

RESEARCH ON NETWORK INTRUSION DETECTION BASED ON MACHINE LEARNING WITH HYPERPARAMETER OPTIMISATION

By
Yifei Sun

Supervisor: Nitheesh Kaliyamurthy

Project submitted as part of the requirements
for the award of BSc (Hons) Computer
Networks

September 2023

Acknowledgements

Declaration of Originality

I, Yifei Sun declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed :

Date :

FORM OF CONSENT

I Yifei Sun hereby consent that my Project, submitted in candidature for the B.Sc.(Hons) Computer Networks degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed :

Date :

Copyright Acknowledgement

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to University of Wales Trinity Saint David, Swansea.

Abstract

In recent years, rapid advancements in new networking technologies have given rise to significant concerns regarding network security as these technologies permeate various sectors. Effectively safeguarding network information and countering malicious network attacks in real-time have become paramount topics of discussion. The development and implementation of an efficient and user-friendly intrusion detection system (IDS) play a pivotal role in filtering out malicious traffic requests and mitigating network security vulnerabilities.

Against this backdrop, this survey outlines its research objectives and experimental outcomes as follows. Firstly, it systematically reviews the application of machine learning in intrusion detection. Subsequently, it designs and implements a comprehensive intrusion detection model, optimizing parameters for six machine learning models: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, Logistic Regression, Neural Network, and Random Forest. These models are subjected to binary classification tasks using the NSL-KDD dataset. Optimal parameters are determined for each model, excluding SVM. Finally, the performance of the five parameter-optimized models is evaluated using standard machine learning metrics, including Accuracy, Precision, Recall, F1 Score, and ROC Curve. Among these models, the Neural Network model achieves the highest accuracy (80.6%) and the maximum AUC value of 0.96, while the Logistic Regression model performs the poorest, with an accuracy of only 75.4%, accompanied by lower values for the other evaluation metrics. Notably, all five machine learning models excel in recognizing positive classes but exhibit slightly diminished performance in the negative class. These findings underscore the significance of machine learning models, especially the Neural Network model, in enhancing intrusion detection and fortifying network security against evolving and sophisticated threats..

Table of Content

Acknowledgements	1
Abstract	2
Table of Content	3
List of Figures	5
List of Tables	6
CHAPTER1 - INTRODUCTION	7
1.1. Research Background	7
1.2. Research Aim	10
1.3. Structure Of The Project	11
CHAPTER 2-RESEARCH & REVIEW OF LITERATURE	13
2.1. Introduction	13
2.2. Theoretical foundation, related technology research	13
2.2.1. Network Intrusion Detection	14
2.2.2. Machine Learning Models	15
2.2.3. Study of the dataset	22
2.2.4. Optimised design of parameters	24
2.3. Research gaps	25
2.4. Summary	25
CHAPTER 3 - RESEARCH METHODOLOGY	27
3.1. Research Methodology	27
3.1.1. Research Philosophy	27
3.1.2. Research Strategy	28
3.1.3. Approaches	29
3.1.4. Risks & Limitations	30
3.1.6. Data Evaluation Methods	31
3.2. Development Methodology	32
3.3. Resources	33
3.3.1. Hardware Resources	33

3.3.2. Software Resources	33
3.4. Summary	33
CHAPTER 4 - DESIGN & IMPLEMENTATION	34
4.1. Introduction	34
4.2. Problem Formulation	34
4.3. Design	34
4.4. Implementation	36
4.4.1. Dataset pre-processing	36
4.4.2. Machine Learning Modelling	41
4.4.3. Optimal design of machine learning model parameters	44
4.4.4. Results output	45
4.5. Summary	46
CHAPTER 5 - TESTING AND EVALUATION	47
5.1. Testing	47
5.1.1. Dataset pre-processing	47
5.1.2. Machine Learning Model Parameter Optimisation	50
5.2. Evaluation	53
5.3. Summary	55
CHAPTER 6 - CONCLUSIONS & RECOMMENDATIONS	57
6.1. Experimental results	57
6.1.1. Optimal parameter design results	57
6.1.2. Results of machine learning model runs	57
6.1.3. Comparison of machine learning models	62
6.2. Full summary	65
6.3. Recommendations for Further Research	66
CHAPTER 7 – REFLECTIONS	68
CHAPTER 8 - REFERENCES	71

List of Figures

Figure. 1	Global consumer IP traffic data per month from 2017 to 2022	7
Figure. 2	Intrusion detection system classification	9
Figure. 3	Schematic of SVM under binary classification	16
Figure. 4	Schematic diagram of a simple KNN model	17
Figure. 5	Schematic diagram of a simple decision tree model	18
Figure. 6	Logistic regression model	20
Figure. 7	Simple artificial neural network models	21
Figure. 8	Research methodology overview	27
Figure. 9	Experimental process	36
Figure. 10	View Dataset	39
Figure. 11	Five random rows in the data set	48
Figure. 12	Attack types and normal types	49
Figure. 13	Encoded dataset	49
Figure. 14	KNN model binary classification confusion matrix	58
Figure. 15	KNN model ROC-AUC curve	58
Figure. 16	DT Model Binary Classification Confusion Matrix	59
Figure. 17	DT model ROC-AUC curve	59
Figure. 18	LR model binary classification confusion matrix	60
Figure. 19	LR model ROC-AUC curve	60
Figure. 20	NN model binary classification confusion matrix	61
Figure. 21	NN model ROC-AUC curve	61
Figure. 22	RF model binary classification confusion matrix	62
Figure. 23	RF model ROC-AUC curve	62
Figure. 24	Accuracy comparison of 5 models	63
Figure. 25	Precision comparison of 5 models	64
Figure. 26	Recall comparison of 5 models	64
Figure. 27	F1-Score Comparison of 5 Models	65

List of Tables

Table. 1	Comparison of datasets	23
Table. 2	Hardware & Software	34
Table. 3	43 characterisations in NSL-KDD	37
Table. 4	Attack types	39
Table. 5	Attack types statistics	40
Table. 6	Confusion matrix	54
Table. 7	Results of optimal design of model parameters	57

CHAPTER1 - INTRODUCTION

1.1. Research Background

The Internet and information technology industry are currently experiencing rapid and robust growth. Computer network technology and its related applications play a crucial role in various domains such as politics, economics, culture, and military affairs. In recent years, the development of new networking technologies like IPv6, cloud computing, the Internet of Things (IoT), and mobile connectivity has matured and become increasingly sophisticated. These advancements have significantly enhanced the convenience of daily life through Internet services[1]. Currently, global Internet traffic is notably characterised by its large volume, high dimensionality and variety, with Figure 1 showing monthly global consumer IP traffic data between 2017 and 2022 [2], A rapid upward trend can be seen in the amount of global consumer IP traffic data each year.

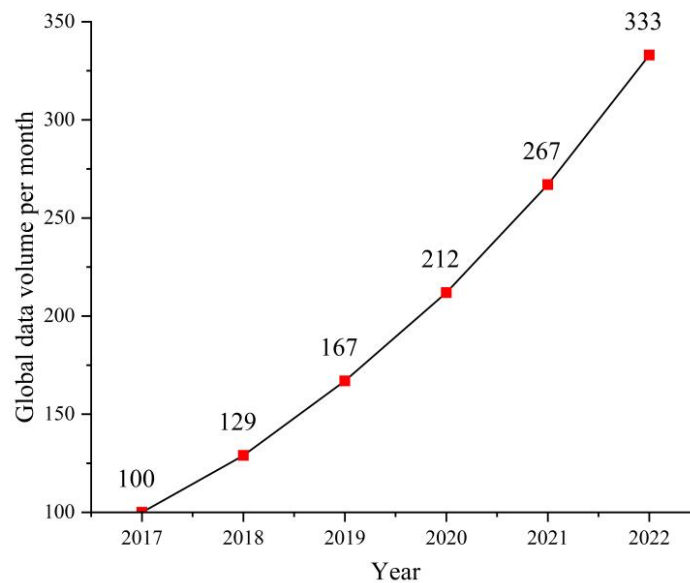


Figure. 1 Global consumer IP traffic data per month from 2017 to 2022

As the volume of diverse Internet traffic continues to grow exponentially, network security issues have become increasingly prevalent. Incidents of user data breaches are on the rise, and the types of network attacks have become more diversified. In recent years, there has been a surge in network attack events, affecting enterprises, governments, and various organizations in different countries on a massive scale. The

resulting losses are difficult to predict, posing not only challenges to individual security but also significant economic repercussions [3]. In 2017 the Equifax data breach quickly festered in the United States, resulting in a cyberattack that left nearly half of the American population exposed to the risk of data theft, causing significant financial loss and personal safety issues [4]. In 2016, Yahoo! reported that at least 500 million accounts had had their personal information stolen in an apparent "state-sponsored" data breach the year before, causing hundreds of millions of dollars in losses [5]. In 2016, hackers used the Mirai botnet to launch a DDoS attack against the US DNS provider Dyn, which resulted in the inaccessibility of hundreds of websites such as Twitter, Spotify, NetniX and others [6]. In November 2019, one hundred and ten nursing home information systems in the United States were also compromised by Ryuk ransomware, from which the hackers profited substantially by stealing personal information [7]. At the same time, the booming development of the Internet of Things (IoT) has given rise to a large number of industrial control devices and systems, but due to the lack of control, IP exposure to external networks, vulnerabilities are not dealt with in a timely manner, etc., it is easy to be exploited by hackers, resulting in extremely serious consequences [8]. The frequent occurrence of various types of data leakage incidents in recent years shows that the total number of data leakage incidents has been continuously increasing, and it is very important for how to protect the government, enterprises, and individuals from cyber-attacks and invasions.

Intrusion detection is a common method of network security protection. It involves monitoring the behavior within networks and on hosts, determining whether it complies with predefined policies to achieve real-time system protection. These behaviors encompass violations of computer security policies and acceptable usage standards. Intrusion detection systems are proactive defense security devices used for implementing intrusion monitoring, as illustrated in Figure 2. They are primarily categorized into Host-based Intrusion Detection Systems (HIDS), Network-based Intrusion Detection Systems (NIDS), and Hybrid Intrusion Detection Systems (HYIDS) [9].

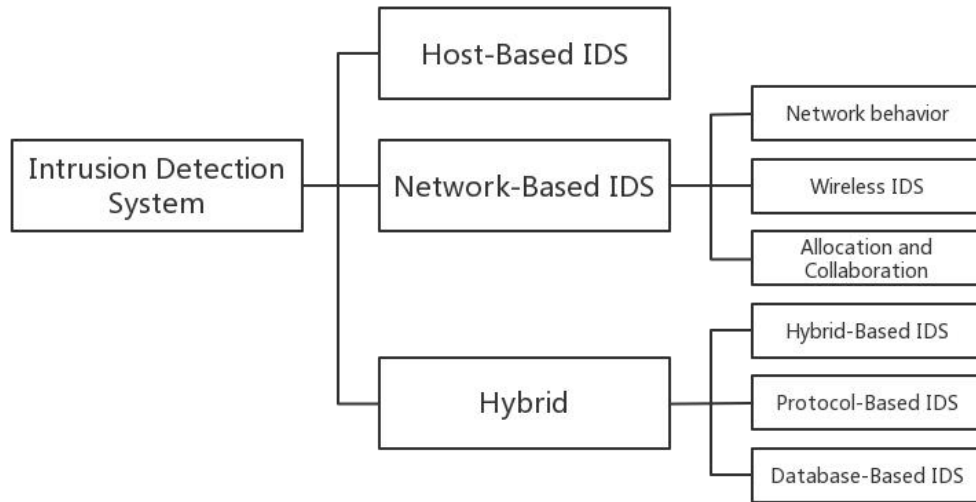


Figure. 2 Intrusion detection system classification

Among them, HIDS focuses on various security threats reaching the host and does not focus on the security of network traffic, and the monitoring and defence of network traffic is the function of NIDS, so this paper focuses on the research of intrusion detection algorithms under the NIDS. Traditional security solutions such as firewall technology [10], virus prevention and control technology [11], SSL encryption technology [12] and other technologies covering the server, network and cloud storage, although played a certain role in protection, but with the increasing complexity of the network system, has been unable to withstand unknown attacks and external interference, so the traditional security solutions are no longer applicable. Therefore, how to do early detection and discovery of the threat time, real-time judgement protection network attacks, timely emergency disposal of data leakage is the current trend of Internet security. Intrusion detection first of all to extract the important features of the computer system and network, and then these features and normal features and known intrusion features to be compared and analysed, in the invasion of the computer system and the network caused by the negative impact of the behaviour of the computer system and the network before the early discovery of it, so as to find out the invasion of the computer system and the network suffered by the invasive behavior and to take the appropriate security measures to eliminate the invasion of the threat. Therefore, intrusion detection is a key research content and an

important part in the field of computer and network security. In practice, by building an intrusion detection system [13] to complete the real-time monitoring of network transmissions, if found suspicious transmissions will send security alerts, according to the category of intrusion to start the corresponding security module to eliminate the threat of intrusion. Depending on the technology used, intrusion detection systems can be generally classified into misuse detection systems and anomaly detection systems [14]. Misuse detection is a static detection technique based on features, which is characterised by the need to build a database of intrusion features in advance based on past intrusion experience, and the researcher is only able to detect existing types of intrusions based on this database, and is unable to detect new types of intrusions until the intrusion features in the database are updated. Anomaly detection is a dynamic detection technique based on behaviour, which is characterised by monitoring and analysing the intruder's complete activity in a certain network time, while comparing it with the behaviour of the current system and network, and if it is found to be in violation of the behavioural norms of the current system and network, the activity is deemed to be an illegal intrusion [15].

Specifically, the task of intrusion detection is to distinguish intrusions from normal behaviours and to determine the specific category of intrusion based on the characteristics, so intrusion detection belongs to a standard classification problem, and machine learning, as the core of artificial intelligence, has been shown to be able to efficiently perform a variety of different classification tasks [16]. Machine learning is more able to cope with today's constantly updated multiple intrusion behaviours by virtue of its autonomous learning intelligence. Therefore, both feature-based misuse detection and behaviour-based anomaly detection can adopt machine learning as an efficient intrusion detection technique, which makes how to adopt machine learning algorithms to improve the effectiveness of intrusion detection become a hot research topic in today's intrusion detection field, which is also the focus of this study.

1.2. Research Aim

The research objectives of this project aim to design a comprehensive intrusion detection model and test machine learning-based intrusion detection models using a selected intrusion detection dataset. The purposes and objectives of this research can be summarized as follows::

- Conduct a comprehensive review of intrusion detection technology, emphasizing its contemporary significance. This review will encompass the historical evolution of intrusion detection, prevalent algorithms, and the application of diverse machine learning models in this domain, while also evaluating the pros and cons of these models.
- Perform a comparative analysis of prominent intrusion detection datasets. Investigate current methods for preprocessing datasets and identify parameters within machine learning models that can be optimized.
- Establish a complete workflow and plan for the project to achieve its intended research objectives. Design a comprehensive research methodology to guide the research process effectively.
- Create a holistic intrusion detection model and design an experimental workflow that will be meticulously executed step by step.
- Develop machine learning models, including Support Vector Machine, K-Nearest Neighbors, Decision Trees, Logistic Regression, Neural Networks, and Random Forest. Focus on parameter optimization, utilizing techniques such as grid search and cross-validation to determine the optimal parameters for these models.
- Conduct binary classification tasks using the NSL-KDD dataset to investigate performance variations among different machine learning models.◦

1.3. Structure Of The Project

In this project, the structure is organized as follows:

- Chapter 1 serves as the introduction to the entire paper. It provides an overview of recent developments in network technologies, the research background related to cybersecurity, and the rationale behind conducting research in the field of intrusion detection. Additionally, it briefly outlines the research objectives.

- Chapter 2 offers a literature review, summarizing relevant past articles and books. It defines the primary research directions and identifies current research gaps.
- Chapter 3 outlines the research methodology, elucidating the research philosophy, strategies, implementation methods, principles, and data collection approaches employed in the project.
- Chapter 4 is the design and implementation section. Building on the literature review and research methodology, it details the comprehensive experimental plan and follows through with the experimental process.
- Chapter 5 covers testing and evaluation. It conducts tests based on the previous implementation to obtain the project's final results and assess these outcomes.
- Chapter 6 provides the conclusion and outlook. It summarizes the research content, critically analyzes patterns and final results obtained in Chapter 5.
- Chapter 7 is the reflection section, offering a detailed retrospective on the research process across all chapters. In this section, the project's progress is thoroughly described, problems encountered and resolved are discussed, and shortcomings in the research are acknowledged.

CHAPTER 2-RESEARCH & REVIEW OF LITERATURE

2.1. Introduction

To ensure the accuracy and validity of the research, various factors need to be considered, and one of these factors is literature retrieval. The source of literature can significantly impact the final results of this work. This project utilized four primary sources when conducting research and literature review: IEEE, ACM Digital Library, Elsevier, and SpringerLink. Specifically, SpringerLink and ScienceDirect were used to search for review literature, including the current state of research in network security intrusion detection, the application of machine learning models in intrusion detection, and comparisons of intrusion detection datasets. In these review articles, a wealth of summarized literature content is available, and selective citation of this content can be used for analysis and reference in the experiments. IEEE Xplore and ACM Digital Library, along with other relevant conferences and journals, were used to critically analyze the methods used in previous research, summarize the work of others, and identify areas of improvement. When selecting content for citation, the following strategies will be employed:

- Ensure the authenticity of the article's content. Articles that do not mention data sources, do not present research results, have no abstract or rely solely on an abstract, or are not written in English will be excluded from the citation list.
- When conducting literature searches with the same keywords, articles from journals with higher impact factors will be given priority if their research content is highly similar.
- In cases where literature searches yield content with similar research topics, the most recent research that covers and summarizes older versions of the research will be selected for citation to avoid redundancy.
- Search results must be related to intrusion detection research, and the publication year of relevant research literature (except for citations of concepts or formulas) should be within the last few years and published before September 2023.

