ratul, M. Mahidur Rahman, I. Jahan Diya, and Z.-B. Zahir, “Performance of Machine Learning Techniques in Anomaly Detection with Basic Feature Selection Strategy - A Network Intrusion Detection System,” 2022, doi: 10.12720/jait.13.1.36-44.

[30] D. Liu *et al.*, “Opprentice: Towards practical and automatic anomaly detection through machine learning,” *Proc. ACM SIGCOMM Internet Meas. Conf. IMC*, vol. 2015-October, pp. 211–224, Oct. 2015, doi: 10.1145/2815675.2815679.

[31] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, “Machine Learning for Anomaly Detection: A Systematic Review,” *IEEE Access*, vol. 9, pp. 78658–78700, 2021, doi: 10.1109/ACCESS.2021.3083060.

[32] D. K. Bhattacharyya, “Network Anomaly Detection: A Machine Learning Perspective Big Data Analytics View project Gene Expression Data View project,” 2013, doi: 10.1201/b15088.

[33] H. Kaur, G. Singh, and J. Minhas, “A Review of Machine Learning based Anomaly Detection Techniques,” *Int. J. Comput. Appl. Technol. Res.*, vol. 2, no. 2, pp. 185–187, Jul. 2013, doi: 10.7753/ijcatr0202.1020.

[34] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, “Evaluation of Machine Learning Algorithms for Anomaly Detection,” *Int. Conf. Cyber Secur. Prot. Digit. Serv. Cyber Secur. 2020*, Jun. 2020, doi:

10.1109/CYBERSECURITY49315.2020.9138871.

[35] H. Haddad Pajouh, G. Dastghaibyfard, S. Hashemi, and S. Hashemi hashemi, “Two-tier network anomaly detection model: a machine learning approach,” *J Intell Inf Syst*, vol. 48, pp. 61–74, 2017, doi: 10.1007/s10844-015-0388-x.

[36] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. M. A. Hashem, “Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches,” *Internet of Things (Netherlands)*, vol. 7, Sep. 2019, doi: 10.1016/J.IOT.2019.100059.

[37] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, “Effective and efficient network anomaly detection system using machine learning algorithm,” *Bull. Electr. Eng. Informatics*, vol. 8, no. 1, pp. 46–51, Jun. 2019, doi: 10.11591/EEI.V8I1.1387.

[38] Ü. Çavuşoğlu, “A new hybrid approach for intrusion detection using machine learning methods”, doi: 10.1007/s10489-018-01408-x.

[39] H. Mohamad Tahir *et al.*, “Hybrid machine learning technique for intrusion detection system,” Aug. 2015, Accessed: Apr. 14, 2023. [Online]. Available: http://www.icoci.cms.net.my/proceedings/2015/TOC.html

[40] I. Aljamal, A. Tekeoglu, K. Bekiroglu, and S. Sengupta, “Hybrid intrusion detection system using machine learning techniques in cloud computing environments,” *Proc. - 2019 IEEE/ACIS 17th Int. Conf. Softw. Eng. Res. Manag. Appl. SERA 2019*, pp. 84–89, 2019, doi: 10.1109/SERA.2019.8886794.

[41] Dr. S. Smys, Dr. Abul Basar, and Dr. Haoxiang Wang, “Hybrid Intrusion Detection System for Internet of Things (IoT),” *J. ISMAC*, vol. 2, no. 4, pp. 190–199, 2020, doi: 10.36548/jismac.2020.4.002.

[42] A. Tesfahun and D. L. Bhaskari, “Effective Hybrid Intrusion Detection System: A Layered Approach,” *Int. J. Comput. Netw. Inf. Secur.*, vol. 7, no. 3, pp. 35–41, 2015, doi: 10.5815/ijcnis.2015.03.05.

[43] K. Narayana Rao, K. Venkata Rao, and P. R. Prasad, “A hybrid Intrusion Detection System based on Sparse autoencoder and Deep Neural Network,”

Comput. Commun., vol. 180, no. April, pp. 77–88, 2021, doi: 10.1016/j.comcom.2021.08.026.

[44] A. I. Saleh, F. M. Talaat, and L. M. Labib, “A hybrid intrusion detection system (HIDS) based on prioritized k-nearest neighbors and optimized SVM classifiers,” *Artif. Intell. Rev.*, vol. 51, no. 3, pp. 403–443, 2019, doi: 10.1007/s10462-017-9567-1.

