German International University

Faculty of Informatics and Computer Science

IntrusionHunter: Detection of Cyber Threats in Big Data

Bachelor Thesis

Author:	Hashem Mohamed
Supervisors:	Dr. Alia El-Bolock
	Dr. Caroline Sabty
Submission Date:	19 January, 2023

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowledgment has been made in the text to all other material used

Hashem Mohamed
19 January, 2023

Abstract

The limitations of previous intrusion detection systems, such as the use of outdated datasets and the focus on binary classification, emphasize the importance of creating a new and improved system that utilizes advanced classification techniques and a more up-to-date dataset for better identification and generalization of results. This work investigates an anomaly-based intrusion detection system (IntrusionHunter) using the CSE-CICIDS2018 dataset, which contains a wide range of network-based attacks on both the network level and the application level. The research includes several phases of data pre-processing, feature selection, and model classification. The classification models used are Random Forest, XGBoost, Convolutional Neural Networks, and Deep Neural Networks, which are chosen for their popularity and effectiveness in solving classification problems. The aim is to identify the best model for each type of classification: binary classification, multiclass classification with 7 classes, and multiclass classification with 15 classes. The results show that Random Forest and XGBoost algorithms outperform state-of-the-art intrusion detection systems in binary and multiclass classification (15 classes) in terms of accuracy and F1 score. The results also indicate that the imbalance of the dataset needs to be handled in the future in order to improve the performance of deep learning techniques.

Contents

Chapter 1

Introduction

Big data poses a major threat to cybersecurity, making it crucial for organizations to implement robust security measures. As big data technologies continue to advance, a new approach to information protection is needed. Confidentiality, integrity, and availability are the key goals of cybersecurity. Security experts work constantly to protect sensitive information from threats and keep it secure, but the most valuable data is also the most vulnerable, as it often resides on unsecured networks. Previous studies in intrusion detection systems have limitations such as outdated datasets, limited scope and generalizability due to only using a subset of data, and focus on binary classification. This highlights the need for recent and comprehensive studies using advanced classification techniques on updated datasets for improved identification and generalization of results.

This work presents an anomaly-based intrusion detection system using the CSE-CICIDS2018 [13] dataset: IntrusionHunter. The CSE-CICIDS2018 [13] dataset contains a wide range of network-based attacks on both the network level and the application level, making it an ideal dataset for evaluating intrusion detection systems. This is done through performing several phases: data cleaning, feature selection, and model classification. Data cleaning includes removal of unused columns, duplicates, and null values. feature selection is done using the random forest classifier, resulting in the selection of the features with the highest importances. The classification algorithms used are Random Forest, XGBoost, Convolutional Neural Networks, and Deep Neural Networks. These algorithms are chosen for their popularity and effectiveness in solving classification problems.

This work investigates the performance of these algorithms in three types of classification: binary classification (2C), multiclass classification with 7 classes (7C), and multiclass classification with 15 classes (15C). The binary classification is performed to distinguish normal traffic from malicious traffic. The multiclass classification with 7 and 15 classes are performed to identify different types of attacks. The aim of this investigation is to identify the most effective model for detecting network-based attacks using the CSE-CICIDS2018 dataset [13], while maintaining high accuracy and f1-score throughout the different types of classifications: 2C, 7C, and 15C. The results show that Random

Forest and XGBoost algorithms outperform state-of-the-art intrusion detection systems in binary (2C) and multiclass classification (15C) in terms of accuracy and F1 score. The results also indicate that the imbalance of the dataset needs to be handled in the future in order to improve the performance of deep learning techniques. This is important because the imbalance in the dataset can cause the classifier to have a bias towards the majority class, leading to poor performance on minority classes.

The thesis is structured as follows: Chapter 2 provides background information on intrusion detection systems, and common machine and deep learning approaches. Chapter 3 reviews previous literature on intrusion detection systems and the classification models that have been used in the field. Chapter 4 presents the methodology used in this research, including the data preprocessing, feature selection, and evaluation of different classification models. Chapter 5 presents the evaluation results, discussing the performance of the different classification models for each type of classification. Finally, Chapter 6 provides conclusions and suggestions for future work.

Chapter 2

Background

In this chapter, a brief background is given about some of the main terminologies used in this paper to provide the reader with the necessary information and background knowledge to understand and evaluate the research.

2.1 Big Data

Big data is a term that refers to large volumes of data that businesses generate and collect from various sources. It refers to large and complex datasets that can be analyzed computationally to reveal patterns, trends, and associations, particularly relating to human behavior and interactions [38]. Big data is found in the digital traces we leave behind in our daily lives, such as clicks, likes, posts, location updates, and search queries, and in the photos we share on social media with metadata embedded within them. It is also a buzzword for a type of data analysis that can reveal insights about people based on their digital footprints and predictive analytics. Big data is characterized by velocity, variety, and volume, which refer to the rate at which new information is generated, the types of information to analyze, and the amount of information overall. In recent years, big data has advanced with new technologies such as cloud computing and predictive analytics, which have given rise to artificial intelligence, machine learning, and deep learning [32]. However, these technologies come with challenges, including the need for more computing power, access to raw data sets, privacy and ethical issues in big data analysis, and questions about creating fair and responsible datasets for training AI systems. The potential benefits of artificial intelligence and deep learning depend on access to large amounts of high-quality datasets that must be processed to produce predictions [14]. These techniques require fast computing power and access to stored information and rely on human supervision, including programming that understands human language for artificial intelligence, algorithms that understand how machines work for machine learning, and supervised feeding with vast amounts of pre-existing content for deep learning. Without these resources, capabilities would be limited.

2.2 Network Security

The use of the Internet and cloud computing has led to an increase in security breaches. These breaches can range from unwanted emails and website intrusions to infiltrations of company networks. Companies use defense mechanisms to protect their data from criminals who want to steal information or cause damage. Cybersecurity is not simple, as it consists of multiple components that make up an effective network security system, each playing a vital role in keeping hackers out and protecting sensitive information. Cyberattacks are becoming more sophisticated and widespread. Cybercriminals are always finding new ways to steal data and make money from it. Cybercrime can cost organizations a lot of money in lost revenue, downtime, and legal fees. It costs the global economy over 600$ billion every year. There have been numerous attacks in recent years, such as the attack on Target that exposed 40 million customers' credit card numbers, largely due to weak network security. Network security is the practice of protecting a computer network from unauthorized access, attacks, and other security threats [10]. It is essential for any modern organization, as networks are the backbone of communication, collaboration, and productivity. One challenge of network security is the constantly changing nature of security threats. As technology advances and the ways people use networks evolve, new vulnerabilities and attack methods emerge, requiring network security to be a dynamic and proactive discipline that adapts to new threats and evolves to stay ahead of attackers. The concept of defense in depth is a fundamental principle of network security, where multiple layers of security are put in place to protect a network, each providing a different level of protection. This helps prevent attackers from gaining access to a network, even if they breach one layer of security. Some common techniques and technologies used in network security are firewalls, intrusion detection and prevention systems, encryption, and authentication. User education and awareness is also an important aspect of network security [17]. Network users must be educated about the importance of security and how they can help protect the network, such as using strong passwords, not sharing passwords, and being cautious when opening email attachments or clicking on links. In conclusion, network security is a critical component of any modern organization. It requires a proactive and multi-layered approach to stay ahead of constantly evolving security threats. By implementing effective security measures, organizations can protect their networks and the sensitive information they contain [40].

