National College of Ireland

Automatic Intrusion Detection System Using Deep Re-Enforcement Learning With Q-network Algorithm (DQN)

MSc Research Project
Cyber Security

Ugochukwu Nwokedi
Student ID: x21235287

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. …UGOCHUKWU NWOKEDI IKENNA……

Student ID: ……X21235287…… …….……

Programme: ……MSC CYBER SECURITY…… … **Year:** ……2022/23…….

Module: ……ACADEMIC INTERNSHIP…… ……

Supervisor: ……MICHAEL PRIOR……

Submission Due Date: ……14TH AUGUST 2023……

Project Title: ……AUTOMATIC INTRUSION DETECTION USING DEEP REINFORCEMENT LEARNING WITH Q NETWORK ALGORITHM…… ……

Word Count: ……11371…… **Page Count**……28……..……..

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

Signature: ……UGOCHUKWU NWOKEDI……

Date: ……12TH AUGUST 2023……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□

You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

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

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

Automatic Intrusion Detection System Using Deep Re-Enforcement Learning With Q-network Algorithm (DQN)

Ugochukwu Nwokedi
X21235287

Abstract

IDS is an important part of the cybersecurity and IT space. Network intrusions are most times very hard to detect and group into their right network patterns. Due to the complexity of network intrusion detection, new age machine learning models that could eliminate some problems that an IDS places on a machine learning model e.g. poorly constructed, non-uniform datasets have to be discovered. Reinforcement learning has been applied to varying successful degrees in automation, gaming etc., and this report shows that it can be successfully applied to the field of network intrusion detection as well. Our proposed model uses the algorithm of a Q-network to operate in a reinforcement learning environment. The approach will be evaluated using a reward against episodes learning curve and will be further compared to more traditional machine learning models that use accuracy, recall and precision as evaluation metrics.

1 Introduction

In today's fast-changing cybersecurity world, safeguarding computer networks and systems from unauthorized access is extremely important. Intrusion Detection Systems (IDS) are crucial tools for finding and reacting to potential security breaches. These systems do this by watching and studying network activity or host behaviour to find and address possible breaches or security problems (Bajtoš, Sokol and Mézešová, 2019).

IDS can be mainly divided into two key types: network-based IDS (NIDS) and host-based IDS (HIDS) (Suroso and Prastya, 2020). NIDS observe network activity at key spots, such as network entry points or switches, checking data for dubious patterns or familiar attack indicators. In contrast, HIDS focus on individual systems, observing system records, file integrity, and user actions to find potential intrusions. Consequently, merging NIDS and HIDS provides a more complete and layered approach to finding intrusions, allowing groups to find and address dangers on various levels. Although IDS have shown their value in cybersecurity, they still have methods that have many natural issues and restrictions. For example, signature-based detection, often used by IDS, depends on known attack patterns (Alarqan, Zaaba and Almomani, 2019). This method is, therefore, inadequate against new and unknown exploits, which may lack recognizable patterns. Furthermore, rule-based detection is inflexible and needs constant updates to remain current with evolving attack tactics (Gupta and Agrawal,

2020). Anomaly detection methods frequently result in many false alarms, making it hard to differentiate between legal and harmful actions (Sadikin, van Deursen and Kumar, 2020). These restrictions emphasize the need for improvements in IDS solutions.

1.1 Research Problem

The research problem in this study revolves around the advancement of Intrusion Detection Systems (IDS) by comparing the effectiveness of reinforcement learning and supervised machine learning algorithms. The aim is to explore the potential of machine learning techniques for improving the accuracy, adaptability, and efficiency of intrusion detection. As cyber-attacks become increasingly sophisticated and dynamic, traditional IDS approaches face challenges in effectively identifying and responding to evolving intrusion techniques (Hossain et al., 2020). Furthermore, the research problem is highly relevant in the current cybersecurity landscape, where organizations and individuals face constant threats of data breaches, malware infections, and unauthorized access attempts. The findings of this study can provide valuable insights into the effectiveness of current machine learning techniques for intrusion detection, enabling practitioners and researchers to make informed decisions regarding the selection and implementation of IDS techniques.

1.2 Research Aim and Objectives

The aim of this project is to create an Intrusion Detection System (IDS) using the DQN algorithm. This IDS aims to accurately identify and categorize network intrusions. It should have the ability to learn from past network data and make educated judgments about whether network activity is typical or harmful. This includes identifying different types of malicious behaviour such as Probing, Denial-of-Service (DOS) attacks, Unauthorized Access (R2L), and User-to-Root (U2R) attacks.

1.3 Research Questions

This study will explore the following research questions:

1. How effectively can the Multi-Layer Perceptron model, a range of supervised machine learning algorithms, including random forests, support vector machines (SVM), K-nearest neighbor (KNN), and Gradient Boosting Machines (GBM), perform multi-class classification of network intrusion attacks, with a specific emphasis on accurately categorizing attack types into DoS, Probe, U2R, and R2L groups?
2. What is the comparative effectiveness of machine learning models and deep learning model, assessed using accuracy, precision, recall, and F1-score evaluation metrics, in classifying network intrusion attacks?
3. How can an IDS be developed using DQN algorithm for accurately classifying various network intrusions by learning from historical data and distinguishing between normal and malicious behaviours, encompassing Probing, DOS, R2L, and U2R attacks?

1.4 Contribution of the Research

This study explores how reinforcement learning and supervised machine learning algorithms work, giving useful insights into their strengths, weaknesses, and usefulness for making intrusion detection more accurate and efficient. The research also explores a deep learning application and compares the different techniques to find ways to identify intrusions better. Using reinforcement learning algorithms, the research checks if IDS can learn and change their ways of finding problems as new attacks happen. The findings of the study can help make IDS solutions that are good at handling new threats and attacks.

1.5 Structure of the Study

This report contains the introductory section which introduces the research problem, its significance, and objectives. The related works section reviews existing literature on intrusion detection systems, highlighting current knowledge and research gaps. The methodology section includes the general implementation methodology, data collection and experimental setup. The design specification section identifies techniques, architecture, and requirements forming the implementation's foundation. The implementation section covers executing the proposed solution, describing outputs like transformed data and models, along with tools used. The evaluation section analyses findings, considering academic and practical perspectives, and focusing on key results. The conclusion and future work section summarizes the research, highlighting key findings, answering objectives, and suggesting future studies.

2 Related Work

IDS have undergone significant transformation as a result of technological advances in artificial intelligence and machine learning techniques. Nevertheless, using reinforcement learning and supervised machine learning algorithms, two well-known IDS techniques, are explored and contrasted in this research review.

2.1 Intrusion Detection Systems (IDS)

The increasing advancements in computer networks and technologies have led to growing concerns about security. Cyber-attacks targeting networks have become a common occurrence, making it crucial to find effective solutions to protect against these threats (Papanikolaou et al., 2023). Using Intrusion Detection Systems (IDS) and other instruments to identify and stop hostile activity in networks is one viable strategy (Talaei Khoei and Kaabouch, 2023). According to Azizan et al. (2021) an intrusion detection system (IDS) is designed to analyse internet traffic and stop illegal or harmful activity. When suspicious or malicious activity is discovered, certain IDSs have the ability to take action, such as restricting traffic from suspect IP addresses. Network assaults have significantly increased in frequency and severity over the past few years and have become more intricate with time. In recent years, there has been a significant increase in the number and severity of network attacks, which have become more complex over time (Qiu et al., 2020). As a result, there has been extensive research on various

security techniques to defend against these cyber-attacks and computer viruses over the past decade. Furthermore, IDS continuously monitors the network and identifies any suspicious or policy-violating behavior. This is especially important in systems like smart grids, where IDS can prevent unauthorized access and exploitation of vulnerabilities.

The three primary categories of intrusion detection systems (IDS) are signature-based, specification-based, and anomaly-based (Ahmad et al., 2020). Cyberattacks are discovered by signature-based IDS by examining patterns of harmful behaviour. IDS that is based on specifications notifies users when activity deviates from expected behavior. Statistical techniques are used by anomaly-based IDS to discriminate between malicious and legitimate behaviour. IDS plays a key role in distinguishing between system intrusions and malfunctions by examining data gathered from the system and detecting any modifications that take place after an attempted attack (Heidari and Jabraeil Jamali, 2022). The choice of algorithm used in IDS is a significant factor that impacts its performance, and determining the best-performing algorithm requires further investigation (Alzahrani and Alenazi, 2021). Nevertheless, given the adaptability of networks, ensuring their security has become crucial. Attackers are capable of controlling the computer system using strong adaptive techniques while interfering with network traffic. In order to prevent network breaches and identify the type of intrusion that has been attempted, it is crucial to do so.

2.2 Machine Learning and Deep Learning

Machine Learning describes computer-based techniques that simulate human learning processes to autonomously acquire knowledge. This encompasses a wide range of disciplines, such as psychology, neuroscience, computer science, and statistics, that are combined to explore and study different aspects (Xu et al., 2021). In recent times, learning algorithms have made significant advancements, largely due to the increased processing power of computers and the availability of big data. According to Shin et al. (2019), machine learning is usually split into three groups of algorithms: supervised learning , unsupervised learning, and reinforcement learning. Supervised learning algorithms involve training models by mapping true output labels to learn the relationship with corresponding feature values (Katole, Sherekar and Thakare, 2018). In contrast, unsupervised learning algorithms learn from the entire training dataset without knowing the output for each input. They do not rely on labeled data. An example of an unsupervised learning algorithm is K-means clustering (Ravipati and Abualkibash, 2019). Further, Reinforcement learning (RL) algorithms focus on learning from the environment by placing an agent within it (Géron, 2019). The agent learns from actions within the environment, experiencing both successes and failures.

Furthermore, in supervised machine learning, a learning algorithm is used to optimize predictions by tuning various parameters based on a dataset of recorded samples (Lopez-Martin, Carro and Sanchez-Esguevillas, 2020). These samples consist of feature vectors and corresponding pre-assigned labels. The algorithm's goal is to accurately predict the labels for new feature vectors. In comparison, there are two essential elements in a conventional Deep Reinforcement Learning (DRL) framework: a representative as well as a setting (Hu, Beuran, and Tan, 2020). The agent makes decisions based on observations of the surroundings. As a consequence of certain actions, the environment undergoes changes, leading to a new state and

a reward that represents the effectiveness of the action. In a conventional reinforcement learning framework, there is no predefined dataset available that pairs examples with their respective actions. Rather, the framework learns the connection between states and actions by means of a reward function (Kim et al., 2017). Learning is not based on immediate information about the best actions, but on a series of rewards that indicate the value of intermediate actions, with the aim of maximizing the total rewards obtained. Machine Learning-based Intrusion Detection Systems (IDS) offer an autonomous learning solution and demonstrate improved performance compared to conventional IDS (Sethi et al., 2019). Due to the conflicting criteria in predicting IDS performance, the algorithms used are becoming increasingly intricate and thus, less transparent.