2.2. Theoretical foundation, related technology research

2.2.1. Network Intrusion Detection

The concept of network intrusion detection was first proposed by Anderson et al. He defined intrusion detection as a means of network security protection that monitors the operational status of a system in real time and detects traffic that has an attack attempt, traffic that is in the middle of an attack, or traffic that has already completed an attack, and the goal of intrusion detection is to safeguard the privacy, integrity, and reliability of system resources [17]. In the following 1987, Denning et al [18] designed an intrusion detection model based on real-time network monitoring records to support only special experimental environments, after which Lunt [19] proposed IDES (Intrusion Detection Expert System), which can be carried out without the need of specific experimental environments, and which is able to independently spontaneously monitor target hosts and can meet different intrusion detection needs. In the early 90's, Heberlein et al [20] proposed NSM (Network Security Monitor) system can monitor the computer network data traffic, capture the packet analysis to determine whether there is an attack, in the dynamic changes in the network environment to achieve the desired detection rate, creating a new world of intrusion detection using network traffic.

Research in the field of intrusion detection focuses on exploring how to efficiently identify normal traffic and attack traffic from massive raw network data, and develop reliable defence rules against malicious attacks. With the rapid development of the field of artificial intelligence, the application of machine learning algorithms to the field of intrusion detection has become one of the research directions in the field of network information security. Intrusion detection models based on machine learning algorithms can actively, stably, and efficiently learn complex pattern features to provide valuable information for decision analysis [21], [22]. Wenke Lee [23] combined data mining techniques such as association rules and statistical sequences to identify anomalous attacks, and this article inspired future research on the use of machine learning for intrusion detection. Currently, intrusion detection methods based on traditional machine learning have been widely used in the field of intrusion

detection, such as support vector machines, K-nearest neighbours, decision trees, logistic regression, neural networks, etc. [24]. The subsequent subsection part will make a comprehensive study of these machine learning models and point out how various machine learning models are currently useful in the field of intrusion detection and critically point out the shortcomings of the study.

2.2.2. Machine Learning Models

2.2.2.1. Support Vector Machine

SVM (Support Vector Machine) was first proposed by Vapnik in 1998 [25] and it is a supervised model for classification, regression and outlier detection. It is the most commonly used and accurate model among most of the current machine learning models. Figure 3 shows a schematic of a simple binary classification SVM, the main idea of SVM is to find a hyperplane in the high-dimensional space as a partition of the two classes for a two-class classification problem to ensure the minimum classification error rate [26]. In the field of intrusion detection, many scholars have used the SVM model as a means of detecting intrusions and made improvements on the traditional SVM algorithms, for example, Pervez et al [27] proposed a filtering algorithm based on SVM and selected NSL-KDD as the intrusion dataset for the classification task, and he experimented with different input features by selecting different input features, all of them reached 99% accuracy rate. However, it performs poorly in the test set with an F1 score of only 0.77, and the generalisation ability of the model is not able to meet the effectiveness in detecting unknown network intrusions. Gu et al [28] proposed an effective intrusion detection framework based on SVM with plain Bayesian feature embedding and validated the model on multiple datasets in intrusion detection, where it obtained a 99.35% accuracy. This indicates that SVM is better at handling classification tasks, especially for datasets that are better pre-processed. More, SVM can be mixed with various other methods to get excellent results. Al-Qatf et al [29] proposed a self-learning model based on the STL framework, which not only improves the effectiveness of SVM for attack detection, but also solves the high time-consumption problem of the SVM model when dealing

with datasets with large data capacity. Tao et al [30] combined the genetic algorithm and support vector machine algorithms to propose a new GA algorithm, which can be combined with feature selection in genetic algorithms to make a reduction in the error rate of SVM, and obtained a true positive rate of 94.53%.

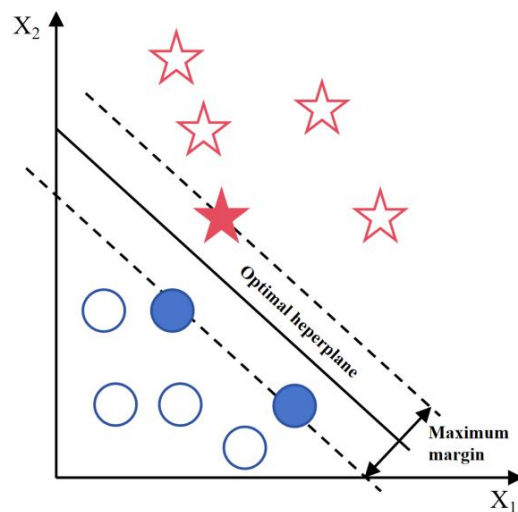


Figure. 3 Schematic of SVM under binary classification

The advantages and disadvantages of the SVM machine learning model after the study of its advantages and disadvantages can be summarised as follows:

- SVM models are mainly used for binary classification problems and are extremely effective when dealing with non-linear data, but are not very advantageous when dealing with multi-classification problems.
- The training of SVM models is time-consuming, especially with large data volumes and high dimensionality, a problem that was verified in the research of this project.
- High accuracy in classification and prediction tasks.
- High robustness to outliers.

2.2.2.2. K-Nearestneighbor

KNN (K-Nearestneighbor) is commonly used in classification and regression problems, and its schematic is shown in Fig. 4. It is based on the principle of calculating the Euclidean distances between different points on the input vectors, and then assigning the unlabelled points to their K-nearest-neighbours' classes [31]. In fact, the KNN algorithm is lazy when KNN solves classification problems, so it has

extremely high time complexity and has obvious disadvantages when dealing with larger datasets [32]. Many scholars are cautious in using KNN for intrusion detection and usually adopt hybrid as well as reduced dataset approaches. For example, Saleh et al [33] first reduced the dimensionality of the sample data through a plain base feature selection technique. This is then used to train optimised SVMs for outlier rejection. Finally, after the outliers are rejected, HIDS can successfully detect the attack by should KNN classifier. And it performs a high detection rate for rare attacks such as R2L and U2R, which is a result not obtained by many current methods because rare attacks such as R2L and U2R account for a very small number of NSL-KDD, which will be explained in more detail later. Shapoorifard et al [34] improved the existing intrusion detection task by combining K-MEANS clustering and KNN classification in KNN classifiers, improving the comprehensive performance of KNN in intrusion detection. However, the article does not give detailed experimental results.

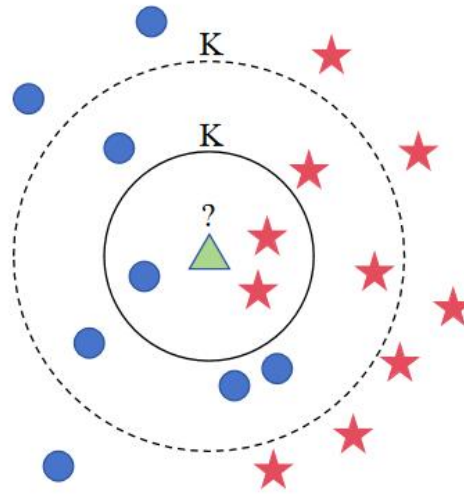


Figure. 4 Schematic diagram of a simple KNN model

The advantages and disadvantages of the KNN machine learning model after the study of its advantages and disadvantages can be summarised as follows:

- KNN model is one of the more popular algorithms in data mining and has been widely used in data analysis applications [35].
- High time complexity when dealing with large data sets with high dimensionality.

2.2.2.3. Decision tree

Decision tree is a tree structure, its brief model diagram is shown in Fig. 5, which classifies the samples through a series of decisions represented in the tree structure, where the current decision helps to make subsequent decisions, such that the sequence of decisions is represented as a tree structure [36]. Decision trees classify samples by training conditions and have better detection accuracy for known conditions, but are not applicable for detection of unknown conditions [37]. Decision tree models are also more common in the field of intrusion detection, Rai et al [38] first segmented the data and after that used the C4.5 model which is common in decision trees for intrusion detection on the NSL-KDD dataset, the shortcoming is that the project has only 52% accuracy. Alkasassbeh et al [39] chose the J48, MLP, and Bayesian network classifiers. The results proved that J48 classifier has the highest accuracy in detecting and classifying all KDD dataset attacks (containing all four types). Liu et al [40] experimented with a decision tree combined with ANFIS approach and the method proposed in this study achieved a detection rate of 77.46% on the KDDTest+ dataset. Azad et al [41] proposed a Genetic Algorithm based intrusion detection system to solve the problem of analysing smallness in decision trees. The designed system obtained the best results by comparing multiple methods in decision trees after testing on the KDD CUP dataset. The shortcoming of this study is that a more outdated KDD CUP dataset was used. It is possible that the results obtained are not very accurate.

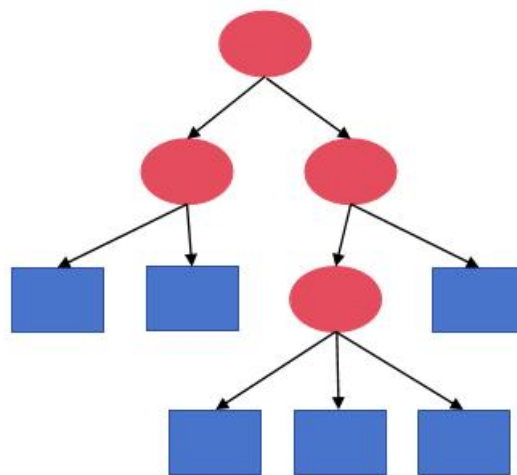


Figure. 5 Schematic diagram of a simple decision tree model

For DT machine learning models after the study its advantages and disadvantages can be summarised as follows:

- High interpretability, easy to understand and interpret through a voting-like decision making.
- Decision trees can be visualised as a tree structure, which helps one to better understand the decision-making process of the model and the importance of the features, and in Weka [42], the overall model of the decision tree can be seen through visualisation.
- Influenced by the data as well as feature selection, unclear features can affect the results and noisy or incorrect data can lead to inaccurate models.

2.2.2.4. Logistic regression

LR (logistic regression) is a statistical learning method for solving classification problems, which is suitable for mapping input data to a discrete output category, usually for binary classification problems, but can also be extended to multiclassification problems [43]. The mathematical expression for logistic regression as well as a simple model is shown in Fig. 6. One of the popular machine learning binary classification methods has long been considered unbiased, especially for high-dimensional data [44]. However, logistic regression does not seem to be a particularly good choice for the field of intrusion detection, in a study by Kaddoura et al [45], he compared several machine learning models such as artificial neural networks, logistic regression, support vector machines and trained them on an intrusion detection dataset, and the result was that the logistic regression algorithm accuracy was only for was 71.4%. Wang et al [46] designed a polynomial logistic regression model and tested it on the KDD-Cup 99 dataset and obtained an excellent result with an AUC of 0.99.

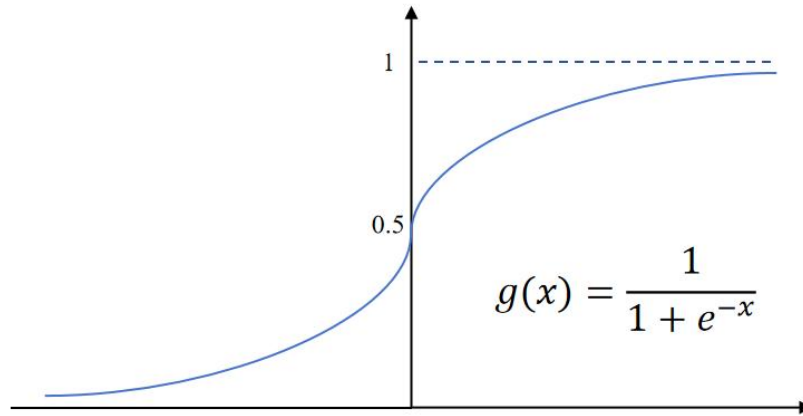


Figure. 6 Logistic regression model

For LR machine learning models after the study its advantages and disadvantages can be summarised as follows:

- Good interpretability, lower computer requirements for computation, and better performance.
- Higher tolerance for feature correlation, even if there are multiple correlated features, it can make better predictions.
- It is only applicable to linear decision boundaries and cannot handle complex nonlinear relationships well.

2.2.2.5. Neural network

Neural network is a machine learning technique that mimics the neural network of the human brain with a view to enable artificial intelligence-like learning, which learns from examples and generalises from noisy and incomplete data in order to perform tasks [47]. Figure. 7 shows a simple neural network model containing three layers, which contains an input layer, a hidden layer and an output layer [48]. For the field of intrusion detection, neural networks are widely used due to their characteristics. Wang et al [49] proposed a new approach based on ANN and fuzzy clustering, followed by training different ANN models to formulate different base models based on different training subsets. Finally, it was concluded that the method outperforms other conventional machine learning models. Ibrahim et al [50] designed a Self-Organising Map (SOM) Artificial Neural Network, which had a detection rate of 92.37% on the KDD -99 dataset and 75.49% on the NSL-KDD

dataset. Govindarajan et al [51] in the data mining domain tried to use multilayer perceptron and concluded that the ensemble of multilayer perceptron and radial basis function significantly improved the prediction accuracy of intrusion detection.

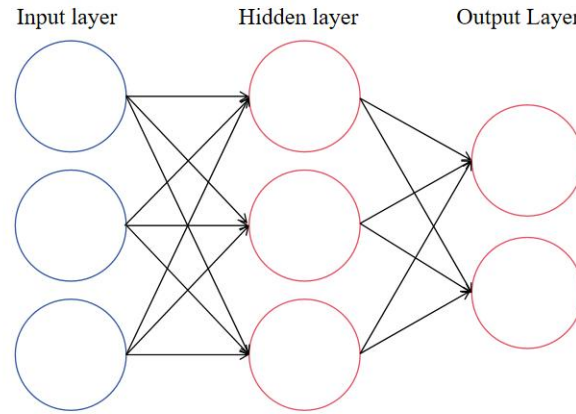


Figure. 7 Simple artificial neural network models

The advantages and disadvantages of neural networks machine learning models after research can be summarised as follows:

- Neural networks usually perform well on large-scale datasets because they have sufficient parameters and flexibility to capture subtle patterns in the data.
- For small-scale datasets, neural networks may perform poorly because they require a large amount of data to train effective parameters.
- Tuning the architecture and hyperparameters of a neural network usually requires a lot of trial and experience to find the optimal settings.

2.2.2.6. Random forest

Random forest model is an integrated learning algorithm which is a superior choice for the classification problem in this study. It is an integrated approach based on decision trees which improves the performance and robustness of the model by combining multiple decision trees. It has been frequently used in the field of intrusion detection, especially for the classification task of NSL-KDD dataset [52]. P Negandhi et al [53] used Random Forest to train the model to detect various cyber-attacks and took feature engineering to process the dataset and obtained an accuracy of 99.752, however, the experiment lacked all the remaining metrics and there was an omission of the key experimental parts. R. K. Vigneswaran et al [54] used LGBM and Random

Forest model on three datasets and obtained high accuracy rates , his study proves that integrated learning will be more promising in future research.

2.2.3. Study of the dataset

2.2.3.1. Intrusion detection datasets

The use of datasets for the validation of machine learning based intrusion detection systems is a prerequisite for the completion of the project. A detailed explanation of the dataset selected for this project and the manipulation of the dataset will be presented in Chapter 3 as well as Chapter 4. Currently, there are several conventional intrusion detection datasets:

- KDD99 dataset, this dataset is used by most of the literature, it is a 9-week network connection and system audit data collected from a simulated US Air Force LAN, divided into training data with labels and unlabelled test data, which have different probability distributions, and the test data contains certain types of attacks that do not exist in the training data centre, which can make the intrusion detection plus realism, each connection record in the KDD99 training dataset contains 41 fixed feature attributes and an identifier type that is used to determine whether the record is a normal or offensive behaviour [55], [56].
- The NSL-KDD dataset [57] is based on the KDD99 dataset and then improved, the improved dataset no longer contains redundant data, so the classifier will no longer be biased towards frequent records, so that the classification rate of different machine learning methods can be varied over a wider range, making the evaluation more accurate. This dataset is chosen as the intrusion detection dataset for this project, and the specific description will be reflected later.
- The raw network packets for the UNSW-NB15 [58] dataset were created by the Cyber Security Laboratory at the Australian Centre for Cyber Security (ACCS) using the IXIA PerfectStorm tool for mixing realistic modern normal behaviour with contemporary attacks.
- The CICIDS 2017 dataset [59] is designed to react to the latest attacks that are similar to real-world data. The dataset reacts to the abstract behaviour of 25 users

based on protocols such as HTTP, HTTPS, FTP, SSH and email. The dataset leverages CICFlowMeter to analyse the data traffic using labelled flows based on timestamps, source and destination ports, source and destination ip, protocols and attacks.

Each of these datasets has its advantages as well as disadvantages as shown in the table below:

Table 1 Comparison of datasets

Dataset	Advantages	Disadvantages
KDD-99	● Highest number of users	● Data types are unbalanced
	● Each data comes with detailed characterisation	● Outdated
	● Multiple types of attacks	● Not applicable to intrusion detection systems
	● Better than KDD99 dataset	● Lacks modern low-script attack scenarios
NSL-KDD	● Reduces duplicates in data	● Not applicable to intrusion detection systems
	● Overcomes limitations of KDD99	
	● Mixes real normal activities and modern attack behaviours	● More complex due to similarities between normal network traffic and modern attack behaviour
UNSW-NB15	● Provides network traffic data files	
	● Network flows with labelling	● Not publicly available
CICIDS 2017	● Provides network traffic data files	● Not applicable to intrusion detection systems

2.2.3.2. Data set pre-processing methods

There are various preprocessing methods for datasets, which is one of the most important aspects of machine learning projects, and different means are needed for different datasets [60], and the specific types of methods are as follows:

- Feature selection: select the most informative features and remove redundant or irrelevant features. This helps to reduce the dimensionality and improve the

performance and training speed of the model, which is especially effective for high-dimensional datasets commonly used for intrusion detection [61].