2.3 Intrusion Detection Systems

Intrusion detection systems (IDS) are used for monitoring network activity and detecting potential security threats. They are a crucial component of any network security strategy because they can alert organizations to early warnings of attacks and other security incidents. There are two main types of intrusion detection systems: network-based IDS and host-based IDS. Network-based IDS, also known as NIDS, monitor network traffic and search for signs of potential security threats such as suspicious network traffic patterns, attempts to access prohibited network resources, and known attack signatures. Host-based IDS, on the other hand, monitor individual devices on a network, looking for signs of malware or other security threats [51]. To fully comprehend intrusion detection systems, one must be aware of three main components: sensors, analysis engine, and reporting. The analysis engine analyzes raw data from sensors and detects anomalies as well as patterns in behavior that could indicate unauthorized access attempts, while reporting helps detect malicious activity with visualization tools such as graphs or charts that monitor the number of successful connections over time. One of the primary advantages of intrusion detection systems is their ability to provide early warning of potential security incidents. By continuously monitoring network activity, IDS can alert administrators to potential threats before they can cause significant damage. This allows organizations to take timely and appropriate action to mitigate the threat and prevent harm to the network and its users. Another advantage of IDS is their ability to provide detailed information about security incidents. By analyzing network traffic and other data, IDS can provide administrators with valuable insights into the nature and scope of security threats. This information can be used to improve the organization's overall security posture and to prevent similar incidents from occurring in the future. However, there are also some disadvantages to using intrusion detection systems. One of the main challenges is the high volume of false positives that IDS can generate. They are designed to be sensitive, which can sometimes result in alerts being produced for harmless or benign activities. This can be frustrating for administrators and can lead to a loss of trust in the IDS system. Another disadvantage of IDS is their potential for evasion by attackers. IDS rely on known attack signatures and other static data, which can be bypassed by attackers using novel or customized attack methods. This can leave a network vulnerable to attack, even if an IDS is in place. In conclusion, intrusion detection systems are valuable tools for network security. They provide organizations with early warning of potential security threats and valuable insights into the nature and scope of security incidents. However, they are not a perfect solution and must be used as part of a comprehensive security strategy [39].

2.4 Machine Learning

Machine learning has become a key component of many businesses and academic institutions as artificial intelligence becomes more prevalent in society. It is also used in cybersecurity and fraud prevention and helps companies optimize their supply chains. It is a subfield of artificial intelligence focused on developing algorithms and models that can learn from data and improve performance over time [53]. This learning relies on the idea that systems can learn from experience and improve performance without explicit programming. Machine learning algorithms use statistical techniques to identify patterns in data and make predictions or decisions. These algorithms can be trained on large datasets and learn to make accurate and reliable predictions or decisions. There are various types of machine learning, including supervised learning, unsupervised learning, and reinforcement learning [9]. Supervised learning involves training a model on a labeled dataset where the correct output is provided for each input. Unsupervised learning involves training a model on an unlabeled dataset and requires the model to identify patterns and relationships in the data on its own. Reinforcement learning involves training a model to take actions in an environment to maximize a reward. The following subsections will provide a brief summary of common machine learning algorithms: Support Vector Machine, K-Nearest Neighbor, Decision Tree, and Random Forest [28].

2.4.1 Support Vector Machine

The support vector machine (SVM) algorithm is a powerful supervised learning method often used for classification and regression tasks. It seeks the hyperplane in high-dimensional space that maximally separates different classes. One advantage of SVM is its ability to handle high-dimensional data effectively. Through the kernel trick, which maps data into a higher-dimensional space, SVM can handle data with many features even if there are few training examples [21]. SVM is also flexible. By using a kernel function to transform data, SVM can be used with different types of data, including linear and nonlinear data, and adapt to a wide range of data and problem types. However, SVM also has some disadvantages. Its computational complexity can make it expensive, especially for large datasets, and it lacks interpretability as the decision boundary is determined by support vectors, making it hard to interpret the resulting model and understand how it makes predictions. Overall, SVM is a powerful tool for supervised learning tasks with advantages including its handling of high-dimensional data and flexibility. However, it also has disadvantages, including computational complexity and lack of interpretability [64].

2.4.2 K-Nearest Neighbor

The k-nearest neighbor (KNN) algorithm is a simple and effective method for classification tasks. It is based on the idea of identifying the k points in the training dataset that are closest to a given data point and using the majority class or average value of those points to make a prediction. One of its main advantages is its simplicity. It relies on a straightforward distance measure and majority voting, making it easy to understand and implement, even for those with little background in machine learning [63]. Another advantage of KNN is its flexibility. It does not make assumptions about the functional form of the data, allowing it to be used with different types of data, including linear and nonlinear data, and adapt to a wide range of data and problem types. There are also some disadvantages to using KNN. One challenge is the computational complexity of the algorithm. It involves calculating the distance between each data point, which can be computationally expensive, particularly for large datasets, making it impractical for some applications [2]. KNN's sensitivity to irrelevant or noisy features is another disadvantage. It uses all available features to calculate distances and can be affected by features that are irrelevant or have a high degree of noise, degrading the performance of the algorithm. Overall, the k-nearest neighbor algorithm is a simple and effective method for classification and regression tasks. It has many advantages, including its simplicity and flexibility. However, it also has some disadvantages, including its computational complexity and sensitivity to irrelevant and noisy features [29].

2.4.3 Decision Tree

The decision tree algorithm is a powerful and interpretable method for classification and regression tasks. It builds a tree-like model that makes predictions by following a series of decisions based on the values of the input features. One advantage of decision trees is their interpretability [42]. The tree structure allows for easy visualization of the model and the decisions it makes, making decision trees highly interpretable. Decision trees can also handle both numerical and categorical data. They can split data based on either type of feature, allowing them to be used with datasets that have a mix of numerical and categorical features. However, decision trees also have some disadvantages [33]. One challenge is the potential for overfitting. Decision trees can keep splitting data until they perfectly classify the training examples, leading to models that fit the noise in the data rather than the underlying pattern. This can result in poor performance on unseen data. Decision trees can also lack smoothness. They make predictions by following a series of binary splits, producing models that can be very piecewise and discontinuous. This can make it difficult to extrapolate the model's predictions to new data. Overall, the decision tree algorithm is a powerful and interpretable method for classification and regression tasks with advantages including interpretability and ability to handle different types of data. However, it also has disadvantages including the potential for overfitting and lack of smoothness [46].

2.4.4 Random Forest

The random forest algorithm is an ensemble method for classification tasks. It builds multiple decision trees on random subsets of the data and combines their predictions to produce a final prediction. One advantage of random forests is their ability to handle high-dimensional data effectively. Random forests can handle datasets with a large number of features, even if there are only a few training examples, because each decision tree is trained on a random subset of the data [8]. Random forests can also reduce the overfitting that can occur with decision trees. By averaging the predictions of multiple decision trees, they are less likely to fit the noise in the data and can produce more stable and generalizable models. However, there are also some disadvantages to using random forests. The computational complexity of the algorithm can make it expensive, especially for large datasets, and it can be difficult to interpret the resulting model and understand how it is making predictions because the final prediction is the average of many decision trees [62].

2.4.5 XGBoost

Predictive modeling, machine learning, data mining, and other disciplines have all made extensive use of the well-known machine learning method known as the XGBoost classifier. Extreme Gradient Boosting, or XGBoost, is a gradient-boosting implementation that aims to be fast and efficient . It was created by Tianqi Chen and has become very well known as a result of its outstanding performance on a variety of datasets [44]. The speed and effectiveness of the XGBoost classifier are two of its primary benefits. Large datasets may be handled, and it works well with both organized and unstructured data. Automatic missing value imputation, feature selection, and model parallelization are just a few of the built-in capabilities that make using XGBoost simple. It can also handle both classification and regression tasks, and it can do well on datasets that are imbalanced. However, using the XGBoost classifier has some drawbacks as well. It may be sensitive to the selection of hyperparameters, and it may require a long time to determine the best set of hyperparameters. Additionally, XGBoost may not be the ideal option for datasets with a small number of features or instances because it is more likely to overfit in these circumstances. Finally, the interpretability of the XGBoost classifier may not be as high as that of certain other algorithms, making it more challenging to comprehend how it generates its predictions [23].