Reinforcement learning stands apart from supervised and unsupervised learning as it revolves around the concept of sequential decision-making. This approach involves the interaction between an agent and an environment across discrete time steps. The agent, driven by goals, functions as a learning algorithm (Subramanian, Chitlangia and Baths, 2022). The environment encompasses everything beyond the agent's control, and it is not necessarily limited to physical boundaries, such as those between a robot and its surroundings. In fact, the agent can even be a component within a larger control system. Typically, the agent possesses limited knowledge regarding the environment and aims to acquire the optimal behavioural strategy through learning. First off, the RL's objective is to maximize reward rather than forecast the output data Y based on the input data X. Second, the learner's goal is to select the values of X; there is no set distribution D(x) from each point X (Barto and Dietterich, 2004). Below is a diagram illustrating the main steps of reinforcement learning.

2.3 Review of Related Studies

The growing prevalence of big data has led to widespread adoption of machine learning techniques in intrusion detection systems. Some studies have utilized traditional machine learning algorithms or their improved versions for this purpose. Qureshi et al., (2020) introduced a unique adversarial intrusion detection system called RNN-ADV, which is based on random neural networks. However, the effectiveness of this model is significantly impacted by the perturbation environment. When the JSMA algorithm is utilized, RNN-ADV surpasses deep neural networks in terms of accuracy and F1-score. Verma and Ranga, (2019) conducted a study on classifier ensembles, evaluating multiple models on three datasets: CIDDS-001, UNSW-NB15, and NSL-KDD. They performed 10-fold cross-validation and found that the Classification and Regression Trees algorithm achieved the highest average accuracy of 96.74% by creating decision trees. However, XGBoost achieved a very close accuracy of 96.73% and achieved the best average True Positive Rate (TPR) of 97.31%.

In a study conducted by Azizan et al. in 2021, the main objective was to determine the classification algorithm that offers optimal detection performance for intrusion in IDSs. The study provides an evaluation of three commonly used classification algorithms, namely random forest (RF), decision jungle (DJ), and support vector machine (SVM), in the context of intrusion detection systems (IDS). The Intrusion Detection Evaluation Dataset (CIC-IDS2017) and Knowledge Discovery in Databases approach are used to implement and test the ML-based NIDS model. SVM achieves 98.18% accuracy on average, followed by RF at 96.76% and DJ

at 96.50%, according to the data. SVM's, RF's, and DJ's average precisions are 98.74, 97.96, and 97.82 respectively. SVM scores 95.63, RF scores 97.62, and DJ scores 95.77 for average recall.. Based on these results, SVM demonstrates the best overall performance for detecting intrusions in IDSs. Furthermore, Chen et al.(2023) proposed FGS-RFRAS, a feature grouping and selection strategy that incorporates a robust fuzzy rough approximation space and graph theory. The effectiveness of this approach was demonstrated through evaluations on 21 datasets, highlighting its ability to improve the model's robustness. Feature selection techniques fall into three categories, namely filtering, embedding, and wrapper. Among these techniques, the wrapper method RFE is deemed more appropriate for IDS datasets that contain extensive data and numerous features because it iteratively selects subsets of features.

In their study, Chowdhury, Ferens, and Ferens (2016) proposed a hybrid approach that combines two machine learning algorithms to classify abnormal behavior in network traffic. The performance of their model was assessed using metrics such as false positive rate, false negative rate, and the time required for intrusion detection. The experimental findings showcased a high detection rate and accuracy of 98.76%, along with a low false-positive rate of 0.09% and false-negative rate of 1.15%. In comparison, the conventional SVM-based scheme achieved a detection accuracy of 88.03%, a false-positive rate of 4.2%, and a false-negative rate of 7.77%.

In their study, Mohamed, Hefny, and Alsawy (2018) proposed a methodology for classifying network activities as normal or abnormal using machine learning algorithms such as Random Forest (RF), Multi-Layer Perceptron (MLP), and Library for Support Vector Machine (LIBSVM). They conducted experiments on the NSL-KDD dataset and employed the Correlation Feature Selection (CFS) technique to eliminate irrelevant attributes. The results revealed that the multilayer perceptron classifier achieved an accuracy of 95.7%. On the other hand, the Random Forest algorithm exhibited an accuracy of 99.6%, while the LIBSVM classifier achieved 94.8% accuracy. The CFS feature selection approach showed a lower accuracy of 91.7%. However, the overall accuracy improved to 97.2% when incorporating LIBSVM into the methodology.

Biswas (2018) proposed an intrusion detection approach that incorporates machine learning and other feature selection techniques to identify important features from the original dataset. By analysing various classifiers and selection methods, they aimed to improve the accuracy of intrusion detection. The performance of the approach was evaluated using a five-fold cross-validation technique on the "NSL-KDD" dataset. The experimental results showed two main findings: (1) the "K-NN" classifier outperformed other methods in terms of performance and efficiency, and (2) the feature selection method based on information gain ratio yielded better results.

3 Research Methodology

The research methodology presented here describes the process of conducting multi-class classification for network intrusion using the NSL_dataset. This project was implemented with Google Colaboratory, a cloud platform that is built on Jupyter Notebooks (Carneiro et al., 2018). It offers a runtime that is completely set up for machine learning and deep learning with

free access to a powerful GPU. The programming language adopted for this study is the Python language. Python is a sturdy programming language that is easy grasp. Its object-oriented programming technique is simple but efficient, and contains sets of high-level data structures (Raschka, Patterson and Nolet, 2020).

3.1 General Classification Approach

Classification is a widely used technique that involves predicting the class of new samples based on a model derived from training data. The general approach is shown in Figure 3.1 below. According to Han, Pei and Tong (2022), the classification process consists of two steps. The first step involves constructing a classification model from a given dataset, referred to as the training set. The second step involves using the model to predict or test the class labels for new, unseen data, which is known as the testing set.

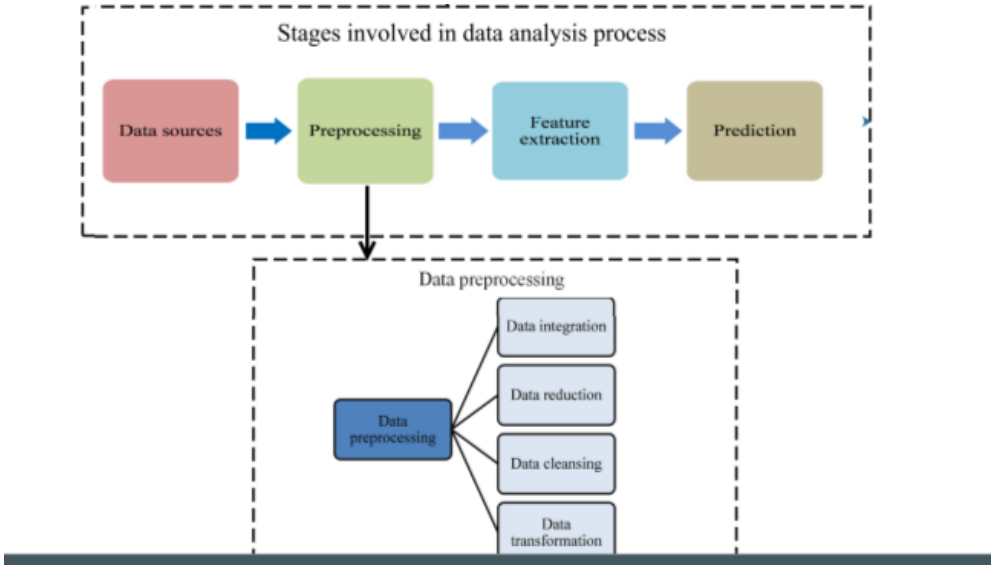


Figure 3.1: General classification approach
Source: (Alghamdi and Javaid, 2022) is a widely

3.1.1 Data Preprocessing

Data preprocessing is an essential initial step where the data is transformed or encoded to enable efficient parsing by the machine (Ramírez-Gallego et al., 2017). This prepares the data for analysis by the model algorithm, allowing for quick and effective interpretation of its features. Preprocessing is done to transform raw data into a format that can be used by artificial intelligence. A data scientist can achieve more accurate results by using structured and clean data. It is possible to use various techniques to complete each step required to achieve accurate automatic prediction. Above in Figure is a schematic that shows the preprocessing steps.

3.1.2 Data Cleaning

Data cleaning is the process of identifying inaccurate or noisy data and either fixing them or eliminating them from the dataset. It primarily focuses on locating and replacing records and data that are erroneous, irrelevant, incomplete, or otherwise noise-related (Ridzuan and Wan Zainon, 2019). The initial step in data cleansing involves analyzing the data to identify errors and inconsistencies present in the dataset.

3.1.3 Data Transformation

The transformation process involves modifying and combining data to create structured configurations that are suitable for knowledge discovery (Maingi, 2015). Data transformation can also be utilized to convert categorical variables into numerical ones, which is advantageous for the development of prediction models (Fan et al., 2021). The widely adopted approach for this transformation is the one-hot encoding method. This method generates a matrix with L − 1 columns for a categorical variable containing L levels (Fan et al., 2019). However, a potential drawback of this approach is that it can lead to high-dimensional data if the categorical variables have numerous levels. To address this issue, deep learning algorithms can be employed. These algorithms represent categorical variables using dense representations, proving especially useful for text data where individual words are represented as vectors for further analysis (Goodfellow, Bengio and Courville, 2016). According to Han, Pei and Tong, (2022), the transformation process achieves its objectives through several sub-processes: smoothing, aggregation, generalisation, data normalisation and feature construction. Some of these processes are used, described in subsequent sections.

3.1.4 Data Integration

Data integration involves combining data from different sources into a unified dataset. With the increase in data volume and complexity, data integration becomes a challenging and iterative process (Nargesian, Asudeh and Jagadish, 2022). This process includes addressing heterogeneity and maintaining data standards across various sources.

3.1.5 Data Reduction

Data reduction is a technique that addresses these issues by reducing the size or dimensionality of a dataset (Ramírez-Gallego et al., 2015). This helps enhance the efficiency of the model's learning process and improves performance by preventing overfitting and addressing skewed data distribution. Large amounts of data in a data warehouse or dataset can create challenges for storage and processing. However, not every model requires a vast amount of data for training purposes. The data may contain numerous attributes, some of which may be irrelevant or interconnected (Ares, Morán-Fernández and Bolón-Canedo, 2022). The data reduction process aims to obtain a smaller representation of the dataset while preserving the integrity of the original data.

3.2 Evaluation metrics

To evaluate the efficiency of our classification models, the following performance metrics are employed.

Precision: Precision is a measure of how accurately the classifier made accurate predictions. It displays the percentage of accurately identified positive occurrences (also known as true positives) in comparison to all positive predictions, including both true and false positives. The formula for calculating precision is TP/(TP + FP).

Recall (Sensitivity): Recall, also referred to as sensitivity or the true positive rate, measures how well a model can pick out positive events from all real positive instances. Where TP stands for true positives and FN for false negatives, the calculation is TP / (TP + FN).
F1-score: The F1-score, which aims to establish a balance between the two measures, is a harmonic mean of precision and recall. It offers a solitary indicator of how well the model performs overall in accurately identifying situations. As 2 * (Precision*Recall) / (Precision+Recall), the F1- score is determined.