- Data conversion: in most cases, datasets are not always saved in one format and need to be converted, e.g., English characters appearing in KDD-99 need to be converted to numbers. There are also some other methods applied in intrusion detection data conversion processing to convert the classification features into a digital form that can be understood by the model, such as One-Hot coding, label coding, etc. [62].
- Feature processing: includes a variety of methods in which features can be scaled so that they have similar scales. Common scaling methods include normalisation (so that features have mean 0 and variance 1) and normalisation (scaling features to a certain range).
- Data balancing process: if the number of samples from different categories in a dataset is unbalanced, the data can be balanced using methods such as undersampling, oversampling, and generating synthetic samples [63], [64]. For the KDD-99 dataset, there is a huge difference between the eigenvalues, and the use of normalisation is taken into account in the follow-up.

2.2.4. Optimised design of parameters

In machine learning, for model design, different parameter design will directly lead to different results, for different datasets, how to design appropriate parameters to get better results and balance time, performance and other factors is the key in model design [65]. The following are references for the design of each hyperparameter in the machine learning model used in this project:

- In SVM models, the choice of kernel function is extremely important, in general, the RBF kernel function performs well in most cases, but different problems may require tuning of the parameters C and gamma [66].
- K is an important parameter in KNN and different values of k lead to different performance. If k is quite large, the neighbours used for prediction will incur a larger classification time and affect the accuracy of the prediction. Generally, the

most suitable value of k can be found by the automatic algorithm function in sklearn [67].

- The number of decision trees combined is the most important parameter in DT models, and it is the basic hyperparameter that controls the complexity of the DT algorithm since deeper trees have more subtrees to make more accurate decisions [68].
- Logistic regression can be solved by using different optimisation algorithms to solve the weights of the model i.e. LRSolver (Optimisation Algorithm), the common ones are "liblinear", "lbfgs", "newton-cg", "sag", etc. Different algorithms may have different adaptability to the size and nature of the dataset.
- In the NN model, the number of iterations, the number of layers and neurons of the neural network, the activation function selection, the Dropout rate, etc. can have a great impact on the model, which needs to be adjusted several times so as to determine the appropriate parameters [69]. Not only that, the number of iterations also affects the final output of the NN model.
- In the random forest model, the depth of the tree, the number of trees, the size of the minimum number of samples, the minimum number of split samples, the number of cross-validations, and so on, all affect the final results, considering that the selected dataset has a high dimensionality and a large scale [70].

2.3. Research gaps

At present, combining machine learning with intrusion detection has become a more popular research direction, through reading a large number of literature found that at this stage of machine learning-based intrusion detection research, most scholars in the various types of machine learning models to do comparisons, the lack of various types of models itself parameters of the optimal design. At the same time, the design ideas of many articles are very cumbersome, and it is difficult to achieve repeated experiments under the conditions of multidisciplinary popularity and use, and it is also difficult to be applied to various fields.

2.4. Summary

In this section, the implementability of the project is established by reviewing the extensive literature. In the first subsection, the norms for selecting cited literature for this project are presented. In the second subsection, the content related to this project is investigated. Through an in-depth understanding of the intrusion detection field, it was found that there is a large amount of published research in the field of intrusion detection that confirms the application of machine learning models to intrusion detection is very effective, and based on this obtained conclusion, the overall idea of this project was determined to be feasible. Next, a summary of the various types of machine learning models that have been used, an overview of the mainstream intrusion detection datasets, and a survey of data pre-processing and hyper-parameter optimisation are presented to pave the way for the identification of the research methodology in Section 3.

CHAPTER 3 - RESEARCH METHODOLOGY

3.1. Research Methodology

In order to be able to implement an intrusion detection project based on machine learning parameter optimisation, this research adopts a quantitative research based on positivism, which is carried out through an inductive approach research strategy, approaches through experiments. where a single methodology is used in the collection part of the data and cross-validation is carried out, and finally, the results obtained from the experiments are evaluated and analysed. The flow of the research methodology of the project is shown in Figure. 8.

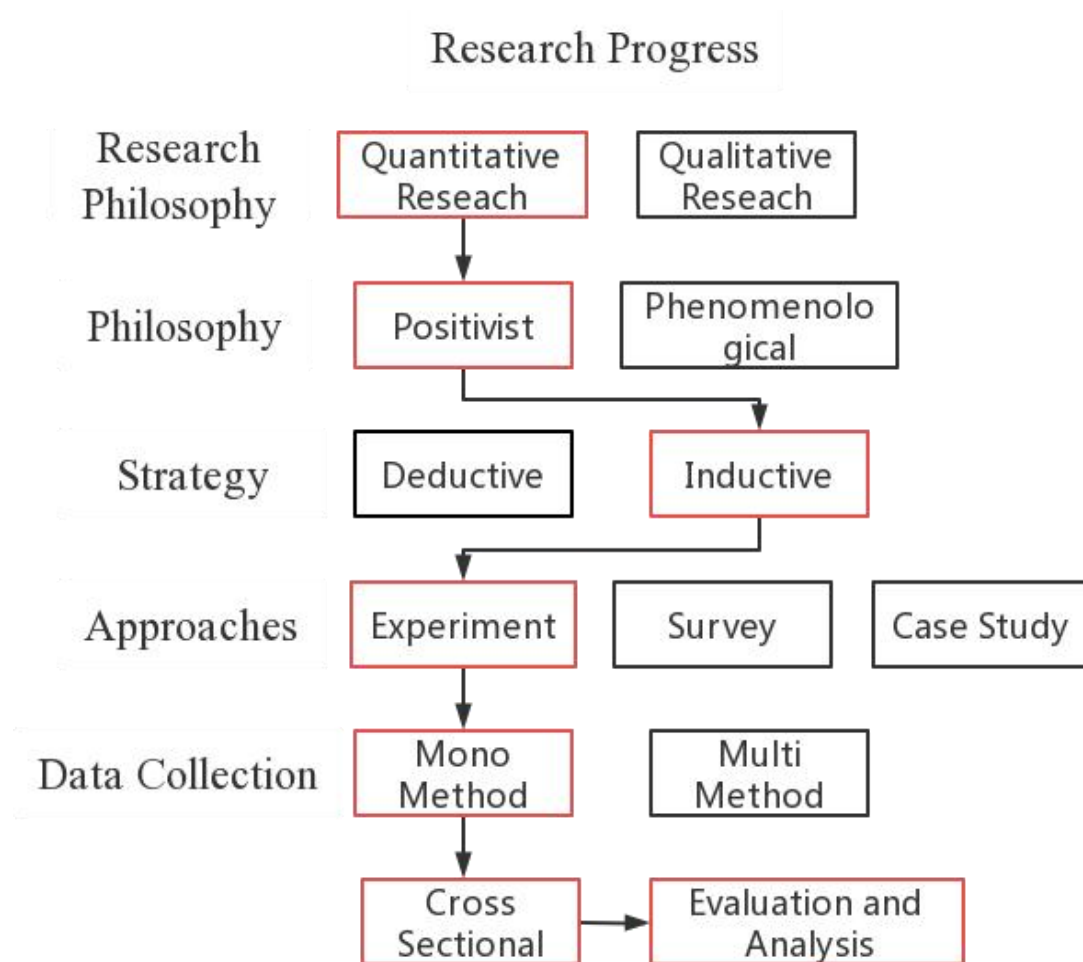


Figure 8 Research methodology overview

3.1.1. Research Philosophy

The research philosophy of this project uses quantitative research based on positivism. Positivism emphasises on testing theories and hypotheses by collecting

and analysing observable data to obtain objective, verifiable and quantifiable conclusions [71]. The strengths of positivism compared to phenomenological are its focus on objectivity, verifiability, quantifiability, scientific validity, and broad application [72]. The methodology in the project includes data collection, preprocessing, model training and parameter tuning, all these steps rely on real data and empirical observations to validate the performance of machine learning models in intrusion detection. Quantitative research focuses on collecting and analysing quantifiable data to validate hypotheses, models and relationships through statistical analysis [73]. In contrast to qualitative research, quantitative research emphasises quantifiable data and statistical analysis aimed at validating hypotheses and inferring relationships. In contrast, qualitative research emphasises in-depth understanding and description, explaining phenomena through textual and qualitative analysis [74]. In the research of this project, a machine learning model is used to train and predict intrusion detection data and focuses on the performance metrics of the model such as accuracy, recall, precision, and F1 score, which are quantifiable metrics. Also the project opted for parameter optimisation and comparative analysis, which further demonstrates the focus on quantifiable data and statistical analysis.

3.1.2. Research Strategy

The research strategy of this project is inductive research method, which is to collect and analyse specific cases or data, extract general laws, patterns or trends from them, and then infer general conclusions or theories [75]. In the research of this project, firstly, by reviewing a large amount of previous literature and comparing the differences between different machine learning models, and different datasets, a large amount of theoretical knowledge can be gained through the inductive approach. After that, the intrusion detection datasets are analysed, and the machine learning models are trained and predicted for these datasets. Then, from the experimental results, general laws will be deduced on the performance of different machine learning models applied in the field of intrusion detection, the impact of parameter optimisation and so on.

The inductive approach may also face some drawbacks, such as sample bias leading to result in inaccurate or limited generalisation of conclusions, poor data quality leading to inductive analyses that may be affected by data errors or missing data, thus affecting the reliability of the conclusions, and the conclusions of inductive studies are usually based on existing data, which may not be sufficient to cover all possible scenarios, leading to limited generalisation of conclusions to different contexts [76]. To address these issues, this project takes the following following measures to solve them, firstly, checking the completeness of the dataset as well as querying the meanings of all the labels in the dataset to ensure the applicability of the dataset. Secondly, using cross-validation, which enables a better assessment of the model's performance and generalisation ability by dividing the dataset into multiple subsets and using different data subsets when training and testing the model [77].

3.1.3. Approaches

3.1.3.1. Literature Review

In the second chapter of this project a lot of previous literature has been reviewed and it is clear from the research that most of the research on intrusion detection based on machine learning has been done through experiments through which the results can be obtained for analysis [78], [79]. Compared to surveys and case studies, experiments are a more concrete and practical method for completing projects and obtaining results. Experimentation allows for the actual processing of data sets and the analysis and interpretation of actual data to draw conclusions. In contrast, surveys and case studies are more orientated towards literature studies and theoretical analysis, and may involve observations and comparisons of past cases, or speculation and analysis based on existing theories. Experimental methods are more concrete, practical and validating in completing projects, whereas surveys and case studies are more focused on integrating and reasoning from existing knowledge [80].

3.1.3.2. Comparative Analysis

In addition to individual model performance evaluation, this project also employs comparative analysis, which is a commonly used method in analysis, especially in the

field of machine learning data analysis [81]. By comparing the performance of different machine learning models on the same dataset, the strengths and weaknesses of each model can be identified. This comparative analysis can help this research to identify the type of model that is most suitable for the intrusion detection task and provide direction for further parameter optimisation and performance enhancement.

3.1.4. Risks & Limitations

The dataset involved in the project is fully public and the use of this dataset does not involve any legal regulation and does not harm people as well as others. The machine learning model selected for the project is a conventional centralised machine learning model that does not go beyond the currently available theoretical knowledge to ensure that it can be run. However, there are still some factors that can cause the results of the project to fail: for example, overfitting or underfitting of the dataset, and inappropriate parameter settings can lead to overfitting or underfitting of the model, which can affect generalisation performance. Model training and parameter optimisation may require significant computational resources, and inadequacy may result in longer training times. Computer performance may be one of the considerations limiting the design of certain machine learning algorithms, and computational cost is also a factor to be considered; excessive computational cost can render the model meaningless.

3.1.5. Validity, Reliability & Generalisability

3.1.5.1. Validity

The project assessed the validity of the model by the results generated after running the code. The assessment of validity includes both internal and external validity [82]. Internal validity relates to the model's performance on training data, evaluating the model's accuracy, recall, and other metrics on the training dataset to determine its performance on the training dataset. External validity focuses on the model's performance on test data or previously unseen data to verify its ability to generalise to new datasets.

3.1.5.2. Reliability

The methods and techniques used in this research should be reliable and able to produce consistent results under different experimental conditions. This means that executing the same method at different times, datasets, or settings should yield similar results. For machine learning models, assessing stability and reliability can involve the use of techniques such as cross-validation to ensure that results are consistent and reliable.

3.1.5.3. Generalisability

A key goal of any research is to establish the ability of its findings to generalise beyond a specific research context [83]. For generalisability, the method of this project is valid under this dataset, if it were to be replaced with another dataset, the method would need to be adapted for the data pre-processing session, as well as optimised in terms of model design.

3.1.6. Data Evaluation Methods

3.1.6.1. Data Collection

In this project, the data collection method used was the Mono method, where the required NSL-KDD dataset is downloaded by visiting an open source website. The choice of this method was influenced by a number of factors, project budget, time constraints and data availability. The Mono method was chosen for simplicity and efficiency reasons, as the data collection process requires access to a specific data source and ensures the legitimacy and availability of the data. This method usually provides faster data acquisition and avoids involving additional data collection costs when the data is already publicly available [84], [85]. However, it is worth noting that the adoption of the Mono method may lead to data limitations, as relying on a single data source alone may not provide a complete coverage of all changes and situations within the domain, which is one of the limitations of this study.

3.1.6.2. Data Evaluation

In this project, data assessment is a crucial step to analyse and interpret the collected data and to ensure the quality, validity and consistency of the data with the objectives of the study. Data assessment involves several aspects, including statistical

analysis of the data, visualisation and treatment of outliers [86]. In the data assessment phase, the project conducted statistical analysis to calculate the values of different attack types in the dataset to make the distribution of the dataset clearer. At the same time, the evaluation means of data visualisation is adopted, visualisation can help the researcher to understand the data better, this study will adopt histogram to make the dataset more intuitive. For the treatment of outliers, since the improved version of the NSL-KDD dataset is selected for this study, it has already treated the redundant data and outliers and does not need to be processed again [87].

3.2. Development Methodology

In this project, a quantitative research methodology based on a positivist research philosophy is used to validate the performance of machine learning models in the field of intrusion detection. The specific development methodology can be summarised in the following steps: Firstly, the NSL-KDD dataset, which contains data related to intrusion detection, is obtained by Mono method. Subsequently, the data is preprocessed using data preprocessing techniques, such as normalisation or standardisation, to ensure that the data is within the same scale and to improve the stability and performance of the model. In terms of machine learning models, different models were selected for the project, including decision trees, neural networks and random forests. These models were selected for the intrusion detection task. For each model, different hyper-parameters such as the depth of the tree, the number of iterations of the neural network and the number of nodes were set to optimise the model performance. The selected models are trained using training data and parameter tuning is performed using techniques such as cross-validation. For each model, hyper-parameters are adjusted through grid search and cross-validation to achieve better generalisation and prediction performance. For each model, evaluation is carried out using test data, and performance indicators such as accuracy, recall, precision, and F1 scores are computed to assess the performance of the model. In addition, metrics such as ROC curves and AUC are used to assess the classification ability and prediction quality of the models. The performance metrics of different

models are compared and analysed to find out the optimal model and explain its performance in intrusion detection tasks. The performance differences between models in different contexts can also be analysed to provide deeper insights. Eventually, the results and analyses of the models are collated into a report or presentation document that describes in detail the research process, experimental results, and conclusions drawn. The outputs from this phase can be used to share the project's findings in academia or practical applications.

3.3. Resources

3.3.1. Hardware Resources

The hardware comes from my personal mainframe computer, including a monitor as well as the mainframe, no other hardware facilities are involved in this project.

3.3.2. Software Resources

All the software involved in this project was taken from the official network, where the virtual machine version VmarePro16 was installed, Ubuntu 20.04 system was taken from the official network, Python 3.8 was used as the programming language, and finally, after arriving at the data, the image was plotted using Origin in order to pursue the aesthetics.

3.4. Summary

In this chapter the research methodology is defined, the whole methodological choices for the research of the project are explained and how the problems that may be encountered during the research are solved. After the research methodology has been determined a complete experimental procedure will be developed and implemented in chapter four.

CHAPTER 4 - DESIGN & IMPLEMENTATION

4.1. Introduction

This chapter standardises the hardware, software used in the project by reviewing the articles in chapter 2 and the research methodology in chapter 3 thus developing the entire experimental programme. In the design section, each step of the experiment can be visualised through the flow block diagram. In the implementation section, each part of the process is implemented, including the selection of the dataset, the pre-processing of the dataset, the construction of the model, and how to characterise the results of the experiment.

4.2. Problem Formulation

The hardware facilities and software used in this project are shown in Table. 2 below. The purpose of selecting virtual machines for the experiment is that virtual machines can quickly achieve distributed deployment, can isolate different operating systems and applications, prevent conflicts between different software, and can also avoid incompatibility between different operating systems [88]. The advantage of choosing Anaconda is that it can collectively manage installation packages and configure the environment, but in this experiment, network lag was encountered several times, which is a problem to be considered. Finally, in order to pursue the aesthetics and the intuition of the results, Origin was used to draw some of the images.

Table 2 Hardware & Software

Hardware		Software	
●	Ubuntu 20.04	●	Python 3.8
●	Intel(R) Core(TM)i7-9750H CPU@2.60GHz	●	Anaconda 3.7
●	RAM 16GB	●	VMarePro 16
●	NVIDIA GeForce RTX1070	●	Origin 8.0

4.3. Design

The experimental framework of this project is designed as follows, following the recommendations of the literature [89], the specific process is as follows, and the rough flow block diagram is shown in Fig. 9:

- Determine the dataset, through the previous literature research comparing the advantages and disadvantages of several common datasets, it is determined that the NSL-KDD dataset is used for the experiments in this project, and the dataset is downloaded from the Kaggle open source website.
- Dataset preprocessing, extract a part of the overall dataset to view the composition of the dataset, and then get the overall distribution state of the dataset through the code. Next, the dataset is preprocessed by deleting the last column, naming the columns according to the dataset definition, converting English to numerical values for coding, normalising the dataset, and saving the modified dataset.
- Design six machine learning methods based on parametric optimal design for experimentation.
- Design the parameter optimisation search strategy and confirm the best parameter combinations through grid search and cross-validation.
- Save the obtained results of hyperparameter combinations of various machine learning models, save the experimental results of test results on the test set, and evaluate each model by accuracy, Precision, Recall, F1score, and ROC-AUC curves.