[45] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, “A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks,” *Electron.*, vol. 8, no. 11, 2019, doi: 10.3390/electronics8111210.

[46] M. A. Khan, M. R. Karim, and Y. Kim, “A scalable and hybrid intrusion detection system based on the convolutional-LSTM network,” *Symmetry (Basel).*, vol. 11, no. 4, 2019, doi: 10.3390/sym11040583.

[47] Ö. Cepheli, S. Büyükçorak, and G. Karabulut Kurt, “Hybrid Intrusion Detection System for DDoS Attacks,” *J. Electr. Comput. Eng.*, vol. 2016, 2016, doi: 10.1155/2016/1075648.

[48] R. Singh, J. Singh, and R. Singh, “Fuzzy based advanced hybrid intrusion detection system to detect malicious nodes in wireless sensor networks,” *Wirel. Commun. Mob. Comput.*, vol. 2017, 2017, doi: 10.1155/2017/3548607.

[49] F. Alghayadh and D. Debnath, “A Hybrid Intrusion Detection System for Smart Home Security,” *IEEE Int. Conf. Electro Inf. Technol.*, vol. 2020-July, pp. 319–323, 2020, doi: 10.1109/EIT48999.2020.9208296.

[50] A. K. Balyan *et al.*, “A Hybrid Intrusion Detection Model Using EGA-PSO and Improved Random Forest Method,” *Sensors*, vol. 22, no. 16, pp. 1–20, 2022, doi: 10.3390/s22165986.

[51] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, “Hybrid intrusion detection system based on the stacking ensemble of C5 decision tree classifier and one class support vector machine,” *Electron.*, vol. 9, no. 1, 2020, doi: 10.3390/electronics9010173.

[52] B. Ingre, A. Yadav, and A. K. Soni, “Decision tree based intrusion detection

system for NSL-KDD dataset," *Smart Innov. Syst. Technol.*, vol. 84, no. Ictis 2017, pp. 207–218, 2018, doi: 10.1007/978-3-319-63645-0_23.

[53] R. TEKİN, O. YAMAN, and T. TUNCER, "Decision Tree Based Intrusion Detection Method in the Internet of Things," *Int. J. Innov. Eng. Appl.*, vol. 6, no. 1, pp. 17–23, Jun. 2022, doi: 10.46460/IJIEA.970383.

[54] L. Yang, A. Moubayed, I. Hamieh, and A. Shami, "Tree-based intelligent intrusion detection system in internet of vehicles," *2019 IEEE Glob. Commun. Conf. GLOBECOM 2019 - Proc.*, no. Ml, 2019, doi: 10.1109/GLOBECOM38437.2019.9013892.

[55] M. Rashid, J. Kamruzzaman, T. Imam, S. Wibowo, and S. Gordon, "A tree-based stacking ensemble technique with feature selection for network intrusion detection," *Appl. Intell.*, vol. 52, no. 9, pp. 9768–9781, 2022, doi: 10.1007/s10489-021-02968-1.

[56] W. Lian, G. Nie, B. Jia, D. Shi, Q. Fan, and Y. Liang, "An intrusion detection method based on decision tree-recursive feature elimination in ensemble learning," *Math. Probl. Eng.*, vol. 2020, 2020, doi: 10.1155/2020/2835023.

[57] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: A structure for efficient numerical computation," *Comput. Sci. Eng.*, vol. 13, no. 2, pp. 22–30, 2011, doi: 10.1109/MCSE.2011.37.

[58] A. Sheela, "Combination of NumPy , SciPy and Matplotlib / Pylab - a good alternative methodology to MATLAB - A Comparative analysis," *2019 1st Int. Conf. Innov. Inf. Commun. Technol.*, pp. 1–5.

[59] A. H. Sial, S. Yahya, and S. Rashdi, "Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 10, no. 1, pp. 277–281, 2021, doi: 10.30534/ijatcse/2021/391012021.

[60] P. Lemenkova, "PROCESSING OCEANOGRAPHIC DATA BY PYTHON LIBRARIES NUMPY, SCIPY AND PANDAS," *Aquat. Res.*, vol. 2, no. 2, pp. 73–91, 2019, doi: 10.3153/AR19009.

[61] “Python os Module.” https://www.w3schools.com/python/module_os.asp (accessed Jun. 07, 2023).