4 Design Specification

This section will discuss the framework of the different machine algorithms used for the model training which include, random forests, support vector machines (SVM), K-nearest neighbor (KNN), Gradient Boosting Machines (GBM), neural networks and reinforcement learning.

4.1 Random Forests

In order to effectively analyze and leverage the vast amount of data available today, it is essential to utilize learning algorithms that can scale with the size of the dataset while maintaining statistical efficiency. Random forests, introduced by L. Breiman in the early 2000s, have emerged as one of the most effective methods for handling large data sets (Esteve et al., 2023). Random Forest (RF) is an ensemble learning technique that involves constructing multiple decision trees during the training phase and combining their outputs to make final predictions, such as the mean value in regression problems (Breiman, 2001). The training process of RF incorporates a dual randomization approach: random sampling of data through bootstrapping and random selection of predictors (Lorenzen, Igel and Seldin, 2019). Although RF is widely used in practice, its theoretical understanding remains limited (Arlot and Genuer, 2014; Denil, Matheson and Freitas, 2014). This is mainly due to the complexity of analyzing the dependencies between the induced partitions of the input space and the predictions within those partitions (Arlot and Genuer, 2014). To address this issue, a variation called purely random forests was introduced by Breiman (2002). Purely random forests avoid dependencies by creating random partitions that are independent of the training data, achieved through random selection of features and splits.

4.2 Support Vector Machines

Support Vector Machines (SVMs), initially proposed by Vladimir Vapnik in the field of statistical learning theory and structural risk minimization, have been widely successful in various classification and forecasting tasks (Vapnik, 1998). SVMs have found applications in pattern recognition, regression estimation, dependency estimation, forecasting, and the development of intelligent machines. One key advantage of SVMs is their ability to handle high-dimensional feature spaces, thanks to the generalization principle rooted in the Structural Risk Minimization Theory (Ekici, 2012). This principle ensures that the algorithm is grounded in robust statistical learning theory, providing guaranteed bounds on the risk associated with

the learning process (Nayak, Naik and Behera, 2015). Support Vector Machines (SVM) were initially developed for classification tasks but have been extended to solve regression problems as well (Roy and Chakraborty, 2023). The regression version of SVM, known as Support Vector Regression (SVR), has demonstrated superior performance by addressing the overfitting issue commonly encountered in regression tasks and improving the accuracy of response approximation.

4.3 K-nearest Neighbour

The K-Nearest Neighbors (KNN) algorithm is a non-parametric classification approach that does not make any assumptions about the underlying dataset. It is well-known for its simplicity and effectiveness in solving classification problems (Taunk et al., 2019). The development of K-nearest neighbour classification stemmed from the necessity to perform discriminant analysis in situations where accurate parametric estimations of probability densities are either unknown or challenging to determine (Peterson, 2009). KNN is a supervised learning algorithm that requires a labelled training dataset, where data points are assigned to different classes. Using this training data, KNN predicts the class of unlabelled data points. The classification is determined based on various characteristics of the data. KNN is commonly used as a classifier, classifying data by identifying the nearest or closest training examples in a given region (Suyal and Goyal, 2022). This approach is preferred due to its simplicity and computational efficiency. For continuous data, KNN utilizes the Euclidean distance metric to calculate the proximity to its nearest neighbours.

4.4 Gradient Boosting Machines

The Gradient Boosting Machine (GBM) is a highly effective supervised learning algorithm that combines multiple weak learners to create an ensemble model with exceptional predictive performance (Lu and Mazumder, 2018). It demonstrates impressive results in various prediction tasks such as spam filtering, online advertising, fraud and anomaly detection, computational physics (e.g., the discovery of the Higgs Boson), and more. GBM is often ranked as a top algorithm in competitions like Kaggle and the KDD Cup. One notable feature of GBM is its ability to handle heterogeneous datasets, including highly correlated data, missing data, and unnormalized data. GBM produces interpretable models by constructing an additive model. The goal of gradient boosting is to repair mistakes caused by previous decision trees by learning a new decision tree at each level (Klug et al., 2019). When higher-order relationships are present in the data, the non-linear method naturally outperforms linear models (Hong, Haimovich and Taylor, 2018). However, in a variety of data problems, gradient boosting has outperformed other machine learning techniques (Eloudi et al., 2023).

4.5 Neural Networks

Over the years, there have been various popularity peaks for neural network (NN) models. The first one got underway with the invention of the perceptron (Rosenblatt 1958) and its training algorithm for classification in linearly separable situations. By the beginning of the 1970s, limitations highlighted by Minsky and Papert (1969) dampened interest in these models. The subsequent phase of success was accompanied by the appearance of findings that presented NNs as universal approximators (Cybenko 1989). By the early 2000s, however, new paradigms

like support vector machines and technical difficulties had virtually reached a deadlock. Neural networks offer a powerful learning framework that is highly suitable for natural language tasks (Gallego and Insua, 2021). A crucial component in language-related neural networks is the utilization of an embedding layer, which converts discrete symbols, like words, into continuous vectors within a lower-dimensional space (Goldberg, 2017). They go from being isolated symbols to manipulable mathematical entities when words are inserted. This enables the comparison of vectors, thereby simplifying the generalization process from one word to another. The network learns this representation of words as vectors during the training process. Moreover, as the network progresses, it also learns how to combine these word vectors effectively for prediction purposes. This capability helps address the challenges posed by data sparsity and discreteness to a certain extent.

5 Implementation

This section will discuss the stages and steps taken to implement the various machine learning models. The goal of the research is to improve the security protocols for computer networks by classifying network intrusion attempts into multiple categories. The main data source used is the NSL_KDD dataset, which has 44 features. The major goal is to organize the attack types into four categories—DoS, Probe, U2R, and R2L—to provide an effective method for recognizing and minimizing potential threats.

5.1 Data Preprocessing

The Google Collaboratory was used to carry out this project. Python and other programming languages are available through Google Colab, a web-based cloud service. For this study, python was the programming language of choice. Data preprocessing is the first step is the implementation of the data. It is the process of changing raw data into information that the machine learning model can work with. It is the primary and most crucial step when carrying out the development of a new model. The data set is loaded and will enable the preprocessing steps to be carried out.

5.2 Data Cleaning

Data cleaning is the procedure used to locate and fix mistakes in the data warehouse. As data is gathered from numerous origins into a data warehouse, guaranteeing quality and consistency across that data becomes a substantial and difficult undertaking. Data cleaning provides essential services for cleaning data, including attribute selection, token construction, algorithm selection for grouping, function selection for similarity, function selection for removal, function selection for merging, etc (Ridzuan and Wan Zainon, 2019). Further, specific data cleaning activities are carried out to guarantee data quality and consistency. Columns with zero values, like "difficulty level" and "num_outbound_cmds," are removed from the dataset because they don't add valuable data. By doing this, the dataset's integrity is guaranteed, and it is also ready for further preparation.

5.3 Data Identification

Data labeling is the process of detecting unlabeled raw data (such as pictures, text files, videos, etc.) and adding one or more useful labels to give it context so that a machine learning model may learn from it (Li et al., 2021). The dataset's label column has been identified as the assault column. The column shows the many network intrusion assault types that must be anticipated throughout the classification process.

5.4 Descriptive statistics on dataset

A data set's features are summarized by making use of descriptive statistics. Descriptive statistics entails three crucial measures which are variability, frequency distribution and central tendency (Cooksey, 2020). Descriptive statistics helps to provide the data engineer with a brief overview of the dataset being studied without drawing unnecessary conclusions. One variable can be described using descriptive statistics (univariate analysis) or multiple variables can be described using descriptive statistics (bivariate/multivariate analysis) (Kaliyadan and Kulkarni, 2019). When too many variables exist from the set, using descriptive statistic tools like scatter plots could help illustrate the correlation between those variables.

5.5 Data Normalization

Machine learning uses data normalization to reduce the sensitivity of model training to feature scale (Izonin et al., 2022). This enables the algorithm we use to converge to better weights, which produces a more accurate model. The characteristics are more uniform after the normalization process which would boost the models ability to make better predictions (Aksu, Güzeller and Eser, 2019). The 'StandardScaler( )' method is used to scale down the numerical features without omitting important details or distorting changes in their value ranges. In order to prevent any one feature from predominating the classification process, this technique scales each feature to have a unit variance while removing the mean.

5.6 One-Hot-Encoding

One-hot Encoding (OHE) is the usual process used to refine categorical information (Ul Haq et al., 2019). According to Anderberg, (2014), OHE is frequently used to convert categorical information to numerical features in many conventional data mining jobs. A single variable with n observations and d unique values is transformed by OHE into d binary variables, each with n observations. One-hot encoding is used to convert the protocol_type, service, and flag categorical variables in the NSL_KDD dataset into numerical variables. The categorical data is encrypted using this method, enabling efficient use of these variables throughout the classification process.

5.7 Multi-Class Classification

Multiclass classification plays a crucial role in numerous practical machine learning applications that require the capability to automatically differentiate among a vast array of classes, sometimes numbering in the thousands (Thrampoulidis, Oymak and Soltanolkotabi, 2020). There are 24 different attack types listed in the label column, which are then divided

into four separate attack types: DoS, Probe, U2R, and R2L. The qualities and goals of each attack type are used to establish these classes. The multi-classification work is made simpler by the reclassification, and the findings are easier to understand.

Based on the dataset, the different attack types are being classified to improve the interpretation of the results. The attacks types are thus discussed below:

1. Denial of Service (DoS): DoS attacks attempt to block authorized users from accessing a system by reducing system availability (Obaid, 2020). They force intensive calculation functions on the target by abusing the system's weaknesses or overwhelming it with numerous useless requests.

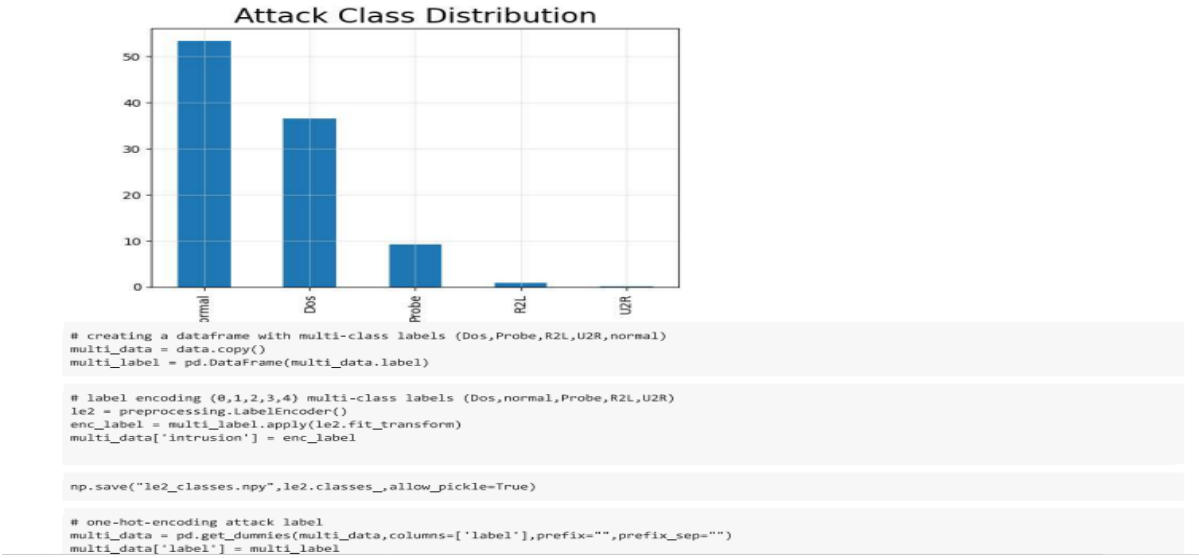
2. Probing Attack (Probe): Assaults such as probe or probe-response assaults pose a fresh risk to collaborative intrusion detection systems. Probing assaults are intended to expose information about the weaknesses of a target system by performing carefully chosen sequences of operations and seeing how the target system or the intrusion detection system (IDS) fronts it responds. Some probing attempts aim to map out the target's IP address space as a preliminary to a campaign of spreading malware.