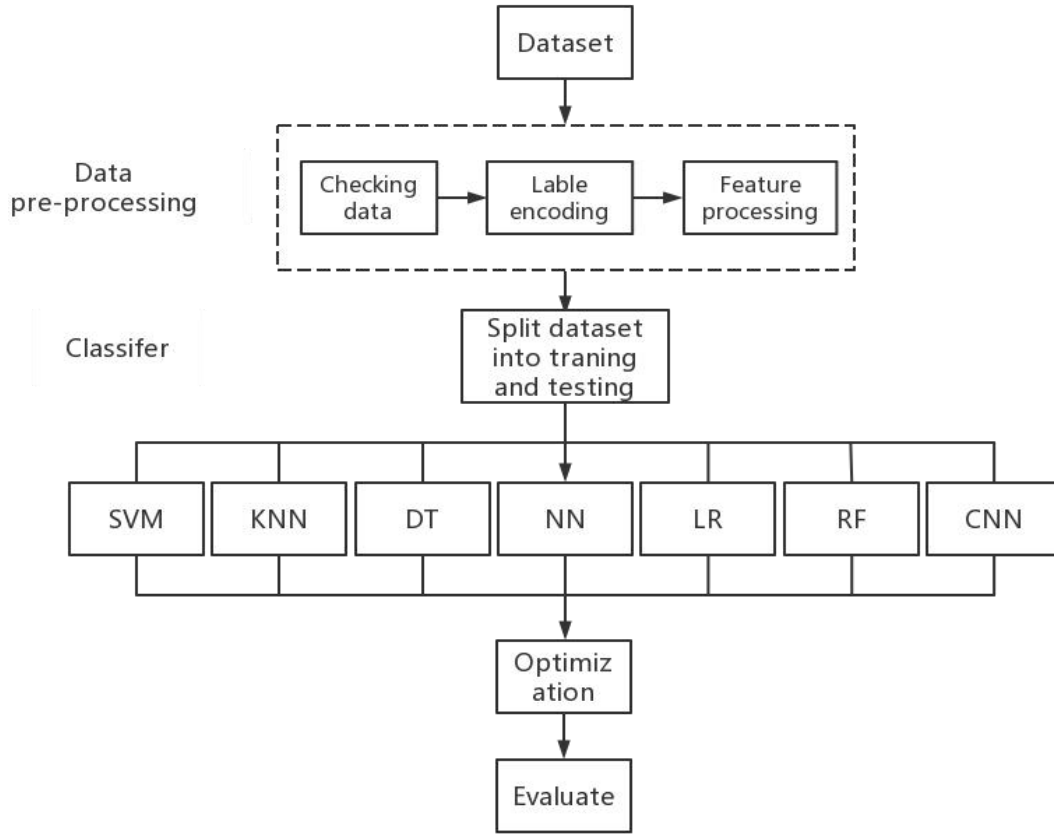


Figure. 9 Experimental process

4.4. Implementation

4.4.1. Dataset pre-processing

Each record in the NSL-KDD dataset has 43 columns, where the 42nd column represents whether the record is labelled as normal or abnormal, and is the label column in the machine learning dataset, and the last column is a number between 0 and 21, which represents the time at which it was correctly classified by the machine learning model, and its value is increased by 1 each time an instance is correctly classified and is deleted during the data preprocessing phase. At this point there are 42 columns left for each record and the data preprocessing consists of the following stages:

Checking the data, open the NSL-KDD dataset and after that view the initial state of the data as shown in Figure 10. By looking at it, it can be found that the dataset does not have the labels of the columns, through the arff file it is possible to view the

names and descriptions of the attributes about these features, and the specific meanings of the attributes of these labels are shown in Table. 3:

Table 3 43 characterisations in NSL-KDD

Attribute	Type	Description
duration	Numeric	Connection length (duration in seconds)
protocol type	Nominal	Protocol type, e.g. TCP, UDP, TCMP.
service	Nominal	Network service of the target host, e.g. http, telnet.
flag	Nominal	Connection status flag (normal or error)
src bytes	Numeric	Size of the data from the data source to the target host
dst bytes	Numeric	Size of data from target host to data source
land	Nominal	Whether the connection is on the same host or port (1 is yes, 0 is no, same as below)
wrong fragment	Numeric	Number of error fragments
urgent	Numeric	Number of emergency packets
hot	Numeric	Number of hotspot indicators
num failed logins	Numeric	Number of failed login attempts
logged in	Nominal	Successful login or not
num compromised	Numeric	Number of connection quality compromise conditions
root shell	Numeric	Availability of program run privileges
su attempted	Numeric	Whether or not you have super privileges
num root	Numeric	Number of accesses accepted by root
num file creations	Numeric	Number of file creation operations
num shells	Numeric	Number of shell prompts
num access files	Numeric	Number of operations on access control files
num outbound cmds	Numeric	Number of commands open to the public in FTP dialogue
is host login	Nominal	Whether the login is a hotlist login
is guest login	Nominal	Number of guest logins
count	Numeric	Number of simultaneous connections to the same host as the current connection
srv count	Numeric	Number of connections to the same service as the current connection
error rate	Numeric	Percentage of connections to hosts with "SYN" errors

srv serror rate	Numeric	Percentage of connections to services with "SYN" errors
error rate	Numeric	Percentage of connections to hosts with "REJ" errors
srv rerror rate	Numeric	Percentage of connections to services with "REJ" errors
same srv rate	Numeric	Percentage of connections with the same service
diff srv rate	Numeric	Percentage of connections with different services
srv diff host rate	Numeric	Percentage of connections to different hosts
dst host count	Numeric	Number of network packets connected to the same host
dst host srv count	Numeric	Number of connections to the same service network package
dst host same srv rate	Numeric	Number of connections to the same service
dst host diff srv rate	Numeric	Number of connections to different services
dst host same src port rate	Numeric	Number of connections to the same port
dst host srv diff host rate	Numeric	Number of connections to different hosts
dst host serror rate	Numeric	Percentage of hosts with "SYN" error connections
dst host srv serror rate	Numeric	Percentage of services with "SYN" error connections
dst host rerror rate	Numeric	Percentage of hosts with "REJ" error connections
dst host srv rerror rate	Numeric	Percentage of hosts with "REJ" error connections
Type of attack	Nominal	There are a total of 40 attacks and 1 normal
level of attack	Numeric	level of attack

0,tcp,http,SF,199,420,0,0,0,0,1,0,0,0,0,0,0,0,0,30,32,0.00,0.00,0.00,0.00,1.00,0.00,0.09,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal,21
0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,19,0.00,0.00,1.00,1.00,0.16,0.06,0.00,255,19,0.07,0.07,0.00,0.00,0.00,0.00,1.00,1.00,neptune,21
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1.00,1.00,0.00,0.00,0.05,0.06,0.00,255,9,0.04,0.05,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,117,16,1.00,1.00,0.00,0.00,0.14,0.06,0.00,255,15,0.06,0.07,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21
0,tcp,remote_job,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,270,23,1.00,1.00,0.00,0.00,0.09,0.05,0.00,255,23,0.09,0.05,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,8,1.00,1.00,0.00,0.00,0.06,0.06,0.00,255,13,0.05,0.06,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21
0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,205,12,0.00,0.00,1.00,1.00,0.06,0.06,0.00,255,12,0.05,0.07,0.00,0.00,0.00,0.00,1.00,1.00,neptune,21
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,199,3,1.00,1.00,0.00,0.00,0.02,0.06,0.00,255,13,0.05,0.07,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21
0,tcp,http,SF,287,2251,0,0,0,0,1,0,0,0,0,0,0,0,0,3,7,0.00,0.00,0.00,0.00,1.00,0.00,0.43,8,219,1.00,0.00,0.12,0.03,0.00,0.00,0.00,0.00,normal,21
0,tcp,ftp_data,SF,334,0,0,0,0,0,1,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,2,20,1.00,0.00,1.00,0.20,0.00,0.00,0.00,0.00,warezclient,15
0,tcp,name,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,233,1,1.00,1.00,0.00,0.00,0.00,0.06,0.00,255,1,0.00,0.07,0.00,0.00,1.00,1.00,0.00,0.00,neptune,19
0,tcp,netbios_ns,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,96,16,1.00,1.00,0.00,0.00,0.17,0.05,0.00,255,2,0.01,0.06,0.00,0.00,1.00,1.00,0.00,0.00,neptune,18
0,tcp,http,SF,300,13788,0,0,0,0,1,0,0,0,0,0,0,0,0,8,9,0.00,0.11,0.00,0.00,1.00,0.00,0.22,91,255,1.00,0.00,0.01,0.02,0.00,0.00,0.00,0.00,normal,21
0,icmp,eco_i,SF,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,16,1.00,0.00,1.00,1.00,0.00,0.00,0.00,0.00,ipsweep,18
0,tcp,http,SF,233,616,0,0,0,0,1,0,0,0,0,0,0,0,0,3,3,0.00,0.00,0.00,0.00,1.00,0.00,0.00,66,255,1.00,0.00,0.02,0.03,0.00,0.00,0.02,0.00,normal,21
0,tcp,http,SF,343,1178,0,0,0,0,1,0,0,0,0,0,0,0,0,9,10,0.00,0.00,0.00,0.00,1.00,0.00,0.20,157,255,1.00,0.00,0.01,0.04,0.00,0.00,0.00,0.00,normal,21
0,tcp,mtp,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,223,23,1.00,1.00,0.00,0.00,0.10,0.05,0.00,255,23,0.09,0.05,0.00,0.00,1.00,1.00,0.00,0.00,neptune,18
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,280,17,1.00,1.00,0.00,0.00,0.06,0.05,0.00,238,17,0.07,0.06,0.00,0.00,0.99,1.00,0.00,0.00,neptune,18
0,tcp,http,SF,253,11905,0,0,0,0,1,0,0,0,0,0,0,0,0,8,10,0.00,0.00,0.00,0.00,1.00,0.00,0.20,87,255,1.00,0.00,0.01,0.02,0.00,0.00,0.00,0.00,normal,21
5607,udp,other,SF,147,105,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,1,0.00,0.85,1.00,0.00,0.00,0.00,0.00,normal,21
0,tcp,mtp,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,248,2,1.00,1.00,0.00,0.00,0.01,0.06,0.00,255,2,0.01,0.06,0.00,0.00,1.00,1.00,0.00,0.00,neptune,19
507,tcp,telnet,SF,437,14421,0,0,0,0,1,3,0,0,0,0,0,1,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,25,0.10,0.05,0.00,0.00,0.53,0.00,0.02,0.16,normal,20
0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,279,7,1.00,1.00,0.00,0.00,0.03,0.06,0.00,255,13,0.05,0.07,0.00,0.00,1.00,1.00,0.00,0.00,neptune,21
0,tcp,http,SF,227,6588,0,0,0,0,1,0,0,0,0,0,0,0,0,5,22,0.00,0.00,0.00,0.00,1.00,0.00,0.18,43,255,1.00,0.00,0.02,0.14,0.00,0.00,0.56,0.57,normal,21

Figure. 10 View Dataset

A detailed description of the NSL-KDD dataset, which contains a total of 39 attack types and 1 normal type, was obtained by searching for relevant information. Among them, only 22 attack types and 1 normal type are included in the training set, and there are 40 types in the testing set, and 17 attack types that were not involved in training in the training set appear [90]. The presence of these attack types makes it more difficult to identify them during model testing. Specific attack types are shown in Table. 4, which can be divided into four broad categories based on different class names: Dos, Probe, R2L, and U2R. DoS attacks cause an increase in the use of system resources or network traffic, resulting in the inability of the system to provide services and causing network failures. u2R attacks are designed to elevate the privileges of an ordinary user to the privileges of the system administrator (root), thereby gain greater control over the system. R2L Attacks are designed to compromise the local environment of a target system from a remote location (e.g., the Internet) and gain unauthorised access. Probe Attacks have the primary goal of probing for vulnerabilities, services, and host information that exists in the target network in order to prepare for subsequent attacks.

Table. 4 Attack types

Class	Training set
DoS	Back, Land, Neptune, Pod, Sumrf, Teardrop, Mailbomb, Processtable, Udpstorm, Apache2, Worm (11)
Probe	Satan, Ipsweep, Nmap, Portswep, Mscan, Saint (6)
R2L	Guess_password, Ftp_write, Imap, Phf, Multihop, Warezmater, Warezclient, Spy,

	Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named (15)
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps (7)

This dataset also suffers from the problem of data imbalance which is common in classification tasks, it will lead to lower accuracy of the model in classifying a few classes and reduce the detection performance of the model. The data of the NSL-KDD dataset can be classified into five types such as Normal, DoS, Probe, R2L, U2R, etc., which will count the attacking and normal ones separately, and in the training set there are 67,343 normal samples and 58,630 attack samples. In the test set, there are 9711 normal samples and 12833 attack samples, and the sample distribution is detailed in Table. 5.

Table. 5 Attack types statistics

Class	Training Set	Test Set
Normal	67343	9711
DoS	45927	7460
Probe	11656	2421
R2L	995	2885
U2R	52	67
Attack	58630	12833
Total	125973	22544

Due to the presence of English language in the dataset and the fact that most machine learning models can only process numerical data, converting the text to numeric enables the model to process the data and perform training and prediction. Therefore it is necessary to convert the non-numeric features into numeric form, the following is the scheme adopted in this project, the exact results of the process will be shown in the next section, firstly the dataset is biclassified by leaving the normal value (normal) of column 42 unchanged and converting all attack types to abnormal.

Normalisation (feature scaling) ensures that features have the same scale, which is useful for certain machine learning algorithms, especially those that rely on distance-based computations such as K nearest neighbours. Pre-processing the dataset using MinMaxScaler makes the feature vectors of each sample normalised in length, which helps the machine learning model to process the data better. MinMaxScaler is a data preprocessing method used to normalise each sample vector of the dataset so that

the Euclidean length (the number of paradigms) of each sample vector is 1 [91]. The specific formula is given below:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (\#4.1)$$

Where X is the original data, X_{min} is the minimum value of the data, X_{max} is the maximum value of the data and $X_{normalized}$ is the normalised data.

Finally, dataset division, dividing the dataset into training and test sets, for ML model training, the original training dataset should be divided into training and validation sets, and the original test dataset should be used for evaluating all models.

4.4.2. Machine Learning Modelling

4.4.2.1. Support vector machine

Importing the library containing the SVM model (SVC) from GridSearchCV, sklearn, SVC(kernel='rbf') is used to create a non-linear kernel for the SVM classifier model, considering the characteristics of this dataset, high-dimensionality, large data volume, etc. It is very useful in mapping the data into a high dimensional space, thus making it easier to separate the data in a high dimensional space, solving the problem of linear indivisibility in the original space. A parameter grid is then defined and param_grid contains the different combinations of hyperparameters to be tried in the grid search. These hyperparameters include the regularisation parameter C and the gamma parameter of the RBF kernel function. Afterwards a parameter-optimised object is created using GridSearchCV, the SVM model and parameter grid are passed to it, and the model is fitted on the training data. A new SVM model, best_svm_model, is created using the best parameters found, and the model is evaluated on the test data.

4.4.2.2. K-nearest neighbours

Import the KNeighborsClassifier in sklearn, next copy the data, define the KNN model and create the K Nearest Neighbours Classifier model via KNeighborsClassifier(). Define the parameter grid, param_grid contains combinations of different hyperparameters to be tried in the grid search. These hyperparameters include the choice of the number of neighbours ('n_neighbors') and the type of weight

function ('weights'). A parameter-optimised object is created using GridSearchCV, and the model is fitted on the training data to most output the best combination. Next, use the best parameters to build the final KNN model: create a new K-nearest neighbour model, `best_knn_model`, using the best parameters found, while evaluating the model on the test data.

4.4.2.3. Decision tree

Start by importing the necessary libraries and modules, including `DecisionTreeClassifier` for creating decision tree classifiers, `GridSearchCV` for parameter optimisation, and a variety of metrics and visualisation tools for evaluating model performance. Duplicate the training and test sets to ensure that the original data is not modified. Define the decision tree model and initialise it to `DecisionTreeClassifier()`. Define the parameter grid `param_grid`, including different splitting criteria ('gini' and 'entropy'), splitting strategies ('best' and 'random'), and maximum depth options (None and some integer values). The `GridSearchCV` object is created for fitting the decision tree model under different combinations of parameters and cross-validation is performed. The evaluation metric is accuracy. Fit the model on the training data and perform a grid search via `grid_search.fit()`. Output the best combination of parameters and store it in `best_params`. Construct the final decision tree model using the best parameters, which include `max_depth`, `min_samples_split`, and `min_samples_leaf`. replicate the validation set. Evaluate the best model on the test data and calculate accuracy, generate classification reports and confusion matrices, and then print out these metrics. Calculate probability scores for plotting ROC curves. Plot ROC curves to visualise the performance of the model.

4.4.2.4. Logistic regression

Create a logistic regression model: `LogisticRegression()` is used to create a logistic regression classifier model. Define a parameter grid: `param_grid` contains combinations of different hyperparameters to be tried in the grid search. These hyperparameters include the value of the regularisation parameter `C` ('C'), the regularisation type ('penalty') and the choice of solver ('solver'). Create a

GridSearchCV object to perform a grid search via `grid_search.fit()` to find the best combination of parameters. Build the final logistic regression model using the best parameters: use the best parameters found to create a new logistic regression model, `best_logistic_model`. subsequently evaluate the model on test data.