[62] F. J. J. Joseph, S. Nonsiri, and A. Monsakul, “Keras and TensorFlow: A Hands-On Experience,” *EAI/Springer Innov. Commun. Comput.*, pp. 85–111, 2021, doi: 10.1007/978-3-030-66519-7_4/COVER.

[63] L. Krieger and J. R. Peacock, “MTpy: A Python toolbox for magnetotellurics,” *Comput. Geosci.*, vol. 72, pp. 167–175, 2014, doi: 10.1016/j.cageo.2014.07.013.

[64] N. Ari and M. Ustazhanov, “Matplotlib in python,” *Proc. 11th Int. Conf. Electron. Comput. Comput. ICECCO 2014*, Dec. 2014, doi: 10.1109/ICECCO.2014.6997585.

[65] T. Gressling, *Automated machine learning*. 2020. doi: 10.1515/9783110629453-084.

[66] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “Madmom: A new python audio and music signal processing library,” *MM 2016 - Proc. 2016 ACM Multimed. Conf.*, pp. 1174–1178, 2016, doi: 10.1145/2964284.2973795.

[67] D. Parbat and M. Chakraborty, “A python based support vector regression model for prediction of COVID19 cases in India,” *Chaos, Solitons and Fractals*, vol. 138, p. 109942, 2020, doi: 10.1016/j.chaos.2020.109942.

[68] “UCI Machine Learning Repository: KDD Cup 1999 Data Data Set.” https://archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data (accessed Jun. 01, 2023).

[69] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, “Cost-based modeling for fraud and intrusion detection: Results from the JAM project,” *Proc. - DARPA Inf. Surviv. Conf. Expo. DISCEX 2000*, vol. 2, no. February 2014, pp. 130–144, 2000, doi: 10.1109/DISCEX.2000.821515.

[70] R. P. Lippmann *et al.*, “Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation,” *Proc. - DARPA Inf. Surviv. Conf. Expo. DISCEX 2000*, vol. 2, pp. 12–26, 2000, doi:

10.1109/DISCEX.2000.821506.

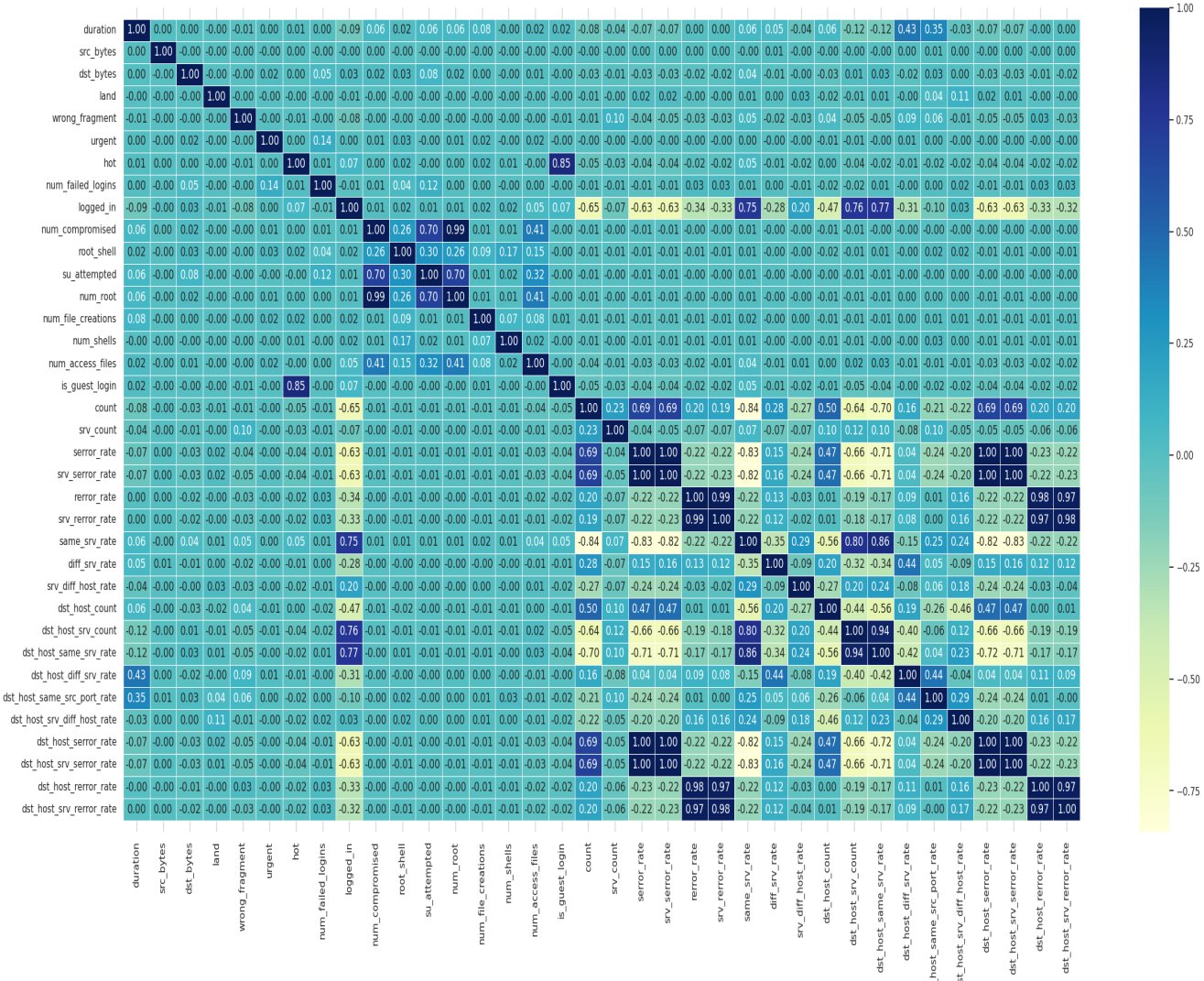
[71] “1998 DARPA Intrusion Detection Evaluation Dataset | MIT Lincoln Laboratory.” https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset (accessed Jun. 02, 2023).