3. User to Root Attack (U2R): In a User to Root attack, the attacker first succeeds in establishing a user session, usually using an interactive shell, or sometimes a TELNET window on the remote system (Palshikar, 2022). The attacker tries to slowly gain more privileged access to the system until he gains super user access by mixing up approaches.

4. Remote to Local Attack (R2L): The user-to-root attack and the remote to local attack have theoretically similar goals, but the remote to local attack has a more constrained end goal (Dhanabal and Shantharajah, 2015). An attacker conducts such an attack when they transmit packets to the target host that are supposed to reveal security weaknesses that would allow them to abuse the privileges of a local user.

5.8 Attack class Distribution

This shows a graph of the different attack types that are being analysed in the training model. This illustration is found in the graph below in Fig 5.8.




```
# creating a dataframe with multi-class labels (Dos,Probe,R2L,U2R,normal)
multi_data = data.copy()
multi_label = pd.DataFrame(multi_data.label)

# label encoding (0,1,2,3,4) multi-class labels (Dos,normal,Probe,R2L,U2R)
le2 = preprocessing.LabelEncoder()
enc_label = multi_label.apply(le2.fit_transform)
multi_data['intrusion'] = enc_label

np.save("le2_classes.npy",le2.classes_,allow_pickle=True)

# one-hot-encoding attack label
multi_data = pd.get_dummies(multi_data,columns=['label'],prefix="",prefix_sep="")
multi_data['label'] = multi_label
```

Fig 5.1: multi-class label distribution on attack types.

During the aforementioned phase, a varied dataset of system logs and network traffic was gathered for covering various time periods and situations. Both instances of typical system activity and recorded assault events of various kinds are included in the collection. The percentage of the normal system states is determined by the frequency of the events classified as "normal" in the dataset system, and from the visualization above we have more than 50% of the normal system states. It also demonstrates that the majority of attacks are from Denial of Service (DoS), which is more than 30%, followed by Probe.

5.9 Feature Extraction

The process of extracting features entails changing a picture's raw pixel values into more valuable, quantifiable information that can be used in other processes like image processing, pattern recognition, and machine learning (Adekunle and Aiyeniko, 2020). The pearson correlation analysis was performed to extract features, determine how other characteristics affected the attack columns, and blend the extracted features with categorical variables. The Pearson's Product Moment Correlation Coefficient, also known as the Pearson's or Pearson's is a measurement of the strength (direction and magnitude) of the linear relationship between two variables (Obilor and Amadi, 2018).

Using the Pearson correlation coefficient, below is an illustration of labelling based on correlation.

$$\mathbf{r} = \frac{\mathbf{n}(\sum \mathbf{xy}) - (\sum \mathbf{x})(\sum \mathbf{y})}{\sqrt{[\mathbf{n}\sum \mathbf{x^2} - (\sum \mathbf{x})^2][\mathbf{n}\sum \mathbf{y^2} - (\sum \mathbf{y})^2]}}$$

Where,
n is the number of stock pairs,
r is the Pearson correlation coefficient,
xy is the total of the products of the paired stocks.
x = total of the x scores,
y = total of the y scores,

x2 = total of the x scores squared
y2 = total of the y scores squared.

After preprocessing the data and extracting the features, there are more features and columns overall, all of which are numeric which is integrated into the machine learning method.

6 Evaluation

After preprocessing the data and categorizing the attack types, the dataset is divided into training and testing sets, and various machine learning algorithms, including random forests, support vector machines (SVM), K-nearest neighbour (KNN), gradient boosting machines (GBM), neural networks, and reinforcement learning, are used to train the models. With the aid of python libraries (such as numpy, pandas, and matplotlib) and keras in tensor flow, the model's performance is assessed using metrics like accuracy, precision, recall, F1-score, and learning curve for reinforcement learning to gauge its efficiency in accurately detecting and classifying network intrusion attacks.

6.1 Evaluation Metrics

The following Table 6.1 shows the various learning algorithms and the results derived from their training model. The table below which is table 6.1 is a summary of the result from the support vector machine.

Table 6.1: Support Vector Machine Model

Hyperparameters	Metrics		
	Precision	Recall	F1-score
gamma='auto', kernel='poly'	Macro avg = 0.56 Weight avg = 0.92	M avg = 0.51 W avg = 0.93	Accuracy = 0.93 M avg = 0.53 W avg = 0.92

From the results in Table 6.1, it is observed from the metrics that the F1 score has an accuracy of 0.93 which is a result of the mean of precision and recall. The hyperparameters used in the model are the gamma and kernel parameters.

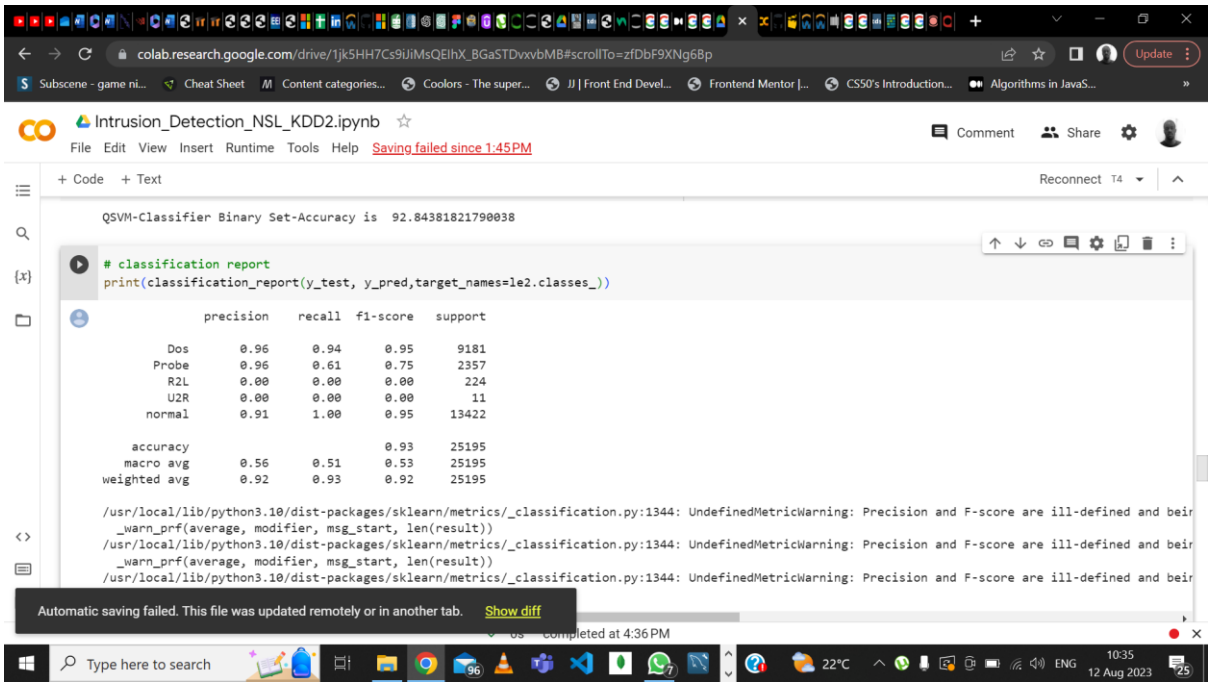


Fig 6.1 Showing the performance metrics for the Support Vector Machine Model

Table 6.2: KNN model

Hyperparameters	Metrics		
	Precision	Recall	F1-score
n_neighbors = 5	Macro avg = 0.84 Weight avg = 0.98	M avg = 0.78 W avg = 0.98	Accuracy = 0.98 M avg = 0.80 W avg = 0.98

Table 6.2 shows the results from the KNN training model

From the results in Table 6.2, it is observed from the metrics that the F1 score has an accuracy of 0.98 which is a result of the mean of precision and recall. The hyperparameters used in the model is the n-neighbors which has a value of 5.

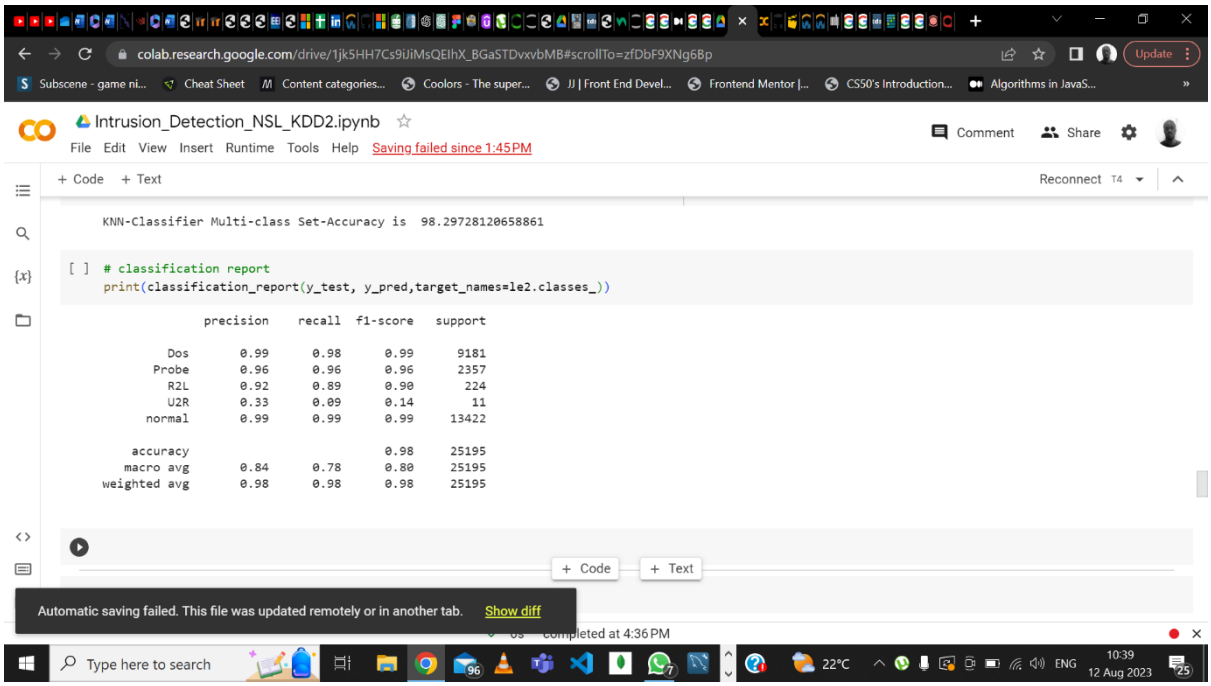


Fig 6.2 Showing the performance metrics for the KNN Model

Table 6.3: Gradient Boosting Model

Hyperparameters	Metrics		
	Precision	Recall	F1-score
Learning_rate = 0.1, n_estimators = 100	Macro avg = 0.76 Weight avg = 0.97	M avg = 0.70 W avg = 0.97	Accuracy = 0.97 M avg = 0.72 W avg = 0.97

From the results in Table 6.3, it is observed from the metrics that the F1 score has an accuracy of 0.97 which is a result of the mean of precision and recall. The hyperparameters used in the model are the learning rate and n-estimators which has a value of 0.1 and 100 respectively. Figure 6.2 highlights the important features of the GBM training model which plays an important role in enhancing the performance and predictive capabilities used.