4.4.2.5. Neural network

Firstly the module is still imported, next the data is copied and subsequently the model is created, using the `create_neural_network` function which defines a function for constructing a neural network model that consists of one or more hidden layers and an output layer, which can be parameterised to pass different hidden layer sizes, activation functions and optimisers. The model uses binary cross entropy as the loss function and is optimised using the passed optimiser. Next a parameter grid is defined. `param_grid` contains combinations of different hyperparameters to try in the grid search. These hyperparameters include different 'hidden_layer_sizes', 'activation', and 'optimiser'. Using the KerasClassifier wrapper `create_neural_network` function, a wrapper for the Keras model was created so that it could be passed to GridSearchCV for parameter search. At the same time, set the model's training-time hyperparameters, including training epoch and batch_size. use GridSearchCV to create a parameter-optimised object to which the Keras model wrapper and parameter mesh can be passed. Finally, the model is fitted on the training data, the best combination of parameters is output and the final neural network model is constructed using the best parameters and evaluated on the test data.

4.4.2.6. Random forests

Firstly, the necessary libraries and modules are imported, including GridSearchCV for parameter optimisation, RandomForestClassifier for creating a random forest classifier graph tool, these libraries are used for data processing, model training. Data replication: data from the original training and test sets are copied into new variables (`x_traincpy`, `y_traincpy`, `x_testcpy`, `y_testcpy`) to ensure that the original data are not modified. A random forest classifier model with default parameters was created with `RandomForestClassifier()`. A parameter grid was defined that used `param_grid` to

contain combinations of different hyperparameters to be tried in the grid search. These hyperparameters include the number of decision trees (`n_estimators`), the maximum depth (`max_depth`), the minimum number of split samples for internal nodes (`min_samples_split`), and the minimum number of samples for leaf nodes (`min_samples_leaf`). Create a parameter-optimised object using `GridSearchCV`, passing the random forest model and parameter grid to it. Evaluate the model performance using 3-fold cross-validation, with `scoring='accuracy'` indicating that accuracy was used as the evaluation metric. Perform a grid search via `grid_search.fit()` to find the best parameter combination. Output the best combination of parameters: print out the best combination of parameters found through the grid search that will be used to build the final random forest model. Use the best parameters found to create a new random forest classifier i.e. `best_random_forest`. the data from the original validation set is copied into the new variables (`x1_testcpy` and `y1_testcpy`) to ensure that the original data is not modified. A random forest model with the best parameters was fitted and predicted on the test data.

4.4.3. Optimal design of machine learning model parameters

In classification models, parameter optimisation is an important means to improve the performance of the model. The parameter optimisation methods used in SVM are also cross-validation and grid search methods. For the parameter optimisation of the Gaussian kernel function of SVM, the `GridSearchCV()` function or `cross_val_score()` is mainly used in the scikit-learn library, and the `GridSearchCV()` function is the main function in the scikit-learn library. `GridSearchCV()` function method. Through this direct call, it greatly improves the efficiency of program writing. The parameters for tuning in the experiment are `C` and `gamma`, for the KNN model, the parameter optimisation method is also cross-validation and grid search method, the parameters for tuning in the experiment are `n_neighbors`, `weights`, the parameter optimisation method for the decision tree is also cross-validation and grid search method, the specific tuning steps and the function used are the same as the previous one. The parameters used for tuning in the experiments are `Criterion`, `splinter` and

max_depth. For the logistic regression model, the parameter optimisation method is also a cross-validation and lattice search method, which uses different regularisation strengths, C, regularisation type, penalty and solver to construct and optimise the logistic regression model. A parametric grid suitable for neural networks is selected, usually a Multi-Layer Perceptron (MLP) configuration. The parameters for which tuning was performed in the experiments were hidden_layer_sizes, activation, and optimizer. the same GridSearchCV technique was used to systematically search for different parameter combinations in order to find the best model configuration in order to maximise model performance. In essence, random forest parameter optimisation is a search problem that finds a set of approximate global optimal solutions by performing a finite number of search traversals in a parameter space of countless parameter combinations according to a certain search strategy formulated in order to optimise the performance of the classification model. The parameter optimisation methods used in this paper are grid search and cross-validation methods. In python programming language, you can install the sklearn machine learning library to achieve the cross-validation and lattice search method and other parameter optimisation methods. For example, calling cross_val_score() in the sklearn.model_selection library enables cross-validation of the optimisation parameters; calling GridSearchCV() in the sklearn.model_selection library enables grid search of the optimisation parameters. The parameters for tuning in the experiment are n_estimators, max_depth, min_samples_split and min_samples_leaf.

4.4.4. Results output

Via sklearn.metrics Import some functions from the sklearn.metrics module that are used to compute the evaluation metrics of the model, using accuracy_score (y_test, predicted) to compute the accuracy, respectively. Use recall_score (y_test, predicted, average="binary") to compute recall. Use precision_score (y_test, predicted, average="binary") to compute precision. Calculate the F1 score using f1_score (y_test, predicted, average="binary"). Extract the probability of a positive category (category 1) predicted by the model via proba[:, 1] for subsequent plotting of the ROC curve.

Calculate the False Positive Class Rate (FPR) and True Class Rate (TPR) of the ROC curve using `roc_curve` (`y_test`, `proba_positive`). Import the `matplotlib.pyplot` module and plot the ROC curves to highlight the performance of the model at different thresholds. Finally save the file.

4.5. Summary

Chapter 4 completes the design and implementation, completing the flow of the design part through the implementation, showing some of the code, and explaining each part of the implementation through detailed descriptions. In the data preprocessing section, viewing the data, feature processing, and label conversion were completed. In the model section, seven machine learning models with the ability to select parameters for optimisation were built, and finally how the results were plotted as well as calculated was described. The testing process and results of all the implementation sessions are described in the next section.

CHAPTER 5 - TESTING AND EVALUATION

5.1. Testing

5.1.1. Dataset pre-processing

The test part will be implemented one by one according to the design part of the previous section. First of all follow the steps, open the dataset to view the dataset, as the dataset does not have the names of the columns, so give it the names of the columns as shown below:

```
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','error_rate','srv_error_rate','error_rate','srv_error_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_error_rate','dst_host_srv_error_rate','dst_host_error_rate','dst_host_srv_error_rate','attack_type','degree'])
```

Subsequently, the last column 'degree' is removed using `drop('degree', axis=1)`, after which the last column 'degree' is removed by `random_sample = data.sample(n=5)` `print(random_sample)` to see the 5 rows in the dataset randomly viewed after naming as in Figure 11.

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	\
29680	0	tcp	ftp_data	SF	641	0	0	
107616	0	tcp	http	SF	291	8901	0	
104982	0	tcp	http	SF	281	34293	0	
108222	0	tcp	klogin	S0	0	0	0	
44709	0	tcp	efs	S0	0	0	0	

	wrong_fragment	urgent	hot	...	dst_host_srv_count	\
29680	0	0	0	...	29	
107616	0	0	0	...	255	
104982	0	0	0	...	255	
108222	0	0	0	...	19	
44709	0	0	0	...	14	

	dst_host_same_srv_rate	dst_host_diff_srv_rate	\
29680	0.26	0.04	
107616	1.00	0.00	
104982	1.00	0.00	
108222	0.07	0.07	
44709	0.05	0.05	

	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	\
29680	0.26	0.00	
107616	0.02	0.02	
104982	0.09	0.03	
108222	0.00	0.00	
44709	0.00	0.00	

	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate	\
29680	0.0	0.0	0.0	
107616	0.0	0.0	0.0	
104982	0.0	0.0	0.0	
108222	1.0	1.0	0.0	
44709	1.0	1.0	0.0	

	dst_host_srv_rerror_rate	attack_type
29680	0.0	normal
107616	0.0	normal
104982	0.0	normal
108222	0.0	neptune
44709	0.0	neptune

[5 rows x 42 columns]

Figure. 11 Five random rows in the data set

To view the distribution status of the dataset, first view all Attack types in the sample, classify the categories into training set normal, abnormal and test set normal, abnormal to facilitate the subsequent binary classification problem, after which a pie chart is plotted as shown in Fig. 12.

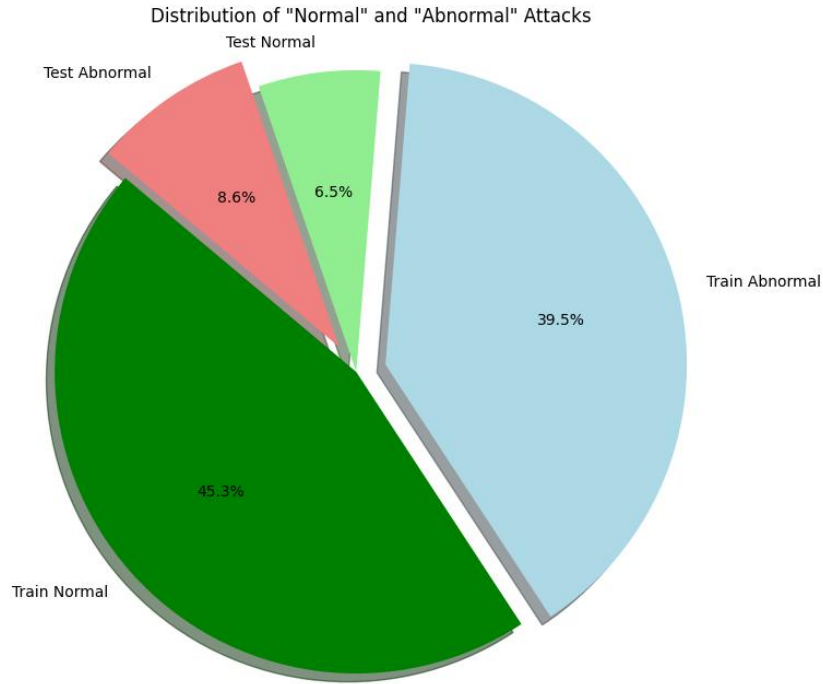


Figure. 12 Attack types and normal types

The dataset is encoded and the categorical variables are converted to numeric labels by Label Encoder. This is necessary for many machine learning models as they can usually only handle numeric data. With the `fit_transform` method, `LabelEncoder` learns the different categories of each feature and maps them to unique integer labels. In this way, the original categorical variables are transformed into numeric labels that can be used for machine learning model training. To do this, the text labels in the "protocol_type", "service", "flag" and "attack_type" columns are converted to numeric labels using `LabelEncoder` for use in the machine learning model. After conversion the data is viewed through `data.describe()` as shown in Figure. 13.

duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment
25972.000000	125972.000000	125972.000000	125972.000000	1.259720e+05	1.259720e+05	125972.000000	125972.000000
287.146929	1.053202	31.226558	6.97998	4.556710e+04	1.977927e+04	0.000198	0.022688
2604.525522	0.426621	16.346504	2.68937	5.870354e+06	4.021285e+06	0.014086	0.253531
0.000000	0.000000	0.000000	0.00000	0.000000e+00	0.000000e+00	0.000000	0.000000
0.000000	1.000000	20.000000	5.00000	0.000000e+00	0.000000e+00	0.000000	0.000000
0.000000	1.000000	24.000000	9.00000	4.400000e+01	0.000000e+00	0.000000	0.000000
0.000000	1.000000	49.000000	9.00000	2.760000e+02	5.160000e+02	0.000000	0.000000
42908.000000	2.000000	69.000000	10.00000	1.379964e+09	1.309937e+09	1.000000	3.000000

Figure. 13 Encoded dataset

Next the dataset is divided into training set and test set and normalised features are applied to the dataset. After processing, the training set is distributed as ((100777, 41), (25195, 41)) and the test set is distributed as (22543, 41).

5.1.2. Machine Learning Model Parameter Optimisation

5.1.2.1. SVM parameter optimisation

- Select C parameter

The regularisation parameter of SVM which controls the complexity of separating hyperplanes. The penalty coefficient C is chosen to take values in the range of [0.1, 1, 10, 50, 100, 150, 200, 250, 300], where smaller values indicate stronger regularisation and help to prevent overfitting, while larger values reduce the strength of the regularisation and allow for a better fit to the training data.

- Select gamma parameters

The coefficients of the kernel function, which controls the influence range of the sample points in the feature space. Taking the values [0.01, 0.1, 1, 1.5, 2], smaller values indicate a larger influence range of sample points and a smoother shape of the kernel function, while larger values will result in a smaller influence range and a more complex shape of the kernel function.

5.1.2.2. KNN parameter optimisation

- Select the n_neighbors parameter

The n_neighbors is a key parameter in the KNN model and indicates the number of nearest neighbours to be considered. In the selected parameter grid, different numbers of neighbours were tried including 1, 3 and 5.

- Select the weights parameter

weights is another parameter in the KNN model that is used to determine the weights of the neighbours. It can be either 'uniform' (all neighbours have the same weight) or 'distance' (weights are inversely proportional to distance). These two different weighting options were tried in the selected parameter grid.

5.1.2.3. DT parameter otimisation

- Select criterion and splitter parameters

In the initial state, select the parameter combination to be optimised for the criterion parameter as (gini, entropy) and the parameter combination to be optimised for the splitter parameter as (best, random), then arrange all possible combinations of the values of criterion and splitter, and generate a "lattice" of all possible combinations. Generate a "lattice" and select the nodes from the generated lattice search interval, i.e. the pairs of criterion and splitter parameters (gini, entropy) and (best, random).

- Select the max_depth parameter

In the initial state, the search interval for the max_depth parameter is [None, 50], and the parameter search step is set to 10, after which the node is moved from the starting point None by a unit step of 10 in the direction of the parameter's growth.

5.1.2.4. LR parameter optimisation

- Select the C parameter

C controls the strength of the regularisation. Initial values are [1, 10, 100, 200, 300], where smaller values indicate strong regularisation and larger values indicate weak regularisation. Regularisation helps to prevent overfitting of the model.

- Select the penalty parameter

Penalty is the type of regularisation. The initial values are ['l1', 'l2', 'elasticnet'], which represent L1 regularisation, L2 regularisation and elasticnet regularisation respectively. Different regularisation types are penalising different types of model parameters and affecting the generalisation ability of the model.

- Select the solver parameter

The solver is the algorithm used to optimise the problem. Taking the values ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], different algorithms use different optimisation strategies when solving for the parameters of the logistic regression model. For example, 'liblinear' is suitable for small datasets while 'saga' is suitable for large datasets.

5.1.2.5. NN parameter optimisation

- Select the `hidden_layer_sizes` parameter

This is the parameter for the size of the hidden layer in the neural network. It indicates that the number in each tuple is the number of neurons in a hidden layer. For example, (128,) denotes a hidden layer containing 128 neurons and (64, 32) denotes two hidden layers containing 64 and 32 neurons respectively. In this experiment, four different structures are designed which are (128,), (64, 32), (128, 64), (128, 64, 32) can be used to build neural networks of different sizes.

- Select the activation parameter

This is the choice of the activation function for the hidden layer, which can be either `relu` (modified linear unit) or `tanh` (hyperbolic tangent). The activation function is used to introduce non-linear properties into the neural network. The parameter grid for this experiment includes these two different activation function options.

- Select the optimiser parameter

This is the choice of optimiser used to train the neural network, either `adam` (Adam optimiser) or `sgd` (stochastic gradient descent). Different optimisers have different optimisation strategies and may perform differently for different problems.

- Select the epochs parameter

The number of iterations has a great impact on the neural network model, and in general, the appropriate number of epochs needs to be experimented and monitored during the training process in order to obtain good performance while avoiding overfitting. In this experiment, only 10, 50, and 100 iterations were done considering the computer performance limitations.

5.1.2.6. Random forest parameter optimisation

- Select the `max_depth` parameter

In the initial state, the approximate range of search interval for the `max_depth` parameter is selected as `[None, 50]`, and the parameter search step is set to 10. Due to the high dimensionality and large scale of the selected dataset, a deeper decision tree may lead to overfitting. Consider reducing the larger value of 50 in the option for `max_depth` to a smaller value `[None, 30]` to limit the depth of the tree.

- Select the `n_estimators` parameter

Increasing the number of trees in a random forest usually helps to improve the performance of the model, especially for large datasets. Try to increase the options of `n_estimators`, e.g. 200, 300 or more, to improve the robustness of the model. In this experiment, we selected the range [50, 500] and set the parameter search step to 50.

- Select the `min_samples_split` parameter

This is the minimum number of samples required for internal node splitting. This parameter controls the growth process of the decision tree, and smaller values may result in the tree growing deeper. The parameter grid for this experiment includes different minimum sample split values, 1, 2 and 5.

- Select the `min_samples_leaf` parameter

This is the minimum number of samples required for leaf nodes. Smaller values will allow the model to create more complex leaf nodes and larger values will result in simpler leaf nodes. The parameter grid for this experiment includes different minimum sample leaf values, 0.1, 1 and 2

5.2. Evaluation

The most commonly used method for model performance evaluation in intrusion detection research is to calculate the classification confusion matrix of the classifier with respect to the test samples, which is a very intuitive reflection of the distribution of the intrusion detection model's prediction results of the test traffic with respect to the true category of the test traffic [92]. The classification confusion matrix of the intrusion detection model is shown in Table sss, where the true case (TP) denotes the number of attack records that are successfully detected as attack traffic by the classification model, the true counterexample (TN) denotes the number of normal records that are successfully detected as normal traffic by the classification model, the false positive (FP) denotes the number of normal records that are incorrectly labelled as attack traffic by the classification model, and the false negative (FN) denotes the number of attack records that are incorrectly labelled as normal traffic by the model. (FN) denotes the number of attack records incorrectly labelled as normal traffic by the

model, and the total test sample size can be obtained by summing the number of row and column elements. In order to check the detection performance of the model in the field of intrusion detection, Accuracy, Precision, Recall, F1 value, etc. are often used as evaluation indexes, and the formulas of the evaluation indexes are shown below:

- Confusion matrix: the confusion matrix can intuitively see the distribution between the model's predicted labels and the real labels, the binary classification is a 2*2 matrix, and the multiclassification is a n*n matrix, where n is the number of classifications, and the definition of the confusion matrix is shown in Table 6.