2.5 Deep Learning

Artificial Intelligence is an area of research that strives to create systems that are capable of intelligent behavior by reasoning and learning from experience, just like humans do [52]. Deep learning methods are currently one of the most promising tools in AI, producing state-of-the-art results in various areas, including computer vision and natural language processing. In recent years, there's been a surge in interest in deep learning, which uses artificial neural networks to analyze and learn from large data sets with more complexity than previous machine learning methods. As technologies improve, deep learning is expected to lead to major breakthroughs that will ultimately shape our future [12]. Deep learning is a subset of machine learning that involves using large amounts of data to train artificial neural networks to perform specific tasks. Neural networks are algorithms designed to recognize patterns in data, and they are composed of many interconnected processing nodes, called "neurons." Deep learning networks are composed of multiple layers of these neurons, and they are able to learn and adapt as they are exposed to more data. This ability to learn from data allows deep learning algorithms to perform tasks such as image and speech recognition with high accuracy. Deep learning algorithms have been applied to a wide range of fields, including computer vision, natural language processing, and medical diagnosis due to their ability to learn from large amounts of data. The following subsections will give a brief summary of the most common deep learning algorithms: convolutional neural networks, artificial neural networks, deep neural networks, recurrent neural networks, and long short-term memory [54].

2.5.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a type of deep learning algorithm that is commonly used for image and video analysis. CNNs are designed to automatically learn and extract features from raw data, such as images or videos, and use these features to make predictions or decisions [6]. One of the main advantages of CNNs is their ability to learn spatial hierarchies of features. This means that CNNs can learn to recognize patterns at different levels of abstraction, from low-level features such as edges and textures, to high-level features such as objects and scenes. This hierarchical structure allows CNNs to perform well on a wide range of tasks, such as image classification, object detection, and face recognition. Another advantage of CNNs is that they are computationally efficient. CNNs use shared weights and local connections, which reduces the number of parameters that need to be learned and speeds up the training process [3]. This makes CNNs well-suited to applications that require real-time performance or that have limited computational resources. However, CNNs also have some disadvantages. One of the main limitations of CNNs is that they require a large amount of labeled training data to perform well. This can make it difficult to apply CNNs to new or unseen domains, or to tasks that have few labeled examples. Additionally, CNNs can be difficult to interpret and understand, which can make it challenging to debug or explain the decisions made by the model. Despite these limitations, CNNs have proven to be powerful and effective tools for a wide range of image and video analysis tasks. With the continued development of new algorithms and techniques, CNNs are likely to remain a key part of the machine learning landscape in the future [1].

2.5.2 Artificial Neural Networks

Artificial neural networks (ANNs) are a type of machine learning algorithm that is inspired by the structure and function of the human brain. ANNs are composed of many interconnected processing units, called neurons, which are organized into layers. The neurons in each layer receive input from the previous layer, process the input using a nonlinear activation function, and pass the output to the next layer [45]. One of the main advantages of ANNs is their ability to learn complex, nonlinear relationships in data. ANNs are able to automatically discover and extract features from raw data, and can learn to make predictions or decisions based on these features. This makes ANNs well-suited to a wide range of applications, including image and speech recognition, natural language processing, and predictive modeling [25]. Another advantage of ANNs is their ability to handle large, noisy, or incomplete datasets. Unlike other machine learning algorithms, which may require data to be pre-processed or cleaned, ANNs are able to learn from raw data and can handle a high degree of noise and missing values. This makes ANNs useful for applications that involve complex, real-world data. However, ANNs also have some disadvantages. One of the main limitations of ANNs is their high computational complexity, which can make them slow to train on large datasets. Additionally, ANNs can be difficult to interpret and understand, which can make it challenging to debug or explain the decisions made by the model. Besides, ANNs are sensitive to the choice of hyperparameters [58].

2.5.3 Deep Neural Networks

Deep neural networks (DNNs) are a type of artificial neural network (ANN) that is composed of many layers of interconnected processing units, called neurons. DNNs are trained using a large amount of labeled data and a supervised learning algorithm, such as backpropagation, which adjusts the weights of the connections between neurons to improve the performance of the network [49]. One of the main advantages of DNNs is their ability to learn complex, nonlinear relationships in data. DNNs are able to automatically discover and extract features from raw data, and can learn to make predictions or decisions based on these features. This makes DNNs well-suited to a wide range of applications, including image and speech recognition, natural language processing, and predictive modeling. Another advantage of DNNs is that they can learn hierarchical representations of data, with each layer of the network learning increasingly complex features of the input data. This allows DNNs to perform well on a wide range of tasks, and to learn useful features that can be used for other tasks. However, DNNs also have some disadvantages [41]. One of the main limitations of DNNs is their high computational complexity, which can make them slow to train on large datasets. Additionally, DNNs can be difficult to interpret and understand, which can make it challenging to debug or explain the decisions made by the model. Finally, DNNs are sensitive to the choice of hyperparameters, which can be difficult to optimize for a specific task or dataset. This can make it difficult to get good performance from DNNs without a significant amount of trial and error, and can also make it challenging to reproduce the results of DNNs trained by others. Despite these limitations, DNNs have proven to be powerful and effective tools for a wide range of tasks, and have achieved state-of-the-art performance on many benchmarks and real-world applications. With the continued development of new algorithms and techniques, DNNs are likely to remain a key part of the machine learning landscape in the future [15].

2.5.4 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a type of artificial neural network (ANN) that is designed to process sequential data, such as time series, natural language, or audio [22]. RNNs are composed of many interconnected processing units, called neurons, which are organized into layers. Unlike feedforward ANNs, which process input data only once, RNNs can process the same input multiple times, using the output of each time step as input to the next. This allows RNNs to capture temporal dependencies in the data and to make predictions or decisions based on the entire sequence of input. One of the main advantages of RNNs is their ability to model long-term dependencies in data. RNNs can learn to remember or forget information from earlier time steps, which allows them to capture patterns that span long periods of time. This makes RNNs well-suited to tasks such as language translation, speech recognition, and time series forecasting [61]. Another advantage of RNNs is their ability to handle variable-length input sequences. Unlike other ANNs, which typically require input data to be of a fixed size, RNNs can process input sequences of different lengths, and can adapt to the specific characteristics of the input data. This makes RNNs useful for applications that involve complex, real-world data. However, RNNs also have some disadvantages. One of the main limitations of RNNs is their susceptibility to vanishing and exploding gradients, which can make it difficult to train deep RNNs. Additionally, RNNs can be difficult to interpret and understand, which can make it challenging to debug or explain the decisions made by the model. Finally, RNNs can be computationally expensive, particularly for long input sequences or large datasets, which can make them difficult to use in real-time applications [16].

2.5.5 Long Short-Term Memory

The long short-term memory (LSTM) algorithm is a type of recurrent neural network (RNN) that is designed to overcome the limitations of traditional RNNs, such as the vanishing gradient problem and the inability to capture long-term dependencies in data. LSTMs are composed of many interconnected processing units, called neurons, which are organized into layers [47]. LSTMs use a specialized type of neuron, called a memory cell, which is able to retain information over long periods of time. This allows LSTMs to capture temporal dependencies in data and to make predictions or decisions based on the entire sequence of input. One of the main advantages of LSTMs is their ability to learn complex, nonlinear relationships in data. LSTMs are able to automatically discover and extract features from raw data, and can learn to make predictions or decisions based on these features. This makes LSTMs well-suited to a wide range of applications, including natural language processing, speech recognition, and time series forecasting. Another advantage of LSTMs is their ability to handle variable-length input sequences. Unlike other ANNs, which typically require input data to be of a fixed size, LSTMs can process input sequences of different lengths, and can adapt to the specific characteristics of the input data [26]. This makes LSTMs useful for applications that involve complex, real-world data. However, LSTMs also have some disadvantages. One of the main limitations of LSTMs is their high computational complexity, which can make them slow to train on large datasets. Additionally, LSTMs can be difficult to interpret and understand, which can make it challenging to debug or explain the decisions made by the model. Finally, LSTMs are sensitive to the choice of hyperparameters, which can be difficult to optimize for a specific task or dataset [16].