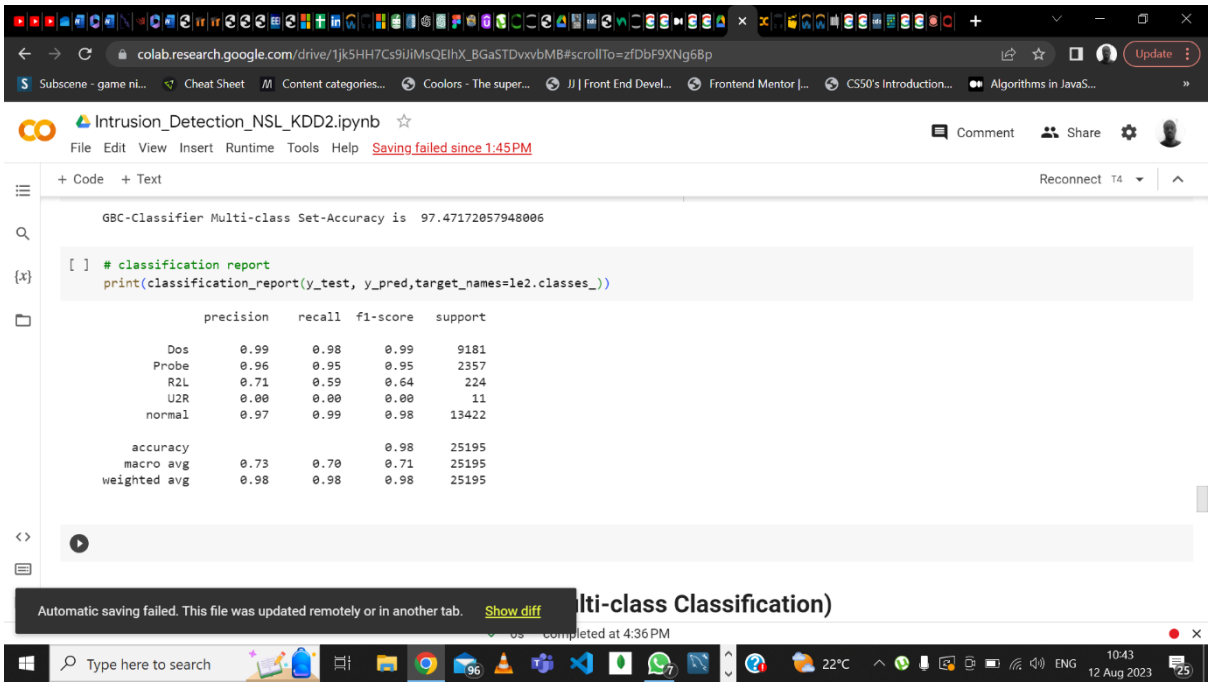


Fig 6.3 Showing the performance metrics for the GB Model

Table 6.4: Random Forest Model

Hyperparameters	Metrics		
	Precision	Recall	F1-score
n_estimators = 100	Macro avg = 0.73 Weight avg = 0.98	M avg = 0.70 W avg = 0.98	Accuracy = 0.97 M avg = 0.71 W avg = 0.98

From the results in Table 6.4, it is observed from the metrics that the F1 score has an accuracy of 0.97 which is a result of the mean of precision and recall. The hyperparameters used in the model is the n-estimators which has a value of 100.

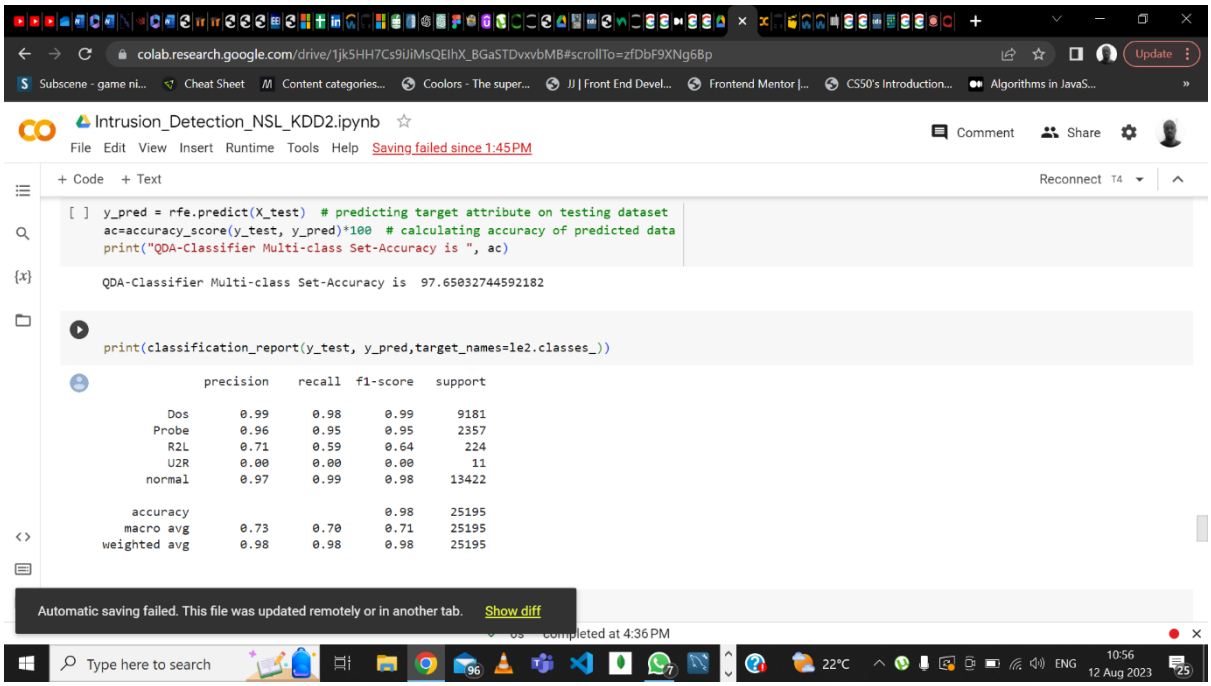


Fig 6.4 Showing the performance metrics for the Random Forest model.

The multi-layer perceptron (MLP) is a feed-forward neural network. It is used in areas like pattern categorization, function approximation, and predictions (Alla, Moumoun and Balouki, 2021). This deep learning approach trains the input variables using a neural network. This model training investigated the three important hyperparameters which include Epoch, Batch size, and Learning rate to learn the well a deep learning algorithm that used neural networks for training performed. Understanding how these hyperparameters of F1-score, accuracy, and recall affect a model's precision was crucial for the training. This shows the results from the Multi-Layer Perceptron Model in table 6.5 below.

Table 6.4: Multi-Layer Perceptron Model Metrics

Accuracy	0.9704
Recall	0.9665
F1-score	0.96783
Precision	0.96916

From the results in Table 6.2, it is observed from the metrics that the F1 score has a value of 0.96 while the accuracy of the model is 0.97. which is a result of the mean of precision and recall. This presents the findings obtained from the experiments using the hyperparameters such as Epoch= 100, Batch size= 5000.

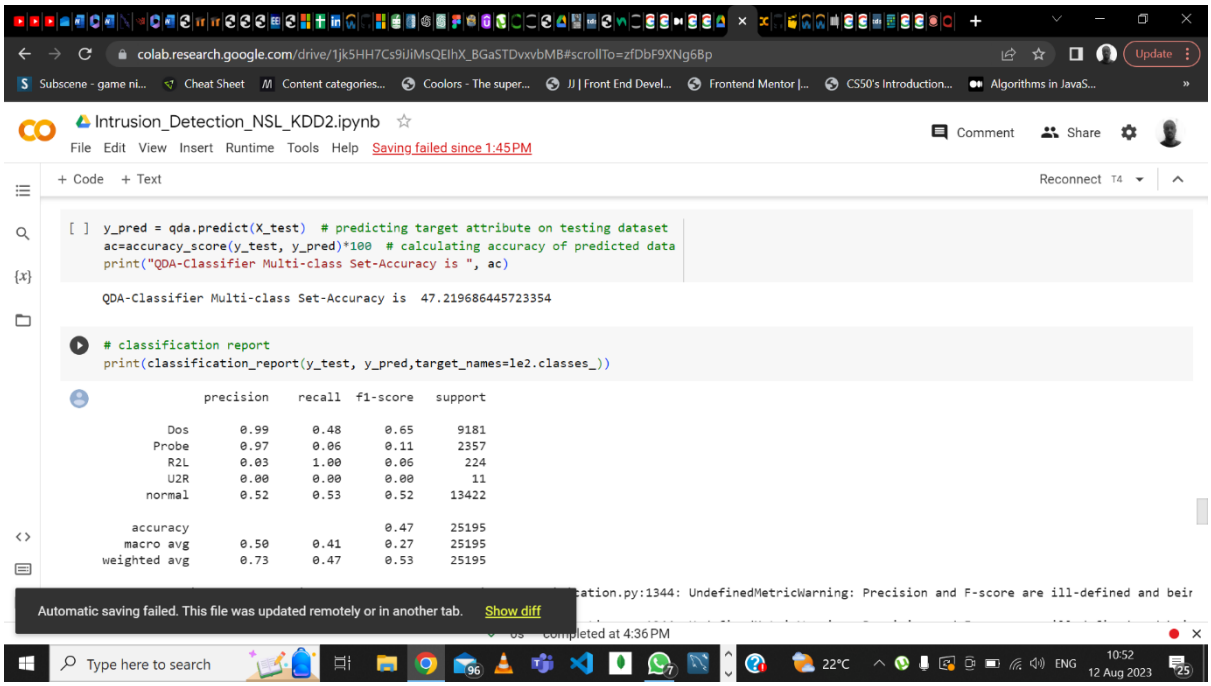


Fig 6.3 Showing the performance metrics for the multi-layer perception model.