Table. 6 Confusion matrix

	Number of samples with positive predictions	Number of samples with negative predictions
Number of samples with positive true	TP	FN
Number of samples that are true negative	FP	TN

- Accuracy: is the ratio of all samples that are correctly predicted, it is a commonly used evaluation metric to measure the overall goodness of the model and is calculated as shown in Equation 5.1

$$Accuracy = \frac{TP+TN}{TP+FN+FP+TN} \text{(#5.1)}$$

- Precision: represents the proportion of samples in the class that truly belong to the class when the prediction made is for a particular class. The formula is as follows:

$$Precision = \frac{TP}{TP+FP} \text{(#5.2)}$$

- Recall: also known as the check all rate, represents the percentage of samples in a category that were correctly predicted. The formula is as follows::

$$Recall = \frac{TP}{TP+FN} \text{(#5.2)}$$

- F1: The F1 value is a comprehensive evaluation of the model and represents the reconciled average of the recall and precision rates, calculated as follows:

$$F1 = \frac{2*Precision*Recall}{Precision+Recall} \text{(#5.3)}$$

- ROC curve (Receiver Operating Characteristic curve) is one of the important evaluation tools for assessing the performance of machine learning models, it is a graphical tool for evaluating the performance of classification models. It shows the relationship between the true positive rate, also known as the recall rate, and the false positive rate at different thresholds. The horizontal axis of the ROC curve is the false positive rate and the vertical axis is the true positive rate. Each point indicates the performance of the model at a particular threshold, and each point on the curve corresponds to a different threshold setting. If the ROC curve is close to the upper left of the graph, it means that the detection rate is higher, the false alarm rate is lower, and the detection performance is superior; and the larger the area under the ROC curve, the more superior the detection performance of the model is represented, so the ROC curve can be used to intuitively evaluate the strengths and weaknesses of intrusion detection methods.
- AUC (Area Under Curve) is the area under the ROC curve surrounded by the coordinate axis, which is the weighted average of the true positive rate and the false positive rate, and is mainly used to measure the performance of machine learning algorithms to deal with dichotomous problems. A positive sample and a negative sample are randomly selected, and the probability that the current machine learning algorithm ranks this positive sample in front of the negative sample according to the calculated value is the AUC value. Obviously, the larger the AUC value, the greater the probability that the algorithm will rank the positive sample in front of the negative sample, so that it can better complete the classification task. AUC is a probabilistic value, so the maximum value of 1, and the ROC curve is generally in the straight line $y = x$, that is, the expected probability of the bisection problem 0.5, so the minimum value is usually 0.5.

5.3. Summary

This chapter completes the full range of test experiments, which includes all the steps from the previous chapter, ensures the reliability of the scheme, and describes how the model was evaluated. All the data and illustrations obtained from the

experimental results have been presented in this chapter, and a summary of the specific patterns, as well as the findings, will be presented in the next chapter.

CHAPTER 6 - CONCLUSIONS & RECOMMENDATIONS

6.1. Experimental results

The final results of the experiment were obtained through testing and evaluation in Chapter 5, which will be shown in the following three subsections. It is worth noting that for the SVM model, due to the excessive time (more than 2 days) without results, the model was taken out of operation considering its poor applicability.

6.1.1. Optimal parameter design results

The grid search of the test phase in the previous subsection leads to the following results, specifically the optimal parameter design for each model as shown in Table 7.

Table 7 Results of optimal design of model parameters	
Type	Optimal parameter design
SVM	unknown
KNN	'n_neighbors': 3, 'weights': 'distance'
DT	'criterion': 'gini', 'max_depth': 40, 'splitter': 'random'
LR	'C': 200, 'penalty': 'l2', 'solver': 'lbfgs'
NN	'activation': 'tanh', 'hidden_layer_sizes': (128, 64), 'optimizer': 'adam'
RF	'max_depth': 30, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200

6.1.2. Results of machine learning model runs

6.1.2.1. KNN

The project's K-Nearest Neighbors model performed well on the test dataset, achieving an accuracy of about 79%. The model successfully captured the majority of instances in category 1, which is normal, with more general precision (68%) and recall (98%), as well as a strong F1 score (80%). Finally, the ROC curve gives an AUC value of 0.83, which indicates that this KNN model is very reliable in network attack detection, and especially excels in the identification of malicious attacks. The specific confusion matrix and ROC curve of this model are shown in Figure 14. 15.

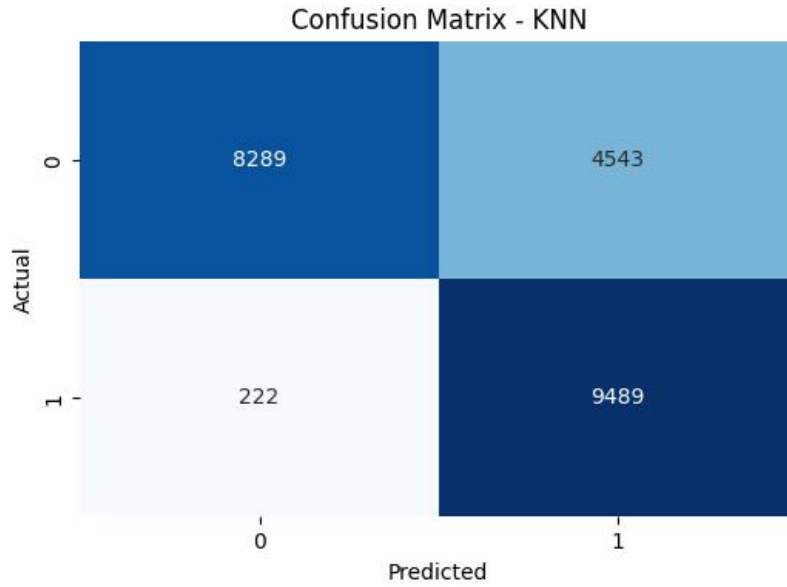


Figure. 14 KNN model binary classification confusion matrix

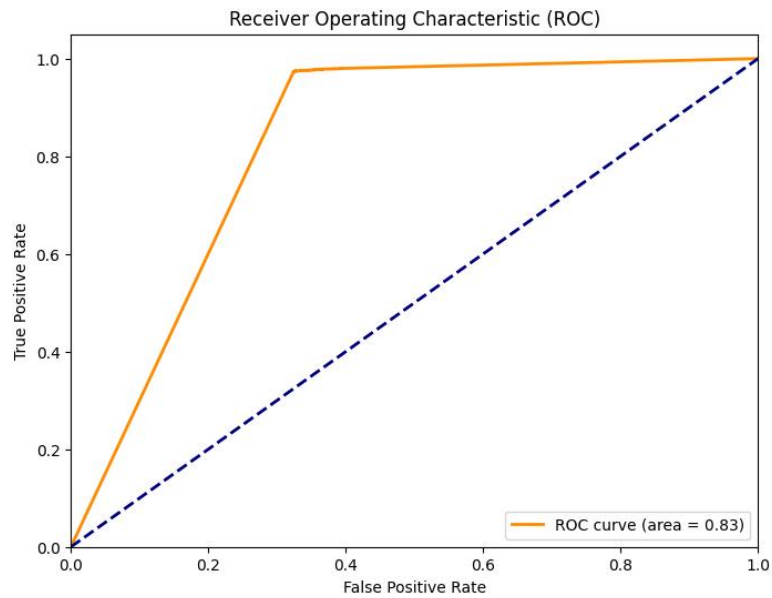


Figure. 15 KNN model ROC-AUC curve

6.1.2.2. DT

The decision tree model also performed well in this project, achieving an accuracy of about 80% on the test dataset after parameter optimisation. The model performed well in detecting cyber-attacks with high precision (70%) and high recall (96%) which in turn achieved a high F1 score (81%). The AUC value of the model is the same as KNN also 0.83. The specific confusion matrix and ROC curve of the model is shown in Figure 16. 17.

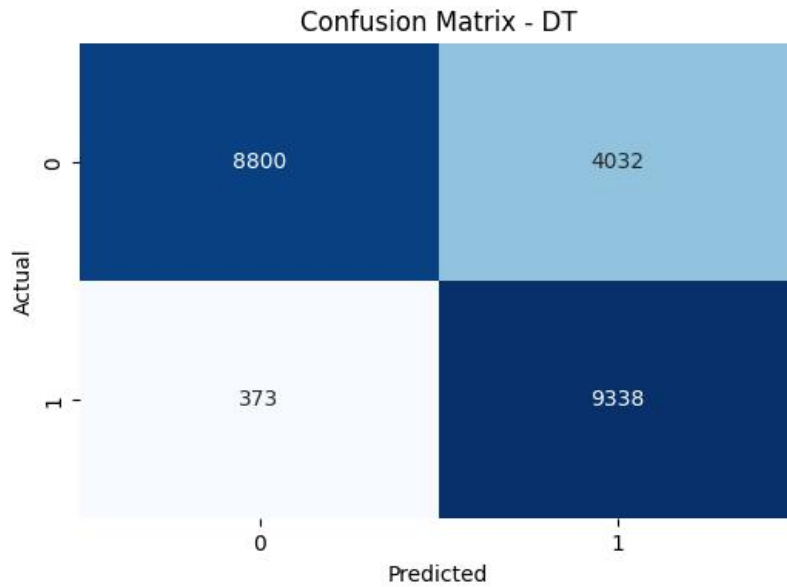


Figure 16 DT Model Binary Classification Confusion Matrix

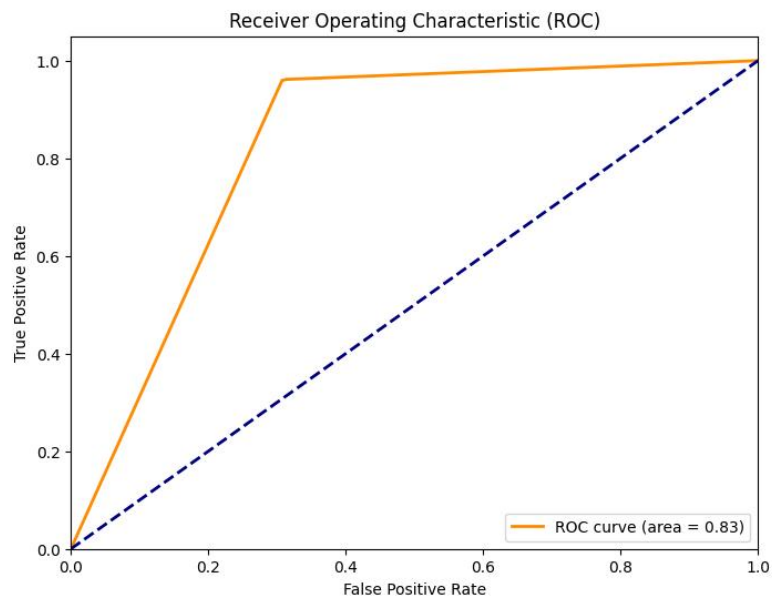


Figure 17 DT model ROC-AUC curve

6.1.2.3. LR

The logistic regression model achieved approximately 75.4% accuracy on the test data. For the positive category, the model demonstrated high recall (93%) but low precision (65%), resulting in an F1 score of 77%. In contrast, for the negative category, the precision was high (93%) but the recall was low (62%), resulting in an F1 score of 74%. Taken together, the model is better at identifying positive categories, but performs slightly less well on negative categories. However, the model has an

AUC value of 0.87, which is better than the KNN and DT models. The specific confusion matrix and ROC curves are shown in Figs. 18, 19.

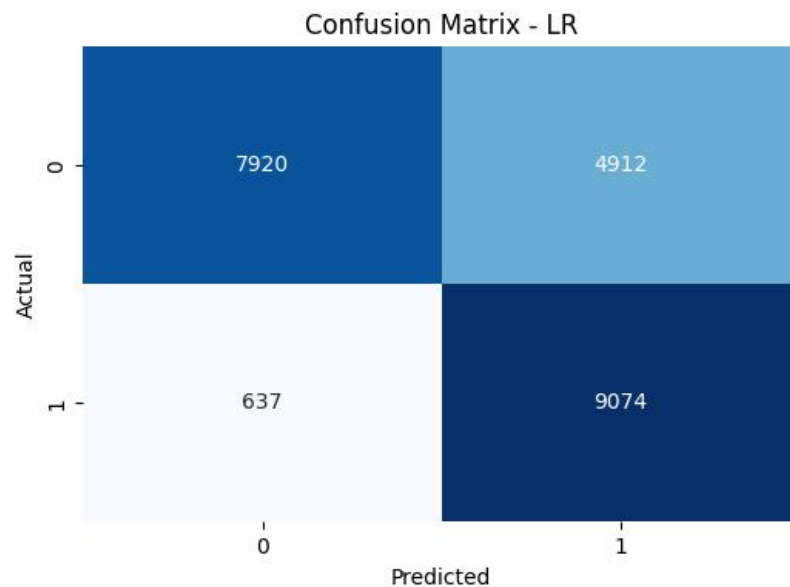


Figure. 18 LR model binary classification confusion matrix

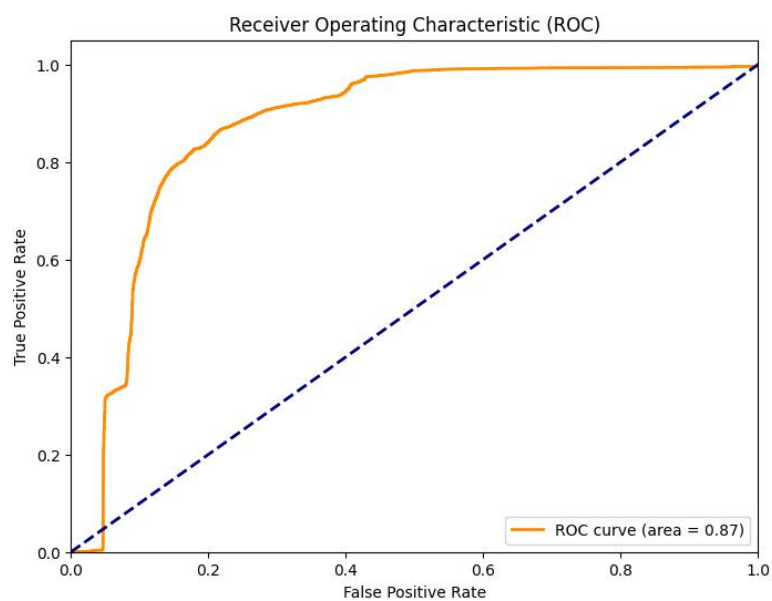


Figure. 19 LR model ROC-AUC curve

6.1.2.4. NN

The neural network model achieved about 80.6% accuracy, which is a bit higher than any of the first three medium models. For the positive category, the model showed high recall (97%) but lower precision (70%), but still obtained a high F1 score of 81%. Correspondingly, for the negative category, the model showed high precision (97%) but relatively low recall (68%), resulting in an F1 score of 80%.

Taken together, the model is better at recognising positive categories, but performs slightly less well on negative categories. It is worth noting that the AUC value of this model is as high as 0.96, which is much higher than that of the previous three models, and the specific confusion matrix and ROC curves are shown in Figs. 20, 21.