2.6 Evaluation Metrics

The performance of an anomaly-based intrusion detection system can be assessed using the commonly used evaluation metric known as the "f1 score" (IDS). The f1 score is determined as the harmonic mean of the precision and recall. Precision is a metric that quantifies the proportion of correctly predicted positive outcomes by a model, computed as the ratio of true positives to the total number of positive predictions. High precision implies a low rate of false positives, meaning that when the model predicts a positive outcome, it is more likely to be accurate. Recall, on the other hand, is a measure of the model's ability to detect all actual positive instances in the dataset. It is calculated by dividing the number of true positive predictions by the total number of actual positive instances. A high recall indicates a low rate of false negatives, meaning that the model is less likely to miss actual positive instances. The harmonic mean of precision and recall is used to calculate the f1 score, which is reported as a value between 0 and 1, with higher values representing greater performance. As it enables evaluation of both the false positive rate (precision) and the false negative rate (recall), the f1 score is highly useful for IDS systems. False positives and false negatives in the context of IDS refer to situations where the IDS wrongly recognized benign traffic as malicious and failed to identify malicious traffic, respectively. The f1 score enables the evaluation of the IDS in terms of both types of errors, both of which have major repercussions [11]. Below are the exact formulas to calculate f1-score 2.1, precision 2.2, recall 2.3, and accuracy 2.4.

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} \tag{2.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

True positive (TP): The number of instances correctly classified as positive.

True negative (TN): The number of instances correctly classified as negative.

False positive (FP): The number of instances incorrectly classified as positive.

False negative (FN): The number of instances incorrectly classified as negative.

Chapter 3

Related Work

This chapter includes a summary of the key findings and contributions of the previous studies, and discusses their implications for the current research. The previous studies use a variety of approaches for intrusion detection, which include machine learning and deep learning algorithms. These approaches will be discussed in terms of effectiveness in detecting intrusions. Besides, identifying the gaps in the existing literature and potential avenues for future research.

3.1 Machine Learning Approaches

To detect intrusions and provide intelligent services in the field of cyber-security, a variety of well-known machine learning classification algorithms are used in [5], including Bayesian Network, Naive Bayes classifier, Decision Tree, Random Decision Forest, Random Tree, Decision Table, and Artificial Neural Network. KDD cup 99 is the dataset used, and it classifies attacks into U2R, R2L, Dos, and Probe. IDS models based on Random Forest classifiers routinely outperform other classifiers at identifying intrusions. The Random Decision Forest in particular produces the highest outcomes in terms of accuracy (99%), precision, recall (93%), and f1-score (0.97). The Random Forest classifier first creates a number of decision trees, from which it derives a set of rules for the forest. A Random Forest Model generates more logic rules by taking into consideration the majority vote of these trees while creating the result because each tree in the model behaves differently as a machine learning classification technique. Therefore, the Random Forest model performs better in terms of accuracy, precision, recall, and f1-score. For future work, extending the cyber-security datasets and having a strategy to create a data-driven intrusion detection system for the community's automated security services.

The suggested system in [27] uses the CART method with the Gini index as a splitting criterion for pattern classification, correlation-based feature selection (CFS) is utilized to reduce dimensionality, and decision trees are used to classify attacks (DoS, R2L, Probe, U2R). Training and testing are done using the NSL-KDD dataset. Additionally, the suggested work is contrasted with other recently published strategies, and it is discovered that the proposed system for the NSL-KDD dataset has greater overall accuracy and detection rates or low false positive rates. As the size is reduced by one-third of the original data size, the feature reduction method lowers the time and space complexity. The suggested system performs better than other published work (j48, Naive Bayes, NB tree, multilayer perceptron, Lib SVM, Simple Cart) in terms of detection rate, overall accuracy (90.3), and false positive rate (9.7).

The effectiveness of machine learning approaches in identifying malicious activity and network threats have already been shown. Some of the machine learning topics that are now receiving the most interest include ensemble-based and semi-supervised learning techniques. However, merging these strategies has received comparatively little attention. This research [60] suggests a novel tri-training methodology combined with Adaboost algorithms for network anomaly identification. To produce the diversity using decision stumps for weak classifiers (both for continuous and categorical data), three separate Adaboost algorithms are used in place of the tri-bootstrap training samples. For each simulation, 30 iterations are conducted in order to get an average result. KDD-cup 99 dataset is used which categorizes attacks into DoS, R2L, Probe, and U2R. Simulations show that the proposed semi-supervised Adaboost method is repeatable and reliable across a range of run counts. Even with a minimal amount of labeled data in the training phase, it outperforms other cutting-edge learning algorithms. It has a strong balance between the detection rate (89.02%), the false-alarm rate (1.38%), precision rate (98.63%), f-measure (97.97%), as well as a very quick execution time (8.33 seconds). The problem of multi-class classification and attack detection will be the main topics of the suggested future work. Future research would include determining the ideal labeled-to-unlabeled data ratio.

The work presented in [24] looked into the issue of machine learning-based threat detection in network security. The stochastic gradient descent enhanced the K-means clustering technique, and therefore the Support vector machine was coupled with it. The dataset utilized was KDD Cup 99, and the attacks are categorized as DoS, R2L, Probe, and U2R. It was discovered that the method used in this study had an (87.1%) detection rate and a (3.1%) false alarm rate, and that its detection effects were clearly superior to those of both the SVM algorithm and a single K-means clustering algorithm. The dependability of the approach used in this investigation was demonstrated by the DoS detection rate, which was (94.5%). The low detection efficiencies for U2R and Probe may have been brought on by the sheer volume of samples. When compared to the old clustering method and the enhanced clustering algorithm, the improved algorithm had a greater detection rate and a lower false alarm rate, showing that the upgrade to the clustering algorithm was successful. The detection effect of the algorithm was then significantly higher than that of the two prior algorithms, according to a comparison of the K-means clustering algorithm, the SVM algorithm, and the algorithm used in this study. Finally, the algorithm had improved detection effects for Normal and DoS attacks from the standpoint of the detection impacts of various categories of network threats.

3.2 Deep Learning Approaches

Through the use of random forest in [56], features are eliminated recursively while the important value of each feature is computed. For classification, deep multilayer perceptrons (DMLP) are used. CICIDS2017 dataset is used which includes the following attack for classification: DDoS, Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, and Botnet. Using the acquired features, a Deep Multilayer Perceptron (DMLP) structure can identify intrusions with a 91 % accuracy rate. In relation to its initial size, the dataset has been cut in half by 95 %. For future work, a larger network traffic dataset that will be gathered using big data platforms like Hadoop and Spark will be used to create an IDS.

The performance of intrusion detection systems is enhanced by the integration of big data and deep learning techniques in [19]. For classification, Deep Feed-Forward Neural Networks (DNN) and two ensemble methods—Random Forest and Gradient Boosting Tree (GBT)—are used. UNSW NB15 and CICIDS2017 are the datasets used in this paper and include attacks such as Fuzzers, Analysis, Backdoors, DOS, Exploits, Generic, Reconnaissance, Shellcode, and worms. DOS, DDoS, Web-based, Brute force, Infiltration, Heart-bleed, Bot, and Scan. On the UNSW-NB15 dataset, the findings demonstrate very short prediction times and high accuracy levels with DNN for binary and multiclass classification (99.19 % and 97.04 %, respectively). Using the CICIDS2017 dataset, the GBT classifier had the best accuracy (99.99 %) for binary classification, and the DNN classifier had the best accuracy (99.57 %) for multiclass classification. It is not stated how well-distributed Apache Spark processing performs with different cluster node numbers. With better feature selection strategies, it will be examined how to increase the effectiveness of intrusion detection with feature selection utilizing homogeneity metric.

Modelling an intrusion detection system based on deep learning is the main aim of the following paper [59]. Using recurrent neural networks, a deep learning approach to intrusion detection (RNN-IDS). The effectiveness of the proposed model in binary and multiclass classification, as well as the effects of the number of neurons and different learning rates, were examined. The NSL-KDD dataset is used which categorizes attacks into 4 main categories: DoS, R2L, Probe, and U2R. The experimental results demonstrate that RNN-IDS performs better than conventional machine learning classification methods in both binary and multiclass classification, and that it is particularly suitable for constructing a classification model with high accuracy. For the KDDTrainC dataset, the studies demonstrate that RNN-IDS performs well with a good detection rate (83.28%) when given 100 epochs. The results were (68.55%) for the KDDTest-21 dataset and (99.81%) for the KDDTrainC dataset. The results of J48, Naive Bayesian, Random Forest, Multi-layer Perceptron, Support Vector Machine, and other classification algorithms are displayed by the authors. The artificial neural network algorithm also provides (81.2%), according to recent research on ANN algorithms used in the intrusion detection field. RNN-IDS model outperforms other binary classification algorithms in terms of performance. The performance achieves a higher accuracy rate and detection rate with a low false positive rate, notably under the job of multiclass classification on the NSL-KDD dataset, when compared to classic classification methods like J48, Naive Bayesian, and random forest. Future work should focus on decreasing training time utilizing GPU acceleration, avoiding exploding and vanishing gradients, and researching how well the LSTM and Bidirectional RNNs algorithms perform when used for intrusion detection.