6.2 Reinforcement Learning

Reinforcement learning presents a promising method to improve the capabilities of Intrusion Detection Systems (IDS) by enabling systems to learn and adjust their behavior based on feedback received from the environment. In order to understand this section clearer, the following terms would be defined:

Agent: An entity that can see, investigate, and respond to its environment.

Environment: an environment or scenario in which an agent is present or encircled by. Since the environment is assumed to be stochastic in RL, it is essentially random.

Action: The steps an agent takes inside the environment are referred to as actions.

State: In response to each action the agent does, the environment returns a state.

Reward: This is a case where the environment provides feedback to the agent in order to assess its performance.
Policy: Based on the present state, policy is a technique that the agent uses to determine what to do next.

Value: This explains the fact that long-term returns are anticipated to be lower than short-term returns due to the discount factor.

Intrusion Detection Environment: This is a unique Gym setting created especially for the intrusion detection system. Dataset_path = '/content/data.npy' is used as the dataset path during initialization, and it imports data using Numpy. It also determines the observation space, action space, current step, and maximum steps. The actions connected to identifying intrusions are represented in the action space, whereas the observation space represents the network traffic properties.

The reset: The environment is restored to its original state using the reset technique.

The calculate reward: It uses the current state to determine the reward; the award is the total of the state values (Badr, 2022). Based on the observed network traffic, the reward function is created to give feedback on the agent's choice.

The step: This performs an action, modifies the environment, and then returns the subsequent state, reward, done flag, and other details.

DQN Agent: The DQN technique is implemented here in order to train an agent to make judgments in an intrusion detection environment (Badr, 2022). The agent uses network traffic observations as input and decides what to do depending on its current policy, which it learns and updates through interactions with the IDS environment. It sets the state_size and action_size initial values. In order to learn from previous interactions and enhance its decision-making, the agent's experience is saved in a replay memory buffer.

Build model: It is a neural network model that makes use of the Keras sequential API. The model has three dense, fully connected layers that are activated by Rectified Linear Units (ReLUs), and the output layer has a linear activation function (Tariq et al., 2022). The mean squared error (MSE) loss and the Adam optimizer are used to build the model. Agent experiences are stored in the agents' memory, which is implemented as a deque (double-ended queue) with a maximum length. The algorithm update equation's discount factor is represented by the gamma parameter. The exploration rate, which changes over time in accordance with epsilon decay, is determined by the epsilon parameter. The minimal exploration rate is indicated by the parameter epsilon min. The optimizer's rate of learning is managed via the learning rate option. The learning curve is plotted based on the total rewards to be obtained in each episode. The graph below in Fig 6.3 shows the learning curve from the training model

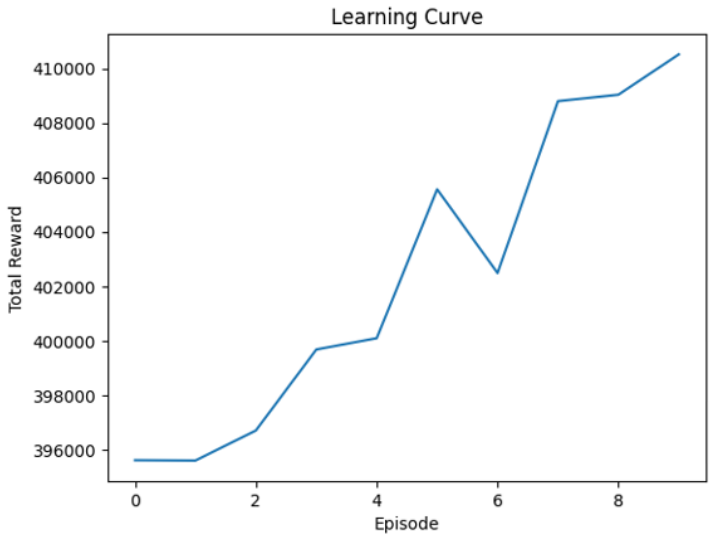


Fig 6.3: Reinforcement learning curve

From the above graph in fig 6.3, it shows the assessment of how well the DQN algorithm is working and keeping track of how the agent's total rewards are convergent. It shows that the DQN model has summed up 410,000 rewards in total and is learning non-linearly but progressively over its 10 episodes.

6.3 Discussion

From the analysis derived from the different training algorithms, with the data preprocessing steps and attack class categorization, the data set was split into testing and training sets where the application of different machine learning models which includes random forests, support vector machines (SVM), K-nearest neighbour (KNN), Gradient Boosting Machines (GBM), neural networks and reinforcement learning were all employed for model training. In the analysis of the study, the training model performance was evaluated using key metrics like accuracy, precision, recall, F1-score and learning curve for reinforcement learning to thus measure the effectiveness acquired to detect and classify network intrusion systems.
Further, from the findings from attack class distributions, the percentage of the normal system states is calculated based on the frequency of the events classified as "normal" in the dataset system, and from the results, it shows that there is then 50% normal system states and it also shows that most attacks are from Denial of Service (DoS), which is more than 30% followed by Probe.

For the SVM model, the reported F1-score of 0.93 shows that precision and recall are well-balanced. The accuracy is also 0.93, as it is highlighted. The overall average F1-score and recall are significantly lower at 0.51 and 0.53 whereas the weighted average precision and recall are reasonably high at 0.92 and 0.93, respectively. The model appears to perform well in some classes while faltering in others, according to the discrepancy. The findings demonstrate that the SVM model offers respectable overall accuracy, although further optimisation may be beneficial to achieve more evenly distributed performance across various attack types.

With n_neighbors set to 5, the KNN model produces impressive results. High precision and recall balance, which result in a well-rounded performance, are indicated by the F1-score

of 0.98. The model's ability to effectively categorise instances is supported by its accuracy of 0.98. High weighted and macro average indicators show that performance is consistent across classes. These findings imply that the KNN model is capable of robustly classifying network intrusion data and capturing its underlying patterns.

The Gradient Boosting model performs admirably when learning_rate is set to 0.1 and n_estimators are set to 100. It performs well, as seen by its accuracy and F1-score of 0.97. The macro average and weighted measures also show consistent performance across various attack types. The model's success in achieving a high F1-score and accuracy suggests that it is effective at identifying network intrusions and correctly classifying them.

With n_estimators set to 100, the Random Forest model generates outcomes that are consistent with those of the earlier models. Solid performance is indicated by an F1-score and accuracy of 0.97. The model's proficiency in managing various assault types is further supported by the weighted and macro average metrics. The Random Forest is suitable for network intrusion detection, as shown by these results, which concur with those of the other models.

With the hyperparameters Epoch=100, Batch size=5000, and Learning rate set to an undetermined value, the Multi-Layer Perceptron (MLP) model, a deep learning technique, is tested. The outcomes, which include an accuracy of 0.9704 and an F1-score of 0.96783, highlight the model's ability to identify intricate patterns in the data. A balanced performance is also indicated by the recall and precision values. The model achieves good accuracy, but there may be potential for improvement in the precision and recall trade-off, as the F1-score is slightly lower than the F1-scores obtained by some of the other models.

The evaluation's overall findings show that the K-Nearest Neighbour (KNN), Gradient Boosting, and Random Forest models regularly outperform other models in effectively identifying and categorising network intrusion attacks. Although the Support Vector Machine (SVM) model performs rather well, it might benefit from more tuning to enhance its balance across various assault types. The Multi-Layer Perceptron (MLP), which achieves high accuracy with some precision and recall loss, is a good example of the promise of deep learning approaches. This suggests that a mix of various models might be used to further improve the detection abilities and robustness of the intrusion detection system. The results of this evaluation are consistent with earlier studies, like A. Aziz, Hanafi and Hassanien, (2017) and Uikey and Gyanchandani, (2019), that demonstrate the efficiency of ensemble techniques for intrusion detection, such as Random Forest and Gradient Boosting. According to a study by Nikhitha and Jabbar, (2019), the KNN model was used for the intrusion detection systems. The trained machine learning classifier, KNN performed best in terms of classification accuracy. In evaluating the model's implementation, experimental analysis was done using the ISCX dataset. According to the experimental results, the suggested model had an enhanced accuracy of 99.96%. Also, another study proposed by Dini and Saponara, (2021) used K-nearest neighbors and artificial neural network to develop an algorithm for an intrusion detection system. it was realized from their results that KNN was better than the other algorithm both in the classification of the anomaly class and in the normal class in terms of accuracy. Furthermore, The RF algorithm produces accuracy results for DOS, Probe, R2L, and U2R of 99.9%, 99.9%, 99.8%, and 99.0% respectively in the method suggested by Farnaaz and Jabbar,

(2016). The investigation of deep learning methods, such as MLP, also demonstrates their capacity to recognise complex patterns seen in network traffic data.

However, more research into hyperparameter tweaking and model ensembling may lead to even better outcomes and performance that is more evenly distributed across various attack classes. The dynamic nature of network attacks and the changing threat landscape must also be taken into account. The effectiveness of intrusion detection systems must be maintained by routine updates and model retraining utilising the most recent data and attack patterns. Therefore, the study's implementation of the Deep Q-Network (DQN) algorithm for network intrusion detection brings an intriguing new perspective to the analysis of intrusion detection models. The DQN algorithm attempts to learn from past data and differentiate between legitimate and malicious behaviours, including various sorts of assaults. A learning curve that graphs the total reward against the number of episodes is the foundation for the DQN algorithm's evaluation. Insights into the algorithm's effectiveness can be gained from the observed pattern in the learning curve, which shows upward movements from episode 395000 to 411000. While deep learning algorithms like DQN can identify complex patterns, they can also be more complicated and difficult to understand when compared to conventional models. DQN is one example of a deep learning algorithm that can be computationally demanding. Therefore, when integrating DQN into intrusion detection systems, it is crucial to evaluate the trade-off between computational resources and performance gains. Despite this, the rising trend in the learning curve indicates that throughout the duration of training episodes, the DQN algorithm is gradually improving its performance. This suggests that the DQN is getting better at differentiating between legitimate and malicious behaviour as it learns from historical data. Therefore, it is only a matter of time before the DQN outperforms the other algorithms.

7 Conclusion.

The primary objective of this study were to develop and evaluate a comprehensive approach for accurately categorizing network intrusion attacks using a combination of traditional supervised machine learning algorithms, deep learning techniques, and reinforcement learning. The focus was on classifying attack types into four distinct groups: DoS, Probe, U2R, and R2L. The objective of the study also include the assessment of the effectiveness of the different algorithms, optimising their performance, and exploring the potential of the DQN algorithm for intrusion detection

7.1 Achievement of objectives

The study has successfully accomplished its research objectives by employing a systematic methodology. The researcher performed multi-class classification using well-established algorithms such as Random Forests, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Gradient Boosting Machines (GBM), achieving promising results in accurately categorizing attack types. Furthermore, feature engineering through Pearson correlation analysis enhanced the performance of these algorithms. The researcher extended investigation into deep learning by employing the Multi-Layer Perceptron (MLP) model, a neural network-based algorithm. Through rigorous hyperparameter tuning, the researcher

optimized the MLP's performance, highlighting the significance of parameters like Epoch, Batch_size, and Learning_rate. Furthermore, the researcher ventured into the realm of reinforcement learning by developing an Intrusion Detection System (IDS) using the DQN algorithm. This system showcased the potential of learning from historical data to differentiate between normal and malicious behaviours, based on Probing, DoS, R2L, and U2R attacks. The performance of the DQN algorithm was assessed using a learning curve that showed improvements in detecting attack types.