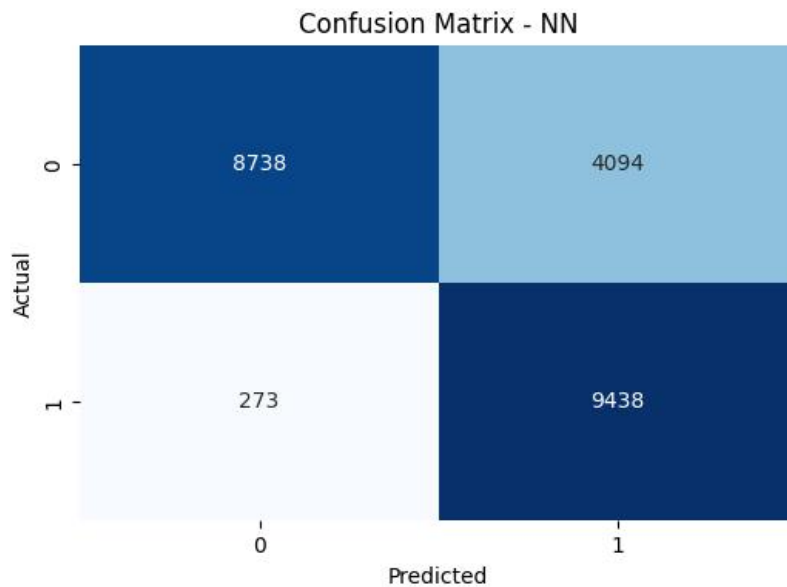


Figure. 20 NN model binary classification confusion matrix

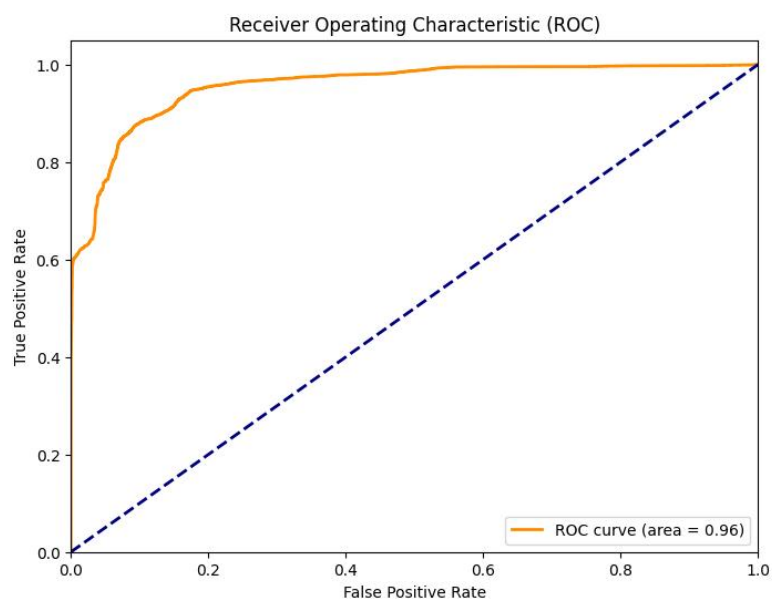


Figure 21NN model ROC-AUC curve

6.1.2.5. RF

The Random Forest model achieved an accuracy of about 77.2%. For the positive category, the model demonstrated high recall (97%) but low precision (66%), resulting in an F1 score of 79%. Correspondingly, for negative categories, the model

showed high precision (97%) but relatively low recall (62%), resulting in an F1 score of 76%. These metrics are worse than the neural network, but a little better than the other models. Random Forest has an AUC of 0.92, again higher than the three previous models, but lower than NN, as shown in Figures 22, 23 below:

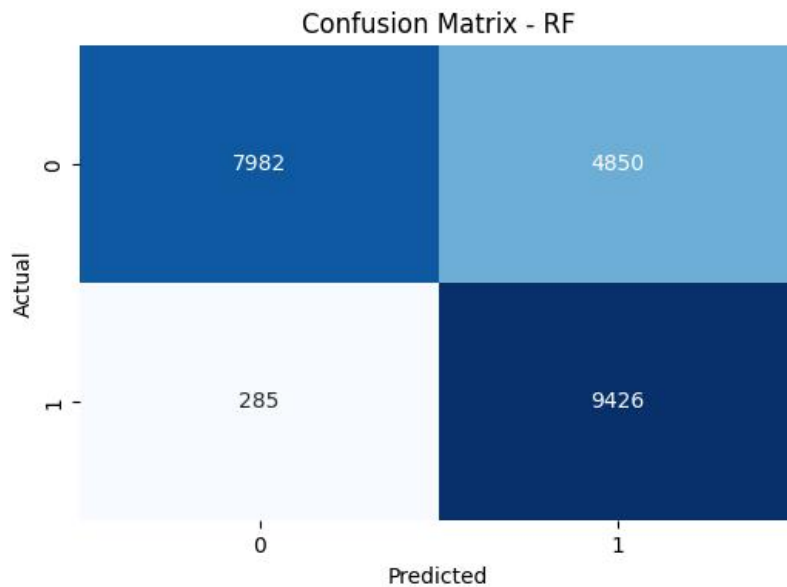


Figure. 22 RF model binary classification confusion matrix

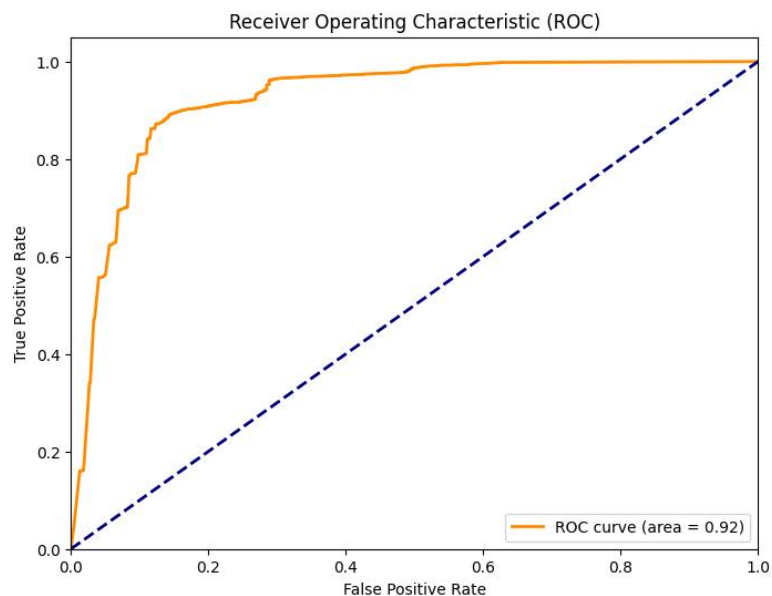


Figure. 23 RF model ROC-AUC curve

6.1.3. Comparison of machine learning models

By summarising the data and images obtained from the experiments and comparing the five different models together, the following conclusions can be drawn:

- The Neural Network model performs best on the test data with the highest accuracy (about 80.6%) and area under the ROC curve (0.96). It has high recall and F1 score for positive categories and is suitable for cyber attack detection.
- The logistic regression model performs relatively poorly in terms of accuracy, achieving only about 75.4%. Despite its relatively high AUC value (0.87), there is a significant trade-off between precision and recall for both positive and negative categories, resulting in a relatively low F1 score.
- Although the neural network model performed well in terms of performance, it was relatively complex and required more computational resources and time for training and tuning. In addition, all models suffer from relatively high recall but low precision for positive categories, which may lead to some false positives. In future work, the models can be further improved to increase their precision and performance balance.

The following Figures 24, 25, 26, and 27 show the Accuracy, Precision, Recall, and F1score of the species models, respectively, and the curves allow for a more intuitive view of the differences between the models.

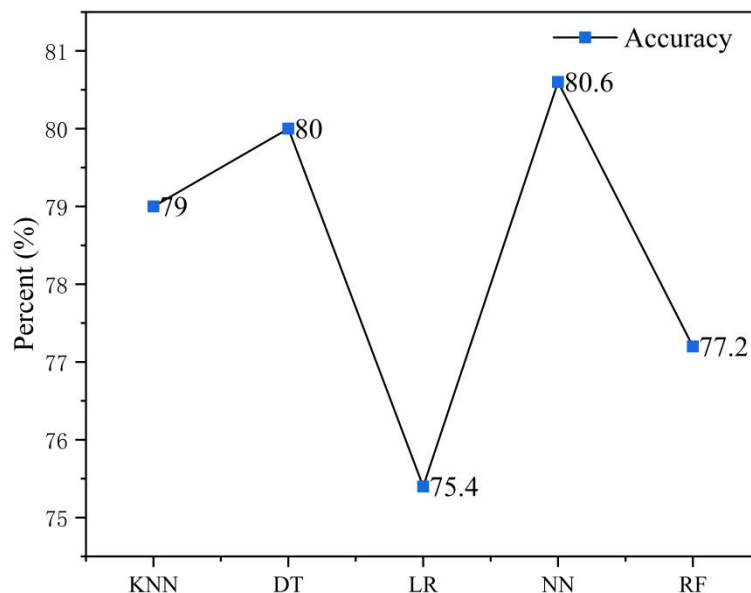


Figure. 24 Accuracy comparison of 5 models

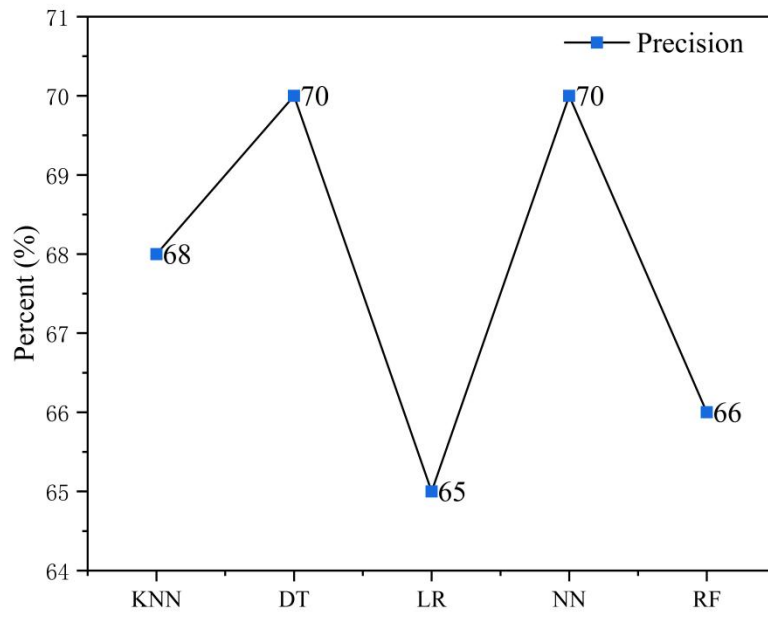


Figure. 25 Precision comparison of 5 models

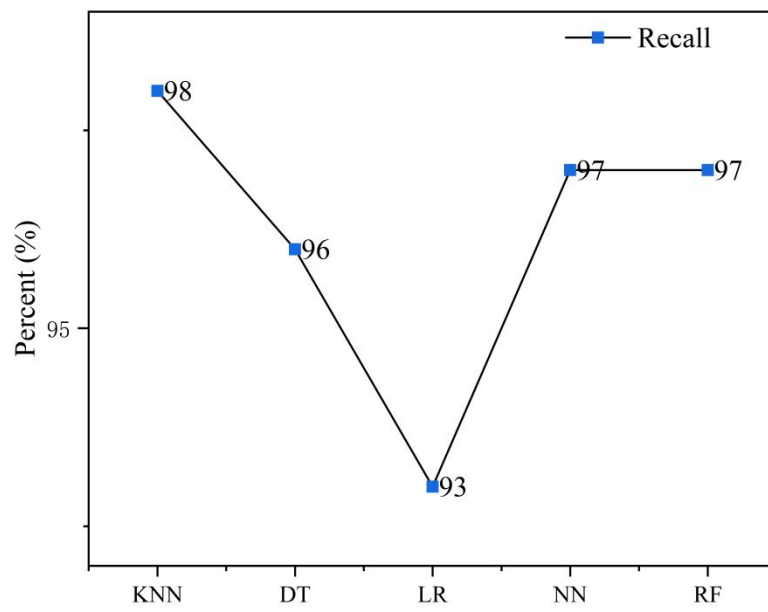


Figure. 26 Recall comparison of 5 models

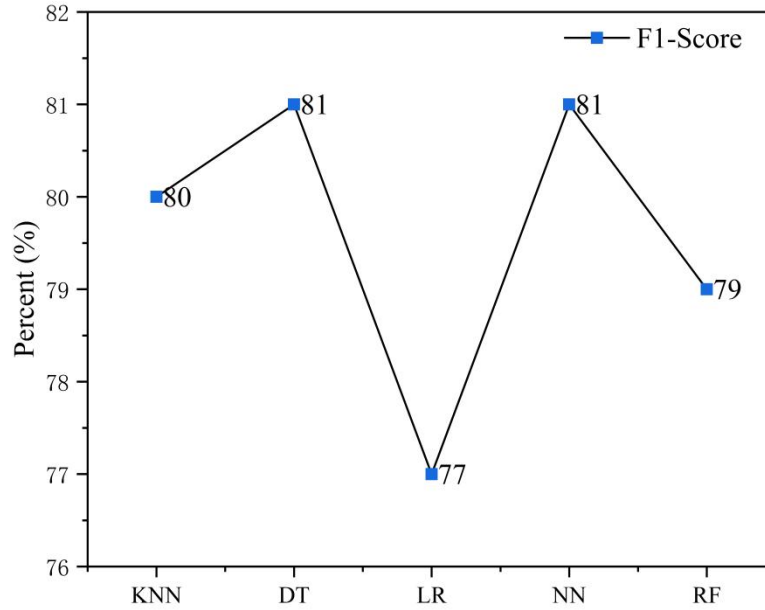


Figure 27 F1-Score Comparison of 5 Models

6.2. Full summary

In recent years, the Internet has developed greatly in terms of scale, scope and complexity, cloud computing, Internet of Things, big data and other new network architectures came into being, with the development of these networks and systems, their security in order to a very important issue. Nowadays, network attacks occur frequently, which can cause serious threats to individuals, families, enterprises and the state, intrusion detection system can detect network attacks and thus protect network infrastructure, has become an essential and important security defence. However, with the diversification of network traffic and attack methods, the traditional intrusion detection model can no longer meet the requirements of the new era. The development of machine learning and deep learning has greatly promoted the improvement of intrusion detection performance, but the current research still suffers from the problems of single structure and insufficient detection effect, etc. In this paper, the following research is carried out to address the above problems:

- Carries out a detailed study of intrusion detection technology. This paper first introduces the common ways of network intrusion attacks, the relevant concepts of intrusion detection technology, development history, algorithmic principles, and the respective characteristics under different classifications. Then, through

researching a large amount of literature, it outlines the current research status of intrusion detection system from relying on traditional feature matching and expert knowledge to the current fusion of machine learning and deep learning technology, and summarises the shortcomings and some challenges faced by the current research in view of the current research status.

- Firstly, the research methodology of the overall experiment is designed based on the gaps purely in the current research, and then five machine learning models based on parameter optimisation are designed and implemented for intrusion detection experiments according to the design and implementation of the five machine learning models, and then the optimal parameters of each model are determined through grid search and cross-validation, and finally the final conclusion is drawn based on the experimental results obtained from the generalisation.
- After generalisation, it was concluded that of the five parameter optimisation-based machine learning models designed in this project in the binary classification task, the neural network model obtained the best accuracy of 80.6% and an AUC score of 0.96, while logistic regression performed the worst with an accuracy of 75.4% and an AUC of 0.83. More over, all the models suffered from a significantly higher performance on the negative category underperformance, which is due to the fact that this dataset is not sufficiently balanced and prone to false positives.

6.3. Recommendations for Further Research

The intrusion detection scheme and model designed in this paper have been verified to have good performance, but there are still some shortcomings, based on the research work in this paper, the following directions are worth further improvement in the follow-up:

- For future research, more complex datasets can be selected for experiments, and more real network data can be obtained through packet-capturing tools, which can make the research more valuable, although the feature processing of the

dataset may be a great challenge, which requires a lot of time as well as collaborative research among researchers.

- The enhances the data preprocessing part by performing more complex feature engineering thus giving better accuracy to the machine learning models. In the research of this project, data transcoding and normalisation is used and in the future, the dataset can be explored more deeply using processing methods such as Dimensionality Reduction, Continuous Attribute Discretisation, One-Hot Coding, etc.
- In this project machine learning models have been selected for experimentation and in future deep learning can be selected for intrusion detection research to try to use latest deep learning techniques. Deep learning technology is changing day by day and can be integrated into the field of intrusion detection by borrowing excellent model structure ideas from other fields such as computer vision to further improve the detection capability.

CHAPTER 7 – REFLECTIONS

The aim of this research was to design a complete set of intrusion detection process, systematically design the research methodology, and design a variety of machine learning models based on parameter optimisation design, and then test each model through NSL-KDD dataset, and arrive at the final conclusions through comparison, and inductive methods. In the overall completion of the project, the specific reflections are as follows:

- In the first chapter, the background of the research is introduced, with data reflecting the surge in the number of network users around the world in recent years, and combined with some actual cases of network security data leakage incidents to illustrate the current importance of preventing network intrusion. Next, the classification of network intrusion detection is briefly introduced, and the feasibility of using machine learning means to combine network intrusion detection. Finally, the research objectives and chapter framework of this project are indicated. In summary, the first chapter, which serves as the introductory chapter, presents why the field of machine learning combined with network intrusion detection should be investigated and states the research objectives.
- In the second chapter, the methodology of this research for searching the literature is firstly presented, and then divided into several subsections systematically reviewing the results of previous studies. Specific REFLECTIONS are as follows, firstly, the development of the intrusion detection field is reviewed, and it is pointed out that the main research tools that are currently popular are combining machine learning and deep learning methods. Then, the concepts of the six machine learning models used in this study are introduced as well as a critical analysis of how researchers have used machine learning models in the intrusion detection field and the research results they have obtained, along with a description of the current datasets commonly used for intrusion detection and a comparison of them. Next, pre-processing methods for datasets that may be used in this research are listed, as well as methods for optimising model parameters.

Finally, research gaps are proposed by summarising the previous research. This chapter achieves a complete and comprehensive review of the aim of the previous research, but the shortcoming is that the preprocessing methods and parameter optimisation take up less space, although in the subsequent chapters these two important variables in the research are repeatedly mentioned again to make up for the previous problems.

- The third chapter is to determine the research methodology of the whole project, through the overview can clearly understand the whole project of each part of the research methodology, complete the research methodology of this aim. there are also shortcomings in the design of the methodology, the methodology is missing the time frame of this link, considering the experiment is to download the dataset from open source websites does not involve questionnaires, and the model is actually adjusted, the running time was sporadic, so this part was ignored.
- Chapter 4 is the design and implementation of the experiment, through the process block diagram can clearly understand the specific process of the whole intrusion detection model, and a detailed description of the data set of the pre-processing process, the model building process and the final results of the output process, to complete the design and implementation of this goal.
- Chapter 5 is the testing and evaluation part of the experiment, in which the implemented parts of the previous chapter are tested one by one, and the parameter optimisation method is written in detail, the intrusion detection binary classification experiment based on the parameter optimisation design is completed, and the optimal parameters are determined, and the aim is completed. however, in the course of this experiment, the SVM model faces the problem of long runtime when dealing with large-scale datasets. Problem. Considering that the NSL-KDD dataset used for this experiment has a large number of features or samples, and SVM needs to consider all the samples in relation to the decision boundary when solving the optimisation problem, while this computational complexity increases with the increase of data volume. Finally, this model is removed.

- Chapter 6 is the chapter that summarises all the previous work, using the evaluation section of Chapter 5 to make a detailed comparison of the five machine learning models and lists the optimal parameters for each model, completing the last aim. finally, an outlook is made for future work, suggesting some possible future attempts.