The work presented in [7] uses various machine learning models to predict DDoS attacks at the application layer in real time. To improve the execution of the algorithms, Apache Spark, a distributed system, and a classification method were utilized. The outcomes of the big data method and how it performs in comparison to the non-big data approach were also contrasted. With the help of Spark ML libraries, Apache Spark is a large data tool that can quickly identify attacks. For the purpose of detecting DoS attacks, the Scikit ML library and big data framework Spark-ML library were used in conjunction with two machine learning techniques, Random Forest (RF) and Multi-Layer Perceptron (MLP). For training and testing, the application layer DDoS dataset from Kaggle was used. The MLP classifier utilizing the non-big data approach has a minimum accuracy of (99.05%). In contrast to the suggested technique, the maximum accuracy achieved with the RF classifier utilizing the big data approach was (99.94%) and the f1-score was (99.95%). The proposed model's minimal processing time employing an MLP classifier and big-data approach was 0.04 seconds. When compared to other existing models, the proposed models have great accuracy and quick processing times. The limitations were only two machine learning models were used, and the dataset has only two classes.

Security in the fog and the cloud is a current concern for every paradigm for managing, storing, or processing data. Attacks, once they happen, have severe and irreparable implications on the growth of cloud computing, IoT, and fog. In order to ensure the security of the fog, numerous security techniques and models have been suggested and/or put into practice. The proposed model in [4] classifies DoS, R2L, Probe, and U2R attacks from the NSL-KDD dataset using multi-layered recurrent neural networks that are intended to be used for Fog computing security that is very close to ending users and IoT devices. The model has excellent sensitivity to DoS attacks, which are one of the main forms of attacks that impede the growth of IoT networks, in addition to identifying other types of attacks in a competitive computational overhead since each record takes an average of 66 seconds to process. The proposed model can therefore function correctly and effectively in real-time situations. The findings show that the suggested cascaded multilayered filtering employing recursive neural networks generated equivalent performance in terms of detection rate (94.27%) and false positive rate (9.8%), f1-score (92.29%), as well as better results in terms of detection accuracy (92.18%) when compared to other approaches. The suggested model achieved nearly complete agreement, as evidenced by K and MCC coefficient values of (84.44%) and (84.36%), respectively. This provides the model's robustness against low-frequency attacks. Additionally, the inclusion of recurrence features has improved the model detection performance's near-perfect independence from the attacks' random occurrence, making it suitable for use in real-time IoT security applications.

The work presented in [50] is to forecast the precise actions that an adversary will take when launching an attack. Recurrent Neural Networks (RNNs) are used to categorize security event data collected from Symantec's intrusion prevention product (anonymized machine ID, timestamp, security event ID, event description, system actions, and other information) in order to predict future events on a machine based on previous observations. The first dataset of 3.4 billion security events was gathered over the course of 27 days by a commercial intrusion prevention system from a daily population of 740,000 devices. The second dataset is made up of 1.2 billion security events that were gathered between November 2017 and February 2018 on the eighth and the 23rd of each month, as well as the first three days of January. Results indicate that Tiresias is capable of accurately predicting the next event that would take place on a machine with a precision of up to (0.93) and achieve over (0.8) recall and f1-measure. These sporadic intrusion attempts may not have been appropriately predicted by the method. Tiresias surpasses the benchmark techniques, but 3-grams also outperform Markov Chains, and Markov Chains outperform the spectral learning approach. The fact that the 3-grams system performs the best among the baselines demonstrates the significance of sequence memory. 3-grams, however, do less well than Tiresias. This is because neural networks have the ability to filter out noise and have a longer-term memory, which are two of their key qualities. In all five days of testing, Tiresias' precision scores have been higher than 0.8, demonstrating a high degree of dependability. Furthermore, it is demonstrated that long-term memory, a major component of RNNs, is crucial for accurately forecasting security occurrences. Resulting from Tiresias using GPUs to train RNN models while the baseline methods rely on conventional CPUs, Tiresias is generally significantly faster to run, hence it is important to note that we did not compare the computation times of the different approaches. Tiresias only needs about 10 minutes for each epoch when employing GPUs, compared to the 3-gram implementation over a 10-day training period.

Traditional machine learning techniques mainly rely on feature engineering, whose feature extraction can be difficult and time-consuming. Therefore, using conventional machine learning techniques to identify attacks in real time is problematic. An end-to-end detection method is suggested for effectively detecting network threats in the following study [37]. Datasets from DARPA1998, KDD 1999, and CNTC 2017 Dataset from Webshell, CSIC-2010. These attacks comprised DoS, R2L, Probe, U2R, and webshell attacks on the web. HTTP datasets were the datasets used in this study. When tested on the DARPA1998 dataset, the approaches obtain accuracy levels of (99.36%) and (99.98 %) and f-measure of (99.38%) and (99.98%); these accuracy levels are on par with (or better than) those of state-of-the-art techniques. The suggested techniques are very effective and useful. In comparison to conventional machine learning models, the proposed models have additional parameters. As a result, model training is challenging and requires a certain set of abilities, and altering the parameters of the models is complicated. Second, the suggested approaches' interpretation is not the best. CNNs and RNNs are black boxes, therefore some researchers do not support deep networks. Thus, the interpretation of these models is the focus of the 570 future work proposals. Long sequences are difficult to train RNN models on. Traditional RNNs are not practical for classifying long texts, since they integrate a limited amount of prior information and are difficult to train.

The AI-SIEM system in [34] was created using a combination of FCNN, CNN, and LSTM artificial neural networks as well as event profiling for data preprocessing. The system focuses on separating genuine positive signals from false positive alerts, assisting security analysts in quickly responding to cyber threats such as DoS, R2L, Probe, U2R, online attacks, Botnet, infiltration, DDoS, and HeartBleed. NSLKDD and CICIDS2017 are the datasets used in this paper. The use of mechanisms as learning-based models for network intrusion detection is possible. The results of the performance tests utilizing two well-known benchmark datasets, such as NSLKDD and CICIDS2017, demonstrated that they were just as effective as traditional machine-learning models by achieving a maximum accuracy of (0.958 and 0.995) and an f-measure of (0.952 and 0.987). The presented algorithms, used in the AI-SIEM system, can therefore be used for learning-based network intrusion detection. The performance of overall accuracy is not as trustworthy as that of benchmark datasets when conventional learning-based approaches, which achieve a decent result through benchmark datasets, are used in the actual world. Despite the abundance of data and the absence of benchmark dataset features, as demonstrated by the ESX-2 result, the accuracy performance of the proposed three EP-ANN models was not noticeably worse. The accuracy of traditional approaches, however, had decreased from roughly (0.90) to (0.85). The primary focus should be on improving earlier threat forecasts through the use of several deep learning approaches to find long-term patterns in historical data in order to solve the evolving issue of cyberattacks.

Defenders want a more effective network detection strategy that benefits from the capacity to quickly adapt to new network behavior with the features suggested for network intrusion detection. This study [36] focused on classifying network threats (DoS, R2L, Probe, and U2R) using convolutional neural networks (CNNs) based on LeNet-5 to identify network intrusions using the KDD-cup 99 dataset. The experiment's findings indicate that for samples larger than 10,000, intrusion detection prediction accuracy increases to (99.65%).