7.2 Key Findings

The investigation yielded significant findings:

1. Traditional machine learning algorithms, when properly tuned and enhanced through feature engineering, can effectively categorize network intrusion attacks.
2. Deep learning, particularly the Multi-Layer Perceptron model, offers a powerful tool for intrusion detection, with the potential to capture complex patterns.
3. Reinforcement learning, exemplified by the DQN algorithm, has promising capabilities in learning from historical data and improving intrusion detection accuracy.

7.3 Implications and Efficacy

The implications of the research are substantial. A robust Intrusion Detection System (IDS) that combines the strengths of various algorithms can significantly increase network security. The efficacy of the approach employed is given by the high accuracy, precision, recall, and F1-score values achieved by the models. The study generally demonstrates that a well-structured hybrid of machine learning and deep learning techniques can provide a comprehensive solution for network intrusion detection.

7.4 Limitations

This research is not without limitations. The use of historical data may not fully capture future attack scenarios, and the effectiveness of the DQN algorithm might vary depending on the environment. Furthermore, the computational complexity of certain algorithms, particularly deep learning, needs to be carefully considered in practical implementations. This study did not do that.

7.5 Future Work and Commercialization

Commercially, this research opens avenues for developing robust and adaptable Intrusion Detection Systems (IDS) that cater to the increasing sophistication of cyber threats. The potential for integration into network security solutions is significant.

Future research could focus on:

1. Exploring ensemble techniques that combine the strengths of various algorithms to further enhance detection accuracy.

2. Investigating more advanced hyperparameter optimization methods for deep learning models.
3. Assessing the adaptability of the DQN algorithm to dynamic and evolving attack environments.

References

A. Aziz, A.S., Hanafi, S.E.-O. and Hassanien, A.E. (2017). Comparison of Classification Techniques Applied for Network Intrusion Detection and Classification. *Journal of Applied Logic*, 24, pp.109–118. doi:https://doi.org/10.1016/j.jal.2016.11.018.

Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J. and Ahmad, F. (2020). Network Intrusion Detection system: A Systematic Study of Machine Learning and Deep Learning Approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1). doi:https://doi.org/10.1002/ett.4150.

Alarqan, M.A., Zaaba, Z.F. and Almomani, A. (2019). Detection Mechanisms of DDoS Attack in Cloud Computing Environment: A Survey. *Communications in Computer and Information Science*, pp.138–152. doi:https://doi.org/10.1007/978-981-15-2693-0_10.

Alghamdi, T.A. and Javaid, N. (2022). A Survey of Preprocessing Methods Used for Analysis of Big Data Originated from Smart Grids. *IEEE Access*, pp.1–1. doi:https://doi.org/10.1109/access.2022.3157941.

Aljabri, M., Aljameel, S.S., Mohammad, R.M.A., Almotiri, S.H., Mirza, S., Anis, F.M., Aboulnour, M., Alomari, D.M., Alhamed, D.H. and Altamimi, H.S. (2021). Intelligent Techniques for Detecting Network Attacks: Review and Research Directions. *Sensors*, 21(21), p.7070. doi:https://doi.org/10.3390/s21217070.

Alla, H., Moumoun, L. and Balouki, Y. (2021). A Multilayer Perceptron Neural Network with Selective-Data Training for Flight Arrival Delay Prediction. *Scientific Programming*, [online] 2021, p.e5558918. doi:https://doi.org/10.1155/2021/5558918.

Alzahrani, A.O. and Alenazi, M.J.F. (2021). Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks. *Future Internet*, 13(5), p.111. doi:https://doi.org/10.3390/fi13050111.

Ares, B., Morán-Fernández, L. and Bolón-Canedo, V. (2022). Reduced Precision Discretization Based on Information Theory. 207, pp.887–896. doi:https://doi.org/10.1016/j.procs.2022.09.144.

Asharf, J., Moustafa, N., Khurshid, H., Debie, E., Haider, W. and Wahab, A. (2020). A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of

Things: Challenges, Solutions and Future Directions. *Electronics*, 9(7), p.1177. doi:https://doi.org/10.3390/electronics9071177.

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

Badr, Y. (2022). Enabling Intrusion Detection Systems with Dueling Double Deep *Q*-learning. *Digital Transformation and Society*. doi:https://doi.org/10.1108/dts-05-2022-0016.

Bajtoš, T., Sokol, P. and Mézešová, T. (2019). Multi-stage Cyber-Attacks Detection in the Industrial Control Systems. *Studies in systems, Decision and Control*, pp.151–173. doi:https://doi.org/10.1007/978-3-030-31328-9_8.

Barto, A. and Dietterich, T. (2004). *Reinforcement Learning and Its Relationship to Supervised Learning*. [online] Available at: http://all.cs.umass.edu/pubs/2004/barto_d_04.pdf.

Biswas, S.K. (2018). Intrusion Detection Using Machine Learning: A Comparison Study. International Journal of Pure and Applied Mathematics. Volume 118 No. 19.

Breiman, L. (2001). Random Forests. *Machine Learning*, [online] 45(1), pp.5–32. doi:https://doi.org/10.1023/a:1010933404324.

Canadian Institute for Cybersecurity (2023). *NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. [online] Www.unb.ca. Available at: https://www.unb.ca/cic/datasets/nsl.html [Accessed Jul. 2023].

Carneiro, T., Medeiros Da Nobrega, R.V., Nepomuceno, T., Bian, G.-B., De Albuquerque, V.H.C. and Filho, P.P.R. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, [online] 6, pp.61677–61685. doi:https://doi.org/10.1109/access.2018.2874767.

Chen, H., Li, T., Sang, B. and Yuan, Z. (2023). Feature Grouping and Selection with Graph Theory in Robust Fuzzy Rough Approximation Space. 31(1), pp.213–225. doi:https://doi.org/10.1109/tfuzz.2022.3185285.

Chowdhury, M. N., Ferens, H., and Ferens, M. (2016). Network Intrusion Detection Using Machine Learning. Int'l Conf. Security and Management.

Cooksey, R.W. (2020). Descriptive Statistics for Summarising Data. *Illustrating Statistical Procedures: Finding Meaning in Quantitative Data*, [online] pp.61–139. doi:https://doi.org/10.1007/978-981-15-2537-7_5.

Cybenko G. (1989). Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. 2(4):303–14

Dhanabal, L. and Shantharajah, S. (2015). A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*, [online] 4. doi:https://doi.org/10.17148/IJARCCE.2015.4696.

Dini, P. and Saponara, S. (2021). Analysis, Design, and Comparison of Machine-Learning Techniques for Networking Intrusion Detection. *Designs*, 5(1), p.9. doi:https://doi.org/10.3390/designs5010009.

Duc, A., Dziembowski, S. and Faust, S. (2018). Unifying Leakage Models: from Probing Attacks to Noisy Leakage. *Journal of Cryptology*, 32(1), pp.151–177. doi:https://doi.org/10.1007/s00145-018-9284-1.

Ekici, S. (2012). Support Vector Machines for Classification and Locating Faults on Transmission Lines. *Applied Soft Computing*, 12(6), pp.1650–1658. doi:https://doi.org/10.1016/j.asoc.2012.02.011.

Esteve, M., Aparicio, J., Rodriguez-Sala, J.J. and Zhu, J. (2023). Random Forests and the Measurement of super-efficiency in the Context of Free Disposal Hull. *European Journal of Operational Research*, 304(2), pp.729–744. doi:https://doi.org/10.1016/j.ejor.2022.04.024.

Fan, C., Chen, M., Wang, X., Wang, J. and Huang, B. (2021). A Review on Data Preprocessing Techniques toward Efficient and Reliable Knowledge Discovery from Building Operational Data. *Frontiers in Energy Research*, 9. doi:https://doi.org/10.3389/fenrg.2021.652801.

Fan, C., Sun, Y., Zhao, Y., Song, M. and Wang, J. (2019). Deep learning-based Feature Engineering Methods for Improved Building Energy Prediction. *Applied Energy*, 240, pp.35–45. doi:https://doi.org/10.1016/j.apenergy.2019.02.052.

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

Gallego, V. and Insua, D.R. (2021). Current Advances in Neural Networks. 9(1), pp.197–222. doi:https://doi.org/10.1146/annurev-statistics-040220-112019.

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn and TensorFlow concepts, tools, and Techniques to Build Intelligent Systems*. 2nd ed. O'Reilly Media, Inc.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*. MIT Press.

Gupta, A.R. and Agrawal, J. (2020). The multi-demeanor Fusion Based Robust Intrusion Detection System for Anomaly and Misuse Detection in Computer Networks. *Journal of Ambient Intelligence and Humanized Computing*. doi:https://doi.org/10.1007/s12652-020-01974-4.

Han, J., Pei, J. and Tong, H. (2022). *Data Mining*. Morgan Kaufmann.

Heidari, A. and Jabraeil Jamali, M.A. (2022). Internet of Things Intrusion Detection systems: A Comprehensive Review and Future Directions. *Cluster Computing*. doi:https://doi.org/10.1007/s10586-022-03776-z.

Hossain, M.D., Ochiai, H., Fall, D. and Kadobayashi, Y. (2020). LSTM-based Network Attack Detection: Performance Comparison by Hyper-parameter Values Tuning. *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. doi:https://doi.org/10.1109/cscloud-edgecom49738.2020.00020.

Hu, Z., Beuran, R. and Tan, Y. (2020). *Automated Penetration Testing Using Deep Reinforcement Learning*. [online] IEEE Xplore. doi:https://doi.org/10.1109/EuroSPW51379.2020.00010.

Kaliyadan, F. and Kulkarni, V. (2019). Types of variables, Descriptive statistics, and Sample Size. *Indian Dermatology Online Journal*, [online] 10(1), pp.82–86. doi:https://doi.org/10.4103/idoj.IDOJ_468_18.

Katole, R.A., Sherekar, S.S. and Thakare, V.M. (2018). Detection of SQL Injection Attacks by Removing the Parameter Values of SQL Query. *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. doi:https://doi.org/10.1109/icisc.2018.8398896.

Khan, M.Y., Qayoom, A., Nizami, M.S., Siddiqui, M.S., Wasi, S. and Raazi, S.M.K.-R. (2021). Automated Prediction of Good Dictionary EXamples (GDEX): a Comprehensive Experiment with Distant Supervision, Machine Learning, and Word Embedding-Based Deep Learning Techniques. *Complexity*, 2021, pp.1–18. doi:https://doi.org/10.1155/2021/2553199.