CHAPTER 8 - REFERENCES

- [1] S. Chen, 'Internet and Beyond: Towards a Better Connected World', *IEEE Internet Comput.*, vol. 24, no. 1, pp. 36–38, Jan. 2020, doi: 10.1109/MIC.2019.2959571.
- [2] J. Clement, 'Data volume of global consumer IP traffic from 2017 to 2022', *Stat. Com Httpswww Stat. Com* accessed Feb 12 2021, 2020.
- [3] L. Cheng, F. Liu, and D. D. Yao, 'Enterprise data breach: causes, challenges, prevention, and future directions: Enterprise data breach', *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 7, no. 5, p. e1211, Sep. 2017, doi: 10.1002/widm.1211.
- [4] Y. Zou, A. H. Mhaidli, A. McCall, and F. Schaub, "'I've Got Nothing to Lose": Consumers' Risk Perceptions and Protective Actions after the Equifax Data Breach', presented at the Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018), 2018, pp. 197–216.
- [5] C. Money, 'Yahoo says 500 million accounts stolen', 2016.
- [6] A. Al-Qerem, B. M. Abutahoun, S. I. Nashwan, S. Shakhathreh, M. Alauthman, and A. Almomani, 'Network-Based Detection of Mirai Botnet Using Machine Learning and Feature Selection Methods', in *Advances in Information Security, Privacy, and Ethics*, B. B. Gupta and D. Gupta, Eds., IGI Global, 2020, pp. 308–318.
- [7] J. Reddy, 'Data Breaches in Healthcare Security Systems', 2021.
- [8] S. Venkatraman and M. Alazab, 'Use of data visualisation for zero-day malware detection', *Secur. Commun. Netw.*, vol. 2018, pp. 1–13, 2018.
- [9] S. M. Kasongo and Y. Sun, 'Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset', *J. Big Data*, vol. 7, pp. 1–20, 2020.
- [10] H. Cavusoglu, S. Raghunathan, and H. Cavusoglu, 'Configuration of and interaction between information security technologies: The case of firewalls and intrusion detection systems', *Inf. Syst. Res.*, vol. 20, no. 2, pp. 198–217, 2009.
- [11] X. Gao, L. Liu, and X. Zhu, 'Research on the main threat and prevention technology of computer network security', presented at the IOP Conference Series: Earth and Environmental Science, IOP Publishing, 2021, p. 052065.
- [12] K. Hickman and T. Elgamal, 'The SSL protocol', 1995.
- [13] H.-J. Liao, C.-H. R. 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, 2013.
- [14] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, 'An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks', *Expert Syst. Appl.*, vol. 29, no. 4, pp. 713–722, 2005.
- [15] C.-Y. Ho, Y.-C. Lai, I.-W. Chen, F.-Y. Wang, and W.-H. Tai, 'Statistical analysis of false positives and false negatives from real traffic with intrusion detection/prevention systems', *IEEE Commun. Mag.*, vol. 50, no. 3, pp. 146–154, 2012.

- [16] M. I. Jordan and T. M. Mitchell, ‘Machine learning: Trends, perspectives, and prospects’, *Science*, vol. 349, no. 6245, pp. 255–260, 2015.
- [17] J. P. Anderson, ‘Computer security threat monitoring and surveillance’, *Tech. Rep. James P Anderson Co.*, 1980.
- [18] D. E. Denning, ‘An Intrusion-Detection Model’, *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987, doi: 10.1109/TSE.1987.232894.
- [19] T. F. Lunt, ‘A survey of intrusion detection techniques’, *Comput. Secur.*, vol. 12, no. 4, pp. 405–418, 1993.
- [20] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, ‘A network security monitor’, Lawrence Livermore National Lab., CA (USA); California Univ., Davis, CA (USA ...), 1989.
- [21] G. Kocher and G. Kumar, ‘Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges’, *Soft Comput.*, vol. 25, no. 15, pp. 9731–9763, Aug. 2021, doi: 10.1007/s00500-021-05893-0.
- [22] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, ‘Intrusion detection by machine learning: A review’, *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, Dec. 2009, doi: 10.1016/j.eswa.2009.05.029.
- [23] W. Lee, S. J. Stolfo, and K. W. Mok, ‘A data mining framework for building intrusion detection models’, presented at the Proceedings of the 1999 IEEE Symposium on Security and Privacy (Cat. No. 99CB36344), IEEE, 1999, pp. 120–132.
- [24] A. L. Buczak and E. Guven, ‘A survey of data mining and machine learning methods for cyber security intrusion detection’, *IEEE Commun. Surv. Tutor.*, vol. 18, no. 2, pp. 1153–1176, 2015.
- [25] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 1999.
- [26] H. Veisi, ‘Introduction to SVM’, in *Learning with Fractional Orthogonal Kernel Classifiers in Support Vector Machines: Theory, Algorithms and Applications*, Springer, 2023, pp. 3–18.
- [27] M. S. Pervez and D. M. Farid, ‘Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs’, in *The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014)*, Dec. 2014, pp. 1–6. doi: 10.1109/SKIMA.2014.7083539.
- [28] J. Gu and S. Lu, ‘An effective intrusion detection approach using SVM with naïve Bayes feature embedding’, *Comput. Secur.*, vol. 103, p. 102158, Apr. 2021, doi: 10.1016/j.cose.2020.102158.
- [29] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, ‘Deep learning approach combining sparse autoencoder with SVM for network intrusion detection’, *Ieee Access*, vol. 6, pp. 52843–52856, 2018.
- [30] P. Tao, Z. Sun, and Z. Sun, ‘An Improved Intrusion Detection Algorithm Based on GA and SVM’, *IEEE Access*, vol. 6, pp. 13624–13631, 2018, doi: 10.1109/ACCESS.2018.2810198.

- [31] Y. Li and L. Guo, 'An active learning based TCM-KNN algorithm for supervised network intrusion detection', *Comput. Secur.*, vol. 26, no. 7–8, pp. 459–467, 2007.
- [32] N. Bhatia, 'Survey of nearest neighbor techniques', *ArXiv Prepr. ArXiv10070085*, 2010.
- [33] 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, pp. 403–443, 2019.
- [34] H. Shapoorifard and P. Shamsinejad, 'Intrusion detection using a novel hybrid method incorporating an improved KNN', *Int J Comput Appl*, vol. 173, no. 1, pp. 5–9, 2017.
- [35] S. Zhang, 'Challenges in KNN Classification', *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 10, pp. 4663–4675, Oct. 2022, doi: 10.1109/TKDE.2021.3049250.
- [36] T. M. Mitchell, 'Machine learning', 1997.
- [37] S. B. Kotsiantis, 'Decision trees: a recent overview', *Artif. Intell. Rev.*, vol. 39, pp. 261–283, 2013.
- [38] K. Rai, M. S. Devi, and A. Guleria, 'Decision tree based algorithm for intrusion detection', *Int. J. Adv. Netw. Appl.*, vol. 7, no. 4, p. 2828, 2016.
- [39] M. Alkasassbeh and M. Almseidin, 'Machine Learning Methods for Network Intrusion Detection'. arXiv, Sep. 01, 2018. Accessed: Aug. 25, 2023. [Online]. Available: <http://arxiv.org/abs/1809.02610>
- [40] J. Liu, W. Yinchai, T. C. Siong, X. Li, L. Zhao, and F. Wei, 'A hybrid interpretable deep structure based on adaptive neuro-fuzzy inference system, decision tree, and K-means for intrusion detection', *Sci. Rep.*, vol. 12, no. 1, p. 20770, Dec. 2022, doi: 10.1038/s41598-022-23765-x.
- [41] C. Azad and V. K. Jha, 'Genetic algorithm to solve the problem of small disjunct in the decision tree based intrusion detection system', *Int. J. Comput. Netw. Inf. Secur.*, vol. 7, no. 8, pp. 56–71, 2015.
- [42] G. Holmes, A. Donkin, and I. H. Witten, 'Weka: A machine learning workbench', presented at the Proceedings of ANZIIS'94-Australian New Zealand Intelligent Information Systems Conference, IEEE, 1994, pp. 357–361.
- [43] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*, vol. 398. John Wiley & Sons, 2013.
- [44] X. Mai, Z. Liao, and R. Couillet, 'A Large Scale Analysis of Logistic Regression: Asymptotic Performance and New Insights', in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 3357–3361. doi: 10.1109/ICASSP.2019.8683376.
- [45] S. Kaddoura, A. E. Arid, and M. Moukhtar, 'Evaluation of Supervised Machine Learning Algorithms for Multi-class Intrusion Detection Systems', in *Proceedings of the Future Technologies Conference (FTC) 2021, Volume 3*, K. Arai, Ed., Cham: Springer International Publishing, 2022, pp. 1–16.
- [46] Y. Wang, 'A multinomial logistic regression modeling approach for anomaly intrusion detection', *Comput. Secur.*, vol. 24, no. 8, pp. 662–674, 2005.

- [47] J. Lawrence, *Introduction to neural networks*. California Scientific Software, 1993.
- [48] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, 'State-of-the-art in artificial neural network applications: A survey', *Heliyon*, vol. 4, no. 11, 2018.
- [49] G. Wang, J. Hao, J. Ma, and L. Huang, 'A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering', *Expert Syst. Appl.*, vol. 37, no. 9, pp. 6225–6232, 2010.
- [50] L. M. Ibrahim, D. T. Basheer, and M. S. Mahmood, 'A comparison study for intrusion database (Kdd99, Nsl-Kdd) based on self organization map (SOM) artificial neural network', *J. Eng. Sci. Technol.*, vol. 8, no. 1, pp. 107–119, 2013.
- [51] M. Govindarajan and R. Chandrasekaran, 'Intrusion detection using neural based hybrid classification methods', *Comput. Netw.*, vol. 55, no. 8, pp. 1662–1671, 2011.
- [52] N. Farnaaz and M. Jabbar, 'Random forest modeling for network intrusion detection system', *Procedia Comput. Sci.*, vol. 89, pp. 213–217, 2016.
- [53] P. Negandhi, Y. Trivedi, and R. Mangrulkar, 'Intrusion Detection System Using Random Forest on the NSL-KDD Dataset', in *Emerging Research in Computing, Information, Communication and Applications*, N. R. Shetty, L. M. Patnaik, H. C. Nagaraj, P. N. Hamsavath, and N. Nalini, Eds., Singapore: Springer Singapore, 2019, pp. 519–531.
- [54] R. K. Vigneswaran, R. Vinayakumar, K. P. Soman, and P. Poornachandran, 'Evaluating Shallow and Deep Neural Networks for Network Intrusion Detection Systems in Cyber Security', in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Jul. 2018, pp. 1–6. doi: 10.1109/ICCCNT.2018.8494096.
- [55] R. P. Lippmann *et al.*, 'Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation', in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, Hilton Head, SC, USA: IEEE Comput. Soc, 1999, pp. 12–26. doi: 10.1109/DISCEX.2000.821506.
- [56] V. Bolon-Canedo, N. Sanchez-Marono, and A. Alonso-Betanzos, 'Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset', *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5947–5957, 2011.
- [57] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, 'A detailed analysis of the KDD CUP 99 data set', presented at the 2009 IEEE symposium on computational intelligence for security and defense applications, Ieee, 2009, pp. 1–6.
- [58] N. Moustafa and J. Slay, 'UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)', presented at the 2015 military communications and information systems conference (MilCIS), IEEE, 2015, pp. 1–6.

- [59] D. Stiawan, M. Y. B. Idris, A. M. Bamhdi, and R. Budiarto, ‘CICIDS-2017 dataset feature analysis with information gain for anomaly detection’, *IEEE Access*, vol. 8, pp. 132911–132921, 2020.
- [60] J. J. Davis and A. J. Clark, ‘Data preprocessing for anomaly based network intrusion detection: A review’, *Comput. Secur.*, vol. 30, no. 6, pp. 353–375, Sep. 2011, doi: 10.1016/j.cose.2011.05.008.
- [61] P. Anukrishna and V. Paul, ‘A review on feature selection for high dimensional data’, presented at the 2017 International Conference on Inventive Systems and Control (ICISC), IEEE, 2017, pp. 1–4.
- [62] W. Xu, J. Jang-Jaccard, A. Singh, Y. Wei, and F. Sabrina, ‘Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset’, *IEEE Access*, vol. 9, pp. 140136–140146, 2021.
- [63] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, ‘Learning from class-imbalanced data: Review of methods and applications’, *Expert Syst. Appl.*, vol. 73, pp. 220–239, 2017.
- [64] E. A. Felix and S. P. Lee, ‘Systematic literature review of preprocessing techniques for imbalanced data’, *IET Softw.*, vol. 13, no. 6, pp. 479–496, Dec. 2019, doi: 10.1049/iet-sen.2018.5193.
- [65] L. N. Smith, ‘A disciplined approach to neural network hyper-parameters: Part 1--learning rate, batch size, momentum, and weight decay’, *ArXiv Prepr. ArXiv180309820*, 2018.
- [66] J. Wainer and P. Fonseca, ‘How to tune the RBF SVM hyperparameters? An empirical evaluation of 18 search algorithms’, *Artif. Intell. Rev.*, vol. 54, no. 6, pp. 4771–4797, Aug. 2021, doi: 10.1007/s10462-021-10011-5.
- [67] F. Pedregosa *et al.*, ‘Scikit-learn: Machine learning in Python’, *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
- [68] L. Yang, A. Moubayed, I. Hamieh, and A. Shami, ‘Tree-Based Intelligent Intrusion Detection System in Internet of Vehicles’, in *2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, HI, USA: IEEE, Dec. 2019, pp. 1–6. doi: 10.1109/GLOBECOM38437.2019.9013892.
- [69] H. Gholamalinezhad and H. Khosravi, ‘Pooling methods in deep neural networks, a review’, *ArXiv Prepr. ArXiv200907485*, 2020.
- [70] R. Genuer, J.-M. Poggi, C. Tuleau-Malot, and N. Villa-Vialaneix, ‘Random Forests for Big Data’, *Big Data Res.*, vol. 9, pp. 28–46, Sep. 2017, doi: 10.1016/j.bdr.2017.07.003.
- [71] L. Dubé and G. Paré, ‘Rigor in information systems positivist case research: current practices, trends, and recommendations’, *MIS Q.*, pp. 597–636, 2003.
- [72] D. Sinha, ‘Phenomenology and positivism’, *Philos. Phenomenol. Res.*, vol. 23, no. 4, pp. 562–577, 1963.
- [73] X. Kang and X. Deng, ‘Design and analysis of computer experiments with quantitative and qualitative inputs: A selective review’, *WIREs Data Min. Knowl. Discov.*, vol. 10, no. 3, May 2020, doi: 10.1002/widm.1358.

- [74] C. Nalmpantis and D. Vrakas, ‘Machine learning approaches for non-intrusive load monitoring: from qualitative to quantitative comparison’, *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 217–243, Jun. 2019, doi: 10.1007/s10462-018-9613-7.
- [75] J. Reichertz, ‘Induction, deduction’, *SAGE Handb. Qual. Data Anal.*, pp. 123–135, 2013.
- [76] A. D’Amato, ‘The Inductive Approach Revisited’, *Indian J. Int. Law*, vol. 6, no. 509, pp. 11–05, 1966.
- [77] M. Artur, ‘Review the performance of the Bernoulli Naïve Bayes Classifier in Intrusion Detection Systems using Recursive Feature Elimination with Cross-validated selection of the best number of features’, *Procedia Comput. Sci.*, vol. 190, pp. 564–570, 2021.
- [78] M. Idhammad, K. Afdel, and M. Belouch, ‘Semi-supervised machine learning approach for DDoS detection’, *Appl. Intell.*, vol. 48, pp. 3193–3208, 2018.
- [79] M. Aledhari, R. Razzak, and R. M. Parizi, ‘Machine learning for network application security: Empirical evaluation and optimization’, *Comput. Electr. Eng.*, vol. 91, p. 107052, 2021.
- [80] G. G. Gable, ‘Integrating case study and survey research methods: an example in information systems’, *Eur. J. Inf. Syst.*, vol. 3, pp. 112–126, 1994.
- [81] S. Choudhury and A. Bhowal, ‘Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection’, presented at the 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), IEEE, 2015, pp. 89–95.
- [82] H. Noble and J. Smith, ‘Issues of validity and reliability in qualitative research’, *Evid. Based Nurs.*, vol. 18, no. 2, pp. 34–35, 2015.
- [83] J. G. Rodriguez, ‘Commentary: Generalisability and validity in qualitative research’, *Br. Med. J.*, vol. 319, no. 7207, pp. 421–421, 1999.
- [84] I. Vizcarguenaga-Aguirre and J.-R. López-Robles, ‘Mono, mixed or multiple strategy approach: a descriptive study of the latest published articles in the International Journal of Project Management’, 2020.
- [85] J. F. Molina-Azorin, ‘Mixed methods research in strategic management: Impact and applications’, *Organ. Res. Methods*, vol. 15, no. 1, pp. 33–56, 2012.
- [86] T. Saito and M. Rehmsmeier, ‘The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets’, *PloS One*, vol. 10, no. 3, p. e0118432, 2015.
- [87] D. H. Deshmukh, T. Ghorpade, and P. Padiya, ‘Improving classification using preprocessing and machine learning algorithms on NSL-KDD dataset’, presented at the 2015 International Conference on Communication, Information & Computing Technology (ICCICT), IEEE, 2015, pp. 1–6.
- [88] B. Han and R. Zhang, ‘Virtual Machine Allocation Strategy Based on Statistical Machine Learning’, *Math. Probl. Eng.*, vol. 2022, pp. 1–6, Jul. 2022, doi: 10.1155/2022/8190296.

- [89] I. F. Kilincer, F. Ertam, and A. Sengur, ‘Machine learning methods for cyber security intrusion detection: Datasets and comparative study’, *Comput. Netw.*, vol. 188, p. 107840, Apr. 2021, doi: 10.1016/j.comnet.2021.107840.
- [90] S. K. Sahu, S. Sarangi, and S. K. Jena, ‘A detail analysis on intrusion detection datasets’, presented at the 2014 ieee international advance computing conference (IACC), IEEE, 2014, pp. 1348–1353.
- [91] A. Abdelkhalek and M. Mashaly, ‘Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning’, *J. Supercomput.*, pp. 1–34, 2023.
- [92] N. Japkowicz and M. Shah, *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.