In [18], an ensemble method that uses a meta-classifier (i.e., logistic regression) adhering to the stacking generalization principle, together with deep models like the Deep Neural Network (DNN) and Long Short-Term Memory (LSTM) is proposed. A Deep Sparse AutoEncoder (DSAE) is used for the feature engineering challenge in the initial step of data pre-processing. A stacking ensemble learning strategy is used for classification in the second phase. IoT-23, LITNET-2020, and NetML-2020 are the datasets used in this paper and include the following attacks for classification: IoT infected by malware, syn-flood, UDP-flood, smurf attack, red worm, reaper worm, land attack, ICMP flood, HTTP flood, fragmentation, blaster worm. When evaluated against pre-specified testing sets, it is implied that the suggested approach offers a significant improvement in terms of assessment metrics. In a nutshell, the suggested framework may solve the problem of supplying recent network traffic datasets and offer a respectable level of accuracy to identify anomalous behaviors in the target network. In terms of accuracy (0.997), F1-score (0.98), and MCC (0.99), the suggested technique beats LSTM, DNN, Random Forest, SVM, and MLP. Future network data processing will be accelerated and made more scalable through the use of cutting-edge computing techniques like Apache Spark. The method must also be verified for use in multi-class problem-solving.

The KDD1999 Cup dataset is used to classify DoS, R2L, Probe, and U2R attacks using SVM and KNN-ACO in [57]. Comparing the proposed approach to existing ones reveals greater precision and a reduced false alert rate. The proposed method's drawbacks include a high rate of false positive detection error, difficulty in resolving slow misbehavior, and expensive computation. After investigation, it was discovered that the total accuracy rate for the suggested technique is approximately (94.17%), whereas the accuracy rates for the existing methods, Back Propagation and Support Vector Machine, are (93%) and (83%), respectively. The overall false alarm rate of the suggested method is (5.82%), compared to (6.90%) for Back Propagation and (16.90%) for Support Vector Machine in other existing methods.

The security of IoT is at great risk from malware and software piracy attempts. It is suggested to use the TensorFlow deep neural network to detect software piracy through source code plagiarism in [55]. To filter the noisy data and further zoom the significance of each token in terms of source code plagiarism, tokenization and weighting feature approaches are utilized. The deep learning methodology is then applied to find instances of source code plagiarism. In addition, by visualizing colored images, the deep convolutional neural network is used to find dangerous viruses in IoT networks. The Google Code Jam (GCJ) dataset is used to categorize malware and piracy attacks on industrial IoT clouds. The experimental findings show that the proposed method for measuring IoT cybersecurity threats performs better in terms of classification than state-of-the-art techniques. Regarding texture feature mining utilizing malware visualizations, substantial computing cost is required. The ongoing generation, updating, and manipulation of malware makes identification more difficult. The maximum accuracy obtained by the proposed approach was (97.46%), GIST + SVM was (86.1%), LBP + SVM was (78.05%), and CLGM + SVM was (92.06%). While the highest F-measure obtained by the proposed method was (97.44%), GIST + SVM was (85.82%), LBP + SVM was (77.49%), and CLGM + SVM was (91.98%).

Previous studies in the field of intrusion detection systems have been found to have several limitations. One major limitation is the use of outdated datasets, which may not accurately reflect current threats and challenges in the field of cybersecurity. Additionally, many studies only use a subset of the available data, which can limit the scope and generalizability of the findings. Another limitation is that most previous studies have focused on binary classification, achieving high accuracy rates, but without taking into account the different types of attacks. This narrows down the range of identification of attacks and the generalizability of the results. These limitations highlight the need for more recent and comprehensive studies that can address current cybersecurity threats, and more advanced classification techniques that can improve the identification of different types of attacks and better generalization of the results.

Chapter 4

Methodology

This chapter provides a description of the tools and methods utilized to develop IntrusionHunter. Fig 4.1 gives an overview of the full process that occurs to detect attacks accurately and efficiently. The full process consists of three main phases: data cleaning (1), feature selection (2), and classification (3).

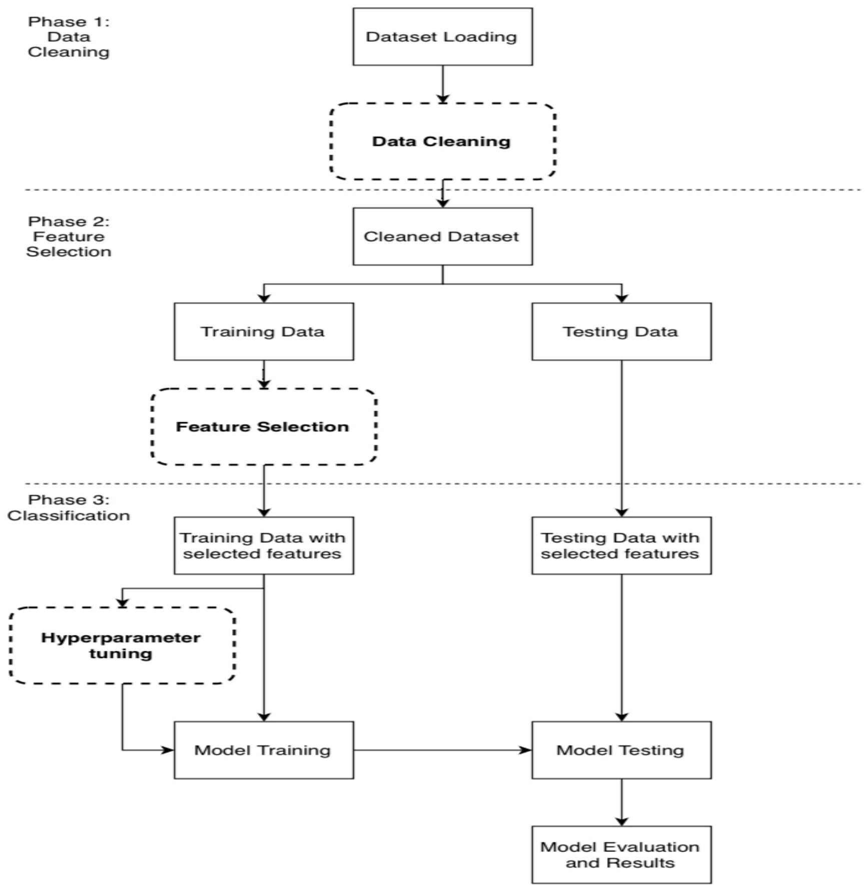

Figure 4.1: The three phases for intrusion detection by IntrusionHunter

The following sections and subsections comprise a description of these phases, along with the experimental dataset, the methodologies, and techniques used to analyze the data, so that the study process is visible and understandable for readers while also allowing them to duplicate and validate the findings.

4.1 Dataset

A popular resource for academics and professionals working on the creation and assessment of intrusion detection systems is the CSE-CIC-IDS2018 dataset [13] (IDS). It was created by the Canadian Institute for Cybersecurity (CIC) and is made up of more than 10 million network traffic examples that were gathered over the course of nine months in 2018. The dataset is a good resource for evaluating how well IDS systems are able to identify various threats because it contains a wide variety of malicious and benign traffic, including 14 attack classes that could be categorized into the following attack scenarios: Brute-force, Botnet, DoS, DDoS, infiltration, and Web attacks. The dataset's realism and diversity are among its important characteristics. The dataset is indicative of real-world network traffic because it contains traffic from a variety of devices and networks, including servers, workstations, and mobile devices. The CSE-CIC-IDS2018 dataset [13] is significant not only for its size and complexity but also for its extensive variety of features. Every instance of network traffic is represented by a collection of attributes, which include information about the source and destination IP addresses, port numbers, protocol, and other specifics. These features can be used to evaluate and train IDS systems' accuracy and false positive rate performance. The dataset additionally includes labels that describe each instance's traffic type (benign or malicious), allowing for the evaluation of IDS systems' recall and precision. The dataset is a priceless tool for academics and professionals involved in the study of cybersecurity and intrusion detection.

Dataset Exploration	
Number of Rows	16,233,002
Number of Features	84
Number of Classes	15
Collected By	Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC)
Collection Date	2018

Table 4.1: IDS Dataset Features

Table 4.1 gives a brief summary of the dataset and its main features, while table 4.2 includes the instances count of each class and categorizes each attack into its main attack category.