Kim, S.K., Kirchner, E.A., Stefes, A. and Kirchner, F. (2017). Intrinsic Interactive Reinforcement Learning – Using error-related Potentials for Real World human-robot Interaction. *Scientific Reports*, 7(1). doi:https://doi.org/10.1038/s41598-017-17682-7.

Klug, M., Barash, Y., Bechler, S., Resheff, Y.S., Tron, T., Ironi, A., Soffer, S., Zimlichman, E. and Klang, E. (2019). A Gradient Boosting Machine Learning Model for Predicting Early Mortality in the Emergency Department Triage: Devising a Nine-Point Triage Score. *Journal of General Internal Medicine*, [online] 35(1), pp.220–227. doi:https://doi.org/10.1007/s11606-019-05512-7.

Lopez-Martin, M., Carro, B. and Sanchez-Esguevillas, A. (2020). Application of Deep Reinforcement Learning to Intrusion Detection for Supervised Problems. *Expert Systems with Applications*, 141, p.112963. doi:https://doi.org/10.1016/j.eswa.2019.112963.

Lorenzen, S.S., Igel, C. and Seldin, Y. (2019). On PAC-Bayesian Bounds for Random Forests. *Machine Learning*, 108(8-9), pp.1503–1522. doi:https://doi.org/10.1007/s10994-019-05803-4.

Lu, H. and Mazumder, R. (2018). *Randomized Gradient Boosting Machine*. [online] Available at: http://web.mit.edu/haihao/www/papers/RGBM.pdf [Accessed 8 Aug. 2021].

Maingi, M.N. (2015). *Survey on Data Preprocessing Concept Applicable in Data Mining*. [online] Available at: https://www.semanticscholar.org/paper/Survey-on-Data-Preprocessing-Concept-Applicable-in-Maingi/9c7be10cf5aa11bdfc59ac94d4100c5863f31ec1 [Accessed 24 Jul. 2023].

Minsky, M. and Papert, S. (1969). *Perceptrons; an Introduction to Computational Geometry*.

Muller, S., Lancrenon, J., Harpes, C., Traon, Y.L., Gombault, S. and Bonnin, J.-M. (2018). A training-resistant Anomaly Detection System. *Computers & Security*, 76, pp.1–11. doi:https://doi.org/10.1016/j.cose.2018.02.015.

Muzzammel, R. and Raza, A. (2020). A Support Vector Machine Learning-Based Protection Technique for MT-HVDC Systems. *Energies*, 13(24), p.6668. doi:https://doi.org/10.3390/en13246668.

Nargesian, F., Asudeh, A. and Jagadish, H.V. (2022). Responsible Data Integration: Next-generation Challenges. doi:https://doi.org/10.1145/3514221.3522567.

Nayak, J., Naik, B. and Behera, H.S. (2015). A Comprehensive Survey on Support Vector Machine in Data Mining Tasks: Applications & Challenges. *International Journal of Database Theory and Application*, 8(1), pp.169–186. doi:https://doi.org/10.14257/ijdta.2015.8.1.18.

Bush, S.F and Evans, S.C.(2001) Information assurancedesign and assessment. Final report, General ElectricResearch and Development Center, August

Nikhitha, M. and Jabbar, M.A. (2019). K Nearest Neighbor Based Model for Intrusion Detection System. *International Journal of Recent Technology and Engineering*, 8(2), pp.2258–2262. doi:https://doi.org/10.35940/ijrte.b2458.078219.

Obaid, H.S. (2020). Denial of Service Attacks: Tools and Categories. *International Journal of Engineering Research and*, V9(03). doi:https://doi.org/10.17577/ijertv9is030289.

Obilor, E.I. and Amadi, E.C. (2018). *(PDF) Test for Significance of Pearson's Correlation Coefficient ( )*. [online] ResearchGate. Available at: https://www.researchgate.net/publication/323522779_Test_for_Significance_of_Pearson.

Palshikar, A. (2022). What Distinguishes Binary from multi-class Intrusion Detection systems: Observations from Experiments. *International Journal of Information Management Data Insights*, 2(2), p.100125. doi:https://doi.org/10.1016/j.jjimei.2022.100125.

Papanikolaou, A., Alevizopoulos, A., Ilioudis, C., Demertzis, K. and Rantos, K. (2023). An autoML Network Traffic Analyzer for Cyber Threat Detection. doi:https://doi.org/10.1007/s10207-023-00703-0.

Peterson, L. (2009). K-nearest Neighbor. *Scholarpedia*, [online] 4(2), p.1883. doi:https://doi.org/10.4249/scholarpedia.1883.

Pham, S.T., Vo, P.S. and Nguyen, D.N. (2021). Effective Electrical Submersible Pump Management Using Machine Learning. *Open Journal of Civil Engineering*, 11(01), pp.70–80. doi:https://doi.org/10.4236/ojce.2021.111005.

Qiu, H., Dong, T., Zhang, T., Lu, J., Memmi, G. and Qiu, M. (2020). Adversarial Attacks against Network Intrusion Detection in IoT Systems. *IEEE Internet of Things Journal*, pp.1–1. doi:https://doi.org/10.1109/jiot.2020.3048038.

Qureshi, A.U.H., Larijani, H., Yousefi, M., Adeel, A. and Mtetwa, N. (2020). An Adversarial Approach for Intrusion Detection Systems Using Jacobian Saliency Map Attacks (JSMA) Algorithm. *Computers*, 9(3), p.58. doi:https://doi.org/10.3390/computers9030058.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M. and Herrera, F. (2017). A Survey on Data Preprocessing for Data Stream mining: Current Status and Future Directions. *Neurocomputing*, [online] 239, pp.39–57. doi:https://doi.org/10.1016/j.neucom.2017.01.078.

Ramírez-Gallego, S., García, S., Mouriño-Talín, H., Martínez-Rego, D., Bolón-Canedo, V., Alonso-Betanzos, A., Benítez, J.M. and Herrera, F. (2015). Data discretization: Taxonomy and Big Data Challenge. *WIREs Data Mining and Knowledge Discovery*, 6(1), pp.5–21. doi:https://doi.org/10.1002/widm.1173.

Raschka, S., Patterson, J. and Nolet, C. (2020). Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information*, 11(4), p.193. doi:https://doi.org/10.3390/info11040193.

Ravipati, R.D. and Abualkibash, M. (2019). *Intrusion Detection System Classification Using Different Machine Learning Algorithms on KDD-99 and NSL-KDD Datasets - a Review Paper*. [online] papers.ssrn.com. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3428211.

Rehman, E., Haseeb-ud-Din, M., Malik, A.J., Khan, T.K., Abbasi, A.A., Kadry, S., Khan, M.A. and Rho, S. (2022). Intrusion Detection Based on Machine Learning in the Internet of things, Attacks and Counter Measures. *The Journal of Supercomputing*, 78(6), pp.8890–8924. doi:https://doi.org/10.1007/s11227-021-04188-3.

Ridzuan, F. and Wan Zainon, W.M.N. (2019). A Review on Data Cleansing Methods for Big Data. *Procedia Computer Science*, 161, pp.731–738. doi:https://doi.org/10.1016/j.procs.2019.11.177.

Rosenblatt F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. 65(6):386

Sadikin, F., van Deursen, T. and Kumar, S. (2020). A Hybrid Zigbee IoT Intrusion Detection System Using Secure and Efficient Data Collection. *Internet of Things*, p.100306. doi:https://doi.org/10.1016/j.iot.2020.100306.

Serrano, W. (2017). Smart Internet Search with Random Neural Networks. *European Review*, [online] 25(2), pp.260–272. doi:https://doi.org/10.1017/S1062798716000594.

Sethi, K., Sai Rupesh, E., Kumar, R., Bera, P. and Venu Madhav, Y. (2019). A context-aware Robust Intrusion Detection system: A Reinforcement learning-based Approach. *International Journal of Information Security*. doi:https://doi.org/10.1007/s10207-019-00482-7.

Shin, J., Badgwell, T.A., Liu, K.-H. and Lee, J.H. (2019). Reinforcement Learning – Overview of Recent Progress and Implications for Process Control. *Computers & Chemical Engineering*, 127, pp.282–294. doi:https://doi.org/10.1016/j.compchemeng.2019.05.029.

Subramanian, A., Chitlangia, S. and Baths, V. (2022). Reinforcement Learning and Its Connections with Neuroscience and Psychology. *Neural Networks*, 145, pp.271–287. doi:https://doi.org/10.1016/j.neunet.2021.10.003.

Suroso, J.S. and Prastya, C.P. (2020). Cyber Security System with SIEM and Honeypot in Higher Education. *IOP Conference Series: Materials Science and Engineering*, 874(1), p.012008. doi:https://doi.org/10.1088/1757-899x/874/1/012008.

Suyal, M. and Goyal, P. (2022). A Review on Analysis of K-Nearest Neighbor Classification Machine Learning Algorithms Based on Supervised Learning. *International Journal of Engineering Trends and Technology*, 70(7), pp.43–48. doi:https://doi.org/10.14445/22315381/ijett-v70i7p205.

Talaei Khoei, T. and Kaabouch, N. (2023). A Comparative Analysis of Supervised and Unsupervised Models for Detecting Attacks on the Intrusion Detection Systems. *Information*, 14(2), p.103. doi:https://doi.org/10.3390/info14020103.

Tariq, Z.U.A., Baccour, E., Erbad, A., Guizani, M. and Hamdi, M. (2022). *Network Intrusion Detection for Smart Infrastructure Using Multi-armed Bandit Based Reinforcement Learning in Adversarial Environment*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICCWS56285.2022.9998440.

Taunk, K., De, S., Verma, S. and Swetapadma, A. (2019). *A Brief Review of Nearest Neighbor Algorithm for Learning and Classification*. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICCS45141.2019.9065747.

Uikey, R. and Gyanchandani, M. (2019). Survey on Classification Techniques Applied to Intrusion Detection System and Its Comparative Analysis. doi:https://doi.org/10.1109/icces45898.2019.9002129.

Vapnik, V.N. (1998). *Statistical Learning Theory*. Wiley-Interscience.

Verma, A. and Ranga, V. (2019). Machine Learning Based Intrusion Detection Systems for IoT Applications. *Wireless Personal Communications*. [online] doi:https://doi.org/10.1007/s11277-019-06986-8.

Xu, N. (2019). Understanding the Reinforcement Learning. *Journal of Physics: Conference Series*, 1207, p.012014. doi:https://doi.org/10.1088/1742-6596/1207/1/012014.

Xu, Y., Wang, Q., An, Z., Wang, F., Zhang, L., Wu, Y., Dong, F., Qiu, C.-W., Liu, X., Qiu, J., Hua, K., Su, W., Xu, H., Han, Y., Cao, X., Liu, E., Fu, C., Yin, Z., Liu, M. and Roepman, R. (2021). Artificial Intelligence: A Powerful Paradigm for Scientific Research. *The Innovation*, [online] 2(4), p.100179. doi:https://doi.org/10.1016/j.xinn.2021.100179.