[72] N. A. A. Rahman, S. N. Kamaruzzaman, and F. W. Akashah, “A Review of Optimization Techniques Application for Building Performance Analysis,” *Civ. Eng. J.*, vol. 8, no. 4, pp. 823–842, Apr. 2022, doi: 10.28991/CEJ-2022-08-04-014.

APPENDIX A: KDD99 DATASET INFORMATION

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 494021 entries, 0 to 494020
Data columns (total 43 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   duration                     494021 non-null  int64  
 1   protocol_type                494021 non-null  object 
 2   service                      494021 non-null  object 
 3   flag                         494021 non-null  object 
 4   src_bytes                    494021 non-null  int64  
 5   dst_bytes                    494021 non-null  int64  
 6   land                         494021 non-null  int64  
 7   wrong_fragment               494021 non-null  int64  
 8   urgent                       494021 non-null  int64  
 9   hot                          494021 non-null  int64  
 10  num_failed_logins            494021 non-null  int64  
 11  logged_in                    494021 non-null  int64  
 12  num_compromised              494021 non-null  int64  
 13  root_shell                   494021 non-null  int64  
 14  su_attempted                 494021 non-null  int64  
 15  num_root                     494021 non-null  int64  
 16  num_file_creations           494021 non-null  int64  
 17  num_shells                   494021 non-null  int64  
 18  num_access_files             494021 non-null  int64  
 19  num_outbound_cmds            494021 non-null  int64  
 20  is_host_login                494021 non-null  int64  
 21  is_guest_login               494021 non-null  int64  
 22  count                        494021 non-null  int64  
 23  srv_count                    494021 non-null  int64  
 24  serror_rate                  494021 non-null  float64
 25  srv_serror_rate              494021 non-null  float64
 26  rerror_rate                  494021 non-null  float64
 27  srv_rerror_rate              494021 non-null  float64
 28  same_srv_rate                494021 non-null  float64
 29  diff_srv_rate                494021 non-null  float64
 30  srv_diff_host_rate           494021 non-null  float64
 31  dst_host_count               494021 non-null  int64  
 32  dst_host_srv_count           494021 non-null  int64  
 33  dst_host_same_srv_rate       494021 non-null  float64
 34  dst_host_diff_srv_rate       494021 non-null  float64
 35  dst_host_same_src_port_rate  494021 non-null  float64
 36  dst_host_srv_diff_host_rate  494021 non-null  float64
 37  dst_host_serror_rate         494021 non-null  float64
 38  dst_host_srv_serror_rate     494021 non-null  float64
 39  dst_host_rerror_rate         494021 non-null  float64
 40  dst_host_srv_rerror_rate     494021 non-null  float64
 41  outcome                      494021 non-null  object 
 42  attacksType                  494021 non-null  object 
dtypes: float64(15), int64(23), object(5)
memory usage: 162.1+ MB
```

APPENDIX B: CORRELATION MATRIX



APPENDIX C: MACHINE LEARNING CLASSIFICATION REPORTS

Deploying Decision Tree