Category	Class	Value Count
Benign	Benign	13484708
DDoS	DDoS attack-HOIC	686012
DDoS	DDoS attacks-LOIC-HTTP	576191
DoS	DoS attacks-Hulk	461912
Botnet	Bot	286191
BruteForce	FTP-BruteForce	193360
BruteForce	SSH-Bruteforce	187589
Infilteration	Infilteration	161934
DoS	DoS attacks-SlowHTTPTest	139890
DoS	DoS attacks-GoldenEye	41508
DoS	DoS attacks-Slowloris	10990
DDoS	DDOS attack-LOIC-UDP	1730
Web Attacks	Brute Force -Web	611
Web Attacks	Brute Force -XSS	230
Web Attacks	SQL Injection	87

Table 4.2: Classes of the IDS Dataset

4.2 Data Preprocessing

This section describes a preprocessing phase that is done on the CSE-CIC-IDS2018 dataset [13] before it is used for training and analysis. The dataset is composed of ten separate CSV files, each containing network traffic records for a single day. The preprocessing step includes cleaning the data (phase 1), by dropping unnecessary columns, removing missing values, duplicates and handling string values such as "infinity" that can't be parsed by pandas method. Moreover, feature selection is performed using Random Forest (phase 2) to select the features with the highest importances. The process is done to make sure the data is clean and suitable for training, and to avoid issues such as skewed statistical results, inaccurate predictions, and poor model performance.

4.2.1 Data Cleaning (Phase 1)

In phase 1, numerous preprocessing steps are undertaken before the analysis of the dataset to make sure the data is clean and suitable for training. In its raw format, the dataset is composed of ten separate CSV files, each of which contains the network traffic records for a single day of operation and is named with the date the traffic was recorded. The existence of several column names as values in the dataset was discovered during dataset exploration. Visual inspection of the input file reveals that the headers are present throughout the file, embedded within the rows of raw data. This shows that a single file was made by merging several CSV files together, which led to the headers being duplicated in the process. In order to resolve this problem, all the data frame's header-containing columns are deleted, and then the data frame is exported to a temporary CSV file so that it can be read again with the appropriate column datatypes.

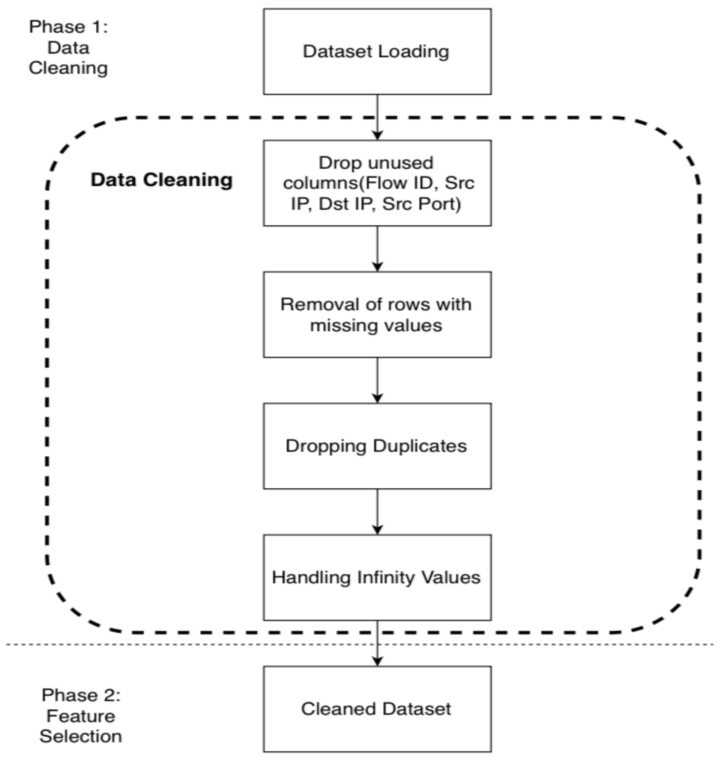
Phase 1: Data Cleaning
Dataset Loading
Data Cleaning
Drop unused columns(Flow ID, Src IP, Dst IP, Src Port)
Removal of rows with missing values
Dropping Duplicates
Handling Infinity Values
Phase 2: Feature Selection
Cleaned Dataset

Figure 4.2: Data Cleaning Steps

As shown in Fig 4.2, the data goes through the above process to ensure it is clean and efficient for usage in the next steps. First, the following columns (Dst IP, Flow ID, Src IP, Src Port) are present in one of the CSV files, but not the rest. These columns are dropped as they are not needed and will create null values when concatenated with the rest of the CSV files. Moreover, all missing values are dropped, as they can lead to a number of issues, including skewed statistical results, inaccurate predictions, and poor model performance. Besides, all duplicates are removed, as they can act as noise and lead to difficulty in understanding the patterns between the data. Last but not least, another visual inspection of the file indicates that a column's numerous rows contain the string "Infinity." This value cannot be appropriately parsed by the pandas (read_csv()) method, since it only accepts the strings "inf/-inf" as the legitimate representation of infinity. To solve this issue, the mean value of the column including "Infinity" is used in place of "Infinity" everywhere.

4.2.2 Feature Selection (Phase 2)

In phase 2, a subset of features must be chosen from a dataset, in order to employ a subset of features in a machine learning model. The model's performance may be enhanced, and the risk of overfitting may be decreased, making it a crucial phase in the modeling process. Random forest, a machine learning algorithm that may be used to find the key features in a dataset, is one popular technique for feature selection. The algorithm is initially trained on the dataset using several decision trees in order to employ random forest for feature selection. The technique assesses the importance of each feature by calculating the decrease in impurity caused by splits using the feature. Each tree is built using a random subset of the features. Averaging the scores across all the trees yields the final importance score for each feature. A final model is more likely to be chosen if features with a higher significance score are more beneficial for predicting the target variable.

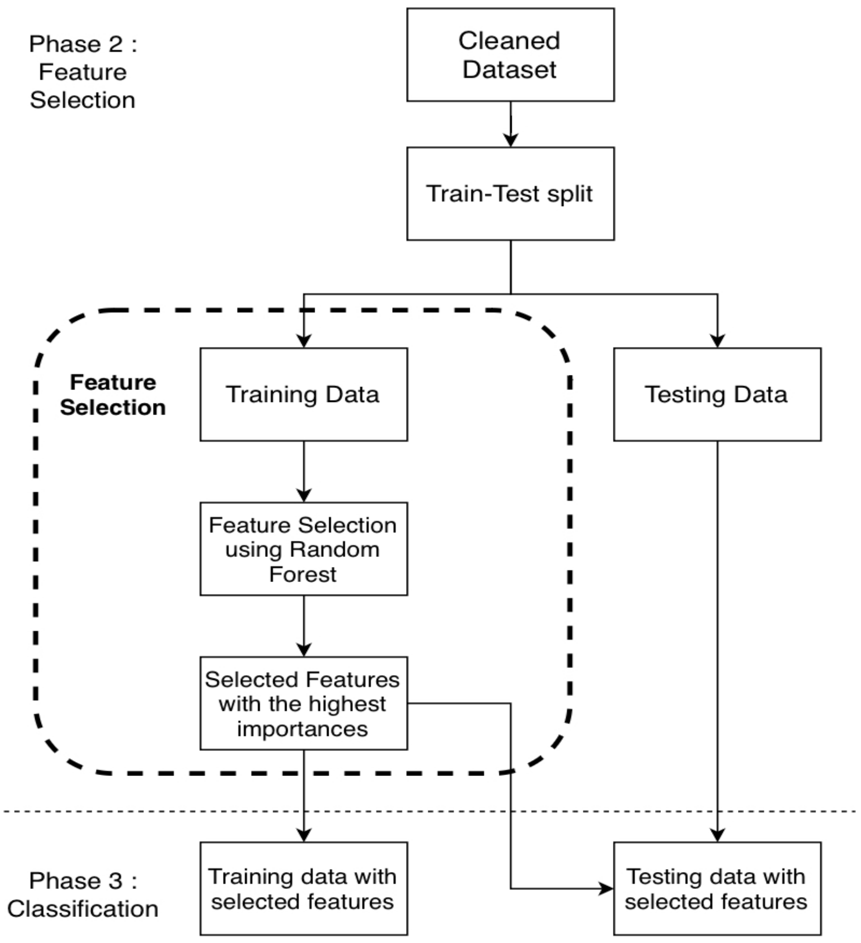
Phase 2 : Feature Selection
Cleaned Dataset
Train-Test split
Feature Selection
Training Data
Testing Data
Feature Selection using Random Forest
Selected Features with the highest importances
Phase 3 : Classification
Training data with selected features
Testing data with selected features

Figure 4.3: Feature Selection Phase

Fig 4.3 explains the process of feature selection process. The process is applied to the training data after splitting the dataset into training and testing data, and it results in 25 features being selected out of the 80 features that the dataset includes.