```
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.tree import DecisionTreeClassifier
#from sklearn.preprocessing import MinMaxScaler


tree_clf = DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=65,
                        max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0,
                        min_samples_leaf=1, min_samples_split=5,
                        min_weight_fraction_leaf=0.0,
                        random_state=None, splitter='best')
tree_clf.fit(X_train, y_train)
gini_importance = tree_clf.feature_importances_

# Compute information gain for each feature
info_gain = mutual_info_classif(X_train, y_train)

# Normalize the Gini index and information gain values
scaler = MinMaxScaler()
gini_importance_norm = scaler.fit_transform(gini_importance.reshape(-1, 1)).flatten()
info_gain_norm = scaler.fit_transform(info_gain.reshape(-1, 1)).flatten()

# Combine the normalized Gini index and information gain using weighted average
gini_weight = 0.7  # Weight for Gini index
info_gain_weight = 0.3  # Weight for information gain

combined_scores = gini_weight * gini_importance_norm + info_gain_weight * info_gain_norm

# Rank the features in descending order based on the combined scores
feature_scores = list(zip(range(len(combined_scores)), combined_scores))
feature_scores.sort(key=lambda x: x[1], reverse=True)

# Select top K features based on combined scores
K = 10  # Choose the desired number of features
selected_features = [feature for feature, _ in feature_scores[:K]]

# Filter the training and testing sets to include only the selected features
X_train_selected = X_train[:, selected_features]
X_test_selected = X_test[:, selected_features]

# Print the selected features
print("Selected Features:")
for feature, score in feature_scores[:K]:
    print("Feature", feature, "Score:", score)

# Print the shapes of the selected sets
print("\nShapes of the selected sets:")
print("X_train_selected shape:", X_train_selected.shape)
print("X_test_selected shape:", X_test_selected.shape)
```

Decision Tree Classification Report

```
[ ] print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
    print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 100.00%
_____________________________________________
CLASSIFICATION REPORT:
                    dos   normal        probe         r2l   u2r  accuracy  \
precision      1.000000      1.0     1.000000    0.997587   1.0  0.999983
recall         0.999977      1.0     0.999411    1.000000   1.0  0.999983
f1-score       0.999989      1.0     0.999705    0.998792   1.0  0.999983
support    43678.000000  70222.0  1697.000000  827.000000  44.0  0.999983

               macro avg   weighted avg
precision       0.999517       0.999983
recall          0.999878       0.999983
f1-score        0.999697       0.999983
support    116468.000000  116468.000000
_____________________________________________
Time taken: 0:02:04.552979
_____________________________________________
Test Result:
================================================
Accuracy Score: 99.98%
_____________________________________________
CLASSIFICATION REPORT:
                    dos   normal       probe         r2l       u2r  accuracy  \
precision      0.999908      1.0    0.995402    0.988304  0.857143  0.999794
recall         1.000000      1.0    0.997696    0.982558  0.750000  0.999794
f1-score       0.999954      1.0    0.996548    0.985423  0.800000  0.999794
support    10894.000000  17610.0  434.000000  172.000000  8.000000  0.999794

              macro avg  weighted avg
precision      0.968151      0.999789
recall         0.946051      0.999794
f1-score       0.956385      0.999790
support    29118.000000  29118.000000
_____________________________________________
Time taken: 0:02:06.190113
_____________________________________________
```

Support Vector Machine

```
from sklearn.svm import SVC


svm_clf = SVC(kernel='rbf', gamma=0.1, C=1.0)
svm_clf.fit(X_train, y_train)

print_score(svm_clf, X_train, y_train, X_test, y_test, train=True)
print_score(svm_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 100.00%
_____________________________________________
CLASSIFICATION REPORT:
                dos   normal   probe    r2l   u2r  accuracy  macro avg  \
precision       1.0      1.0     1.0    1.0   1.0       1.0        1.0
recall          1.0      1.0     1.0    1.0   1.0       1.0        1.0
f1-score        1.0      1.0     1.0    1.0   1.0       1.0        1.0
support     43678.0  70222.0  1697.0  827.0  44.0       1.0   116468.0

            weighted avg
precision            1.0
recall               1.0
f1-score             1.0
support         116468.0
_____________________________________________
Time taken: 0:02:36.874662
_____________________________________________
Test Result:
================================================
Accuracy Score: 100.00%
_____________________________________________
CLASSIFICATION REPORT:
                dos   normal  probe    r2l  u2r  accuracy  macro avg  \
precision       1.0      1.0    1.0    1.0  1.0       1.0        1.0
recall          1.0      1.0    1.0    1.0  1.0       1.0        1.0
f1-score        1.0      1.0    1.0    1.0  1.0       1.0        1.0
support     10894.0  17610.0  434.0  172.0  8.0       1.0    29118.0

            weighted avg
precision            1.0
recall               1.0
f1-score             1.0
support          29118.0
_____________________________________________
Time taken: 0:02:40.778982
_____________________________________________
```

K-Nearest Neighbours

```
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_train)

print_score(knn_clf, X_train, y_train, X_test, y_test, train=True)
print_score(knn_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 99.98%
_____________________________________________
CLASSIFICATION REPORT:
                    dos        normal        probe         r2l        u2r  \
precision      0.999931      0.999772     1.000000    0.997561   0.951220
recall         1.000000      0.999986     0.995286    0.989117   0.886364
f1-score       0.999966      0.999879     0.997637    0.993321   0.917647
support    43678.000000  70222.000000  1697.000000  827.000000  44.000000

              accuracy      macro avg   weighted avg
precision     0.999803       0.989697       0.999801
recall        0.999803       0.974150       0.999803
f1-score      0.999803       0.981690       0.999801
support       0.999803  116468.000000  116468.000000
_____________________________________________
Time taken: 0:06:11.878235
_____________________________________________
Test Result:
================================================
Accuracy Score: 99.99%
_____________________________________________
CLASSIFICATION REPORT:
                    dos        normal       probe         r2l  u2r  accuracy  \
precision      0.999908      0.999830    1.000000    1.000000  1.0  0.999863
recall         1.000000      1.000000    0.993088    0.994186  1.0  0.999863
f1-score       0.999954      0.999915    0.996532    0.997085  1.0  0.999863
support    10894.000000  17610.000000  434.000000  172.000000  8.0  0.999863

               macro avg  weighted avg
precision       0.999948      0.999863
recall          0.997455      0.999863
f1-score        0.998697      0.999862
support     29118.000000  29118.000000
_____________________________________________
Time taken: 0:06:59.907852
_____________________________________________
```

Logistic regression

```
from sklearn.linear_model import LogisticRegression

logreg_clf= LogisticRegression(solver='lbfgs', multi_class='auto')
logreg_clf.fit(X_train, y_train)

print_score(logreg_clf, X_train, y_train, X_test, y_test, train=True)
print_score(logreg_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 100.00%
_____________________________________________
CLASSIFICATION REPORT:
               dos   normal   probe    r2l   u2r  accuracy  macro avg  \
precision      1.0      1.0     1.0    1.0   1.0       1.0        1.0
recall         1.0      1.0     1.0    1.0   1.0       1.0        1.0
f1-score       1.0      1.0     1.0    1.0   1.0       1.0        1.0
support    43678.0  70222.0  1697.0  827.0  44.0       1.0   116468.0

           weighted avg
precision           1.0
recall              1.0
f1-score            1.0
support        116468.0
_____________________________________________
Time taken: 0:11:53.706078
_____________________________________________
Test Result:
================================================
Accuracy Score: 100.00%
_____________________________________________
CLASSIFICATION REPORT:
               dos   normal  probe    r2l  u2r  accuracy  macro avg  \
precision      1.0      1.0    1.0    1.0  1.0       1.0        1.0
recall         1.0      1.0    1.0    1.0  1.0       1.0        1.0
f1-score       1.0      1.0    1.0    1.0  1.0       1.0        1.0
support    10894.0  17610.0  434.0  172.0  8.0       1.0    29118.0

           weighted avg
precision           1.0
recall              1.0
f1-score            1.0
support         29118.0
_____________________________________________
Time taken: 0:11:54.701982
_____________________________________________
```