4.3 Classes

After preprocessing and selecting relevant features with feature selection, different classification algorithms and methods can be applied to the data to perform this task. The dataset is subjected to three different classification methods: binary classification (2C), as well as multi-class classification with seven (7C) and fifteen classes (15C). The following subsections will focus on explaining each classification type's importance.

4.3.1 Binary Classification (2 Classes)

In an anomaly based intrusion detection system, it is important to accurately classify network traffic as either legitimate or malicious in order to protect against potential security threats. Binary classification is a useful approach for distinguishing between benign and malicious traffic, as it allows the system to identify and classify potentially malicious activity in a simple and straightforward way. Binary classification is particularly important in an anomaly-based intrusion detection system because it allows the system to focus on identifying unusual or suspicious behavior that may indicate a security threat. By accurately classifying network traffic as either benign or malicious, the system can take appropriate action to prevent or mitigate the impact of potential attacks. In addition to its effectiveness in identifying malicious activity, binary classification also has several other benefits when used in an anomaly-based intrusion detection system. For example, it is a relatively simple machine learning task that can be easily implemented and understood. Binary classification algorithms also often perform well on large datasets and can achieve high accuracy rates, making them well-suited for handling large volumes of network traffic in real time.

4.3.2 Multiclass Classification (7 Classes)

Multiclass classification was done in two ways in this study. One way to do this is by grouping similar attacks together into clusters and classifying them using a multi-class classification system. Using a multi-class classification system with seven classifications, similar attacks can be grouped together into one of the following classes: Benign, Brute-force, Botnet, DoS, DDoS, infiltration, and Web attacks. This can be helpful for identifying the most prevalent attack types in the dataset and for grouping together similar attacks. By accurately classifying different types of malicious activity, an intrusion detection system can more effectively identify and mitigate the risk of potential security threats. For example, by knowing the types of attacks that are most prevalent in the dataset, the system can prioritize its efforts to defend against those types of attacks. Additionally, by grouping similar attacks together, the system can more easily identify patterns and trends in malicious activity, which can help to improve its overall effectiveness in detecting and preventing attacks.

4.3.3 Multiclass Classification (15 Classes)

Another way to do multiclass classification was by classifying each attack on its own (15 classes), which means performing the classification on the original attacks in the dataset without grouping any of them together. By classifying each attack on its own, the system can more accurately and identify particular attack types and understand their features. This can be helpful for improving the system's ability to detect and prevent specific types of attacks and for identifying patterns and trends in malicious activity.

4.4 Classification Models (Phase 3)

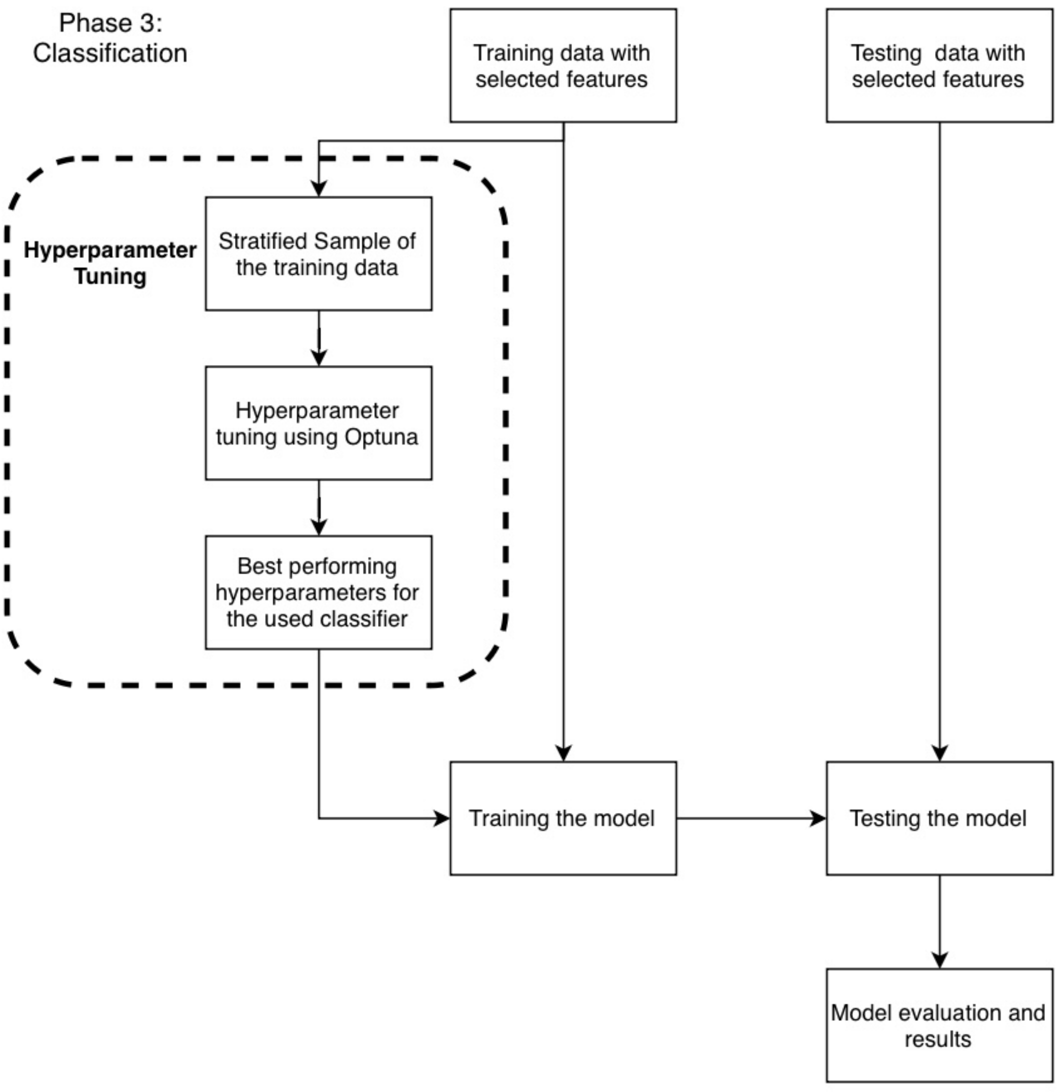
Phase 3:
Classification
Training data with selected features
Testing data with selected features
Hyperparameter Tuning
Stratified Sample of the training data
Hyperparameter tuning using Optuna
Best performing hyperparameters for the used classifier
Training the model
Testing the model
Model evaluation and results

Figure 4.4: Model Classification

In phase 3, different classification algorithms are used to detect intrusions. This section provides an overview of the four primary classifiers used to carry out anomaly-based intrusion detection: random forest, xgboost, convolutional neural networks (CNNs), and deep neural networks (DNNs). Each classifier will be thoroughly explained, including with its operation, salient characteristics, and any unique parameters or implementation-specific settings. Chapter 2 gives a brief background about the classifier in terms of their advantages and disadvantages. Each model performs three types of classifications, as described in section 4.3. Models will be referred to with unique names. The unique names consist of the abbreviation of the model along with the number of classes that the model classifies. For example, a random forest classifier that performs binary classification will be referred to as (RF-2C). Fig 4.4 gives a brief overview of the code flow until testing the performance of the proposed intrusion detection system. After the data is split into training and testing data and the process of feature selection is applied, the hyperparameter tuning process is performed. Hyperparameter tuning is the process of optimizing the hyperparameters of a machine learning model to improve its performance. One way to do this is by using a library such as Optuna, which performs a search for the best hyperparameters by trying out different combinations of hyperparameters and evaluating the performance of the model for each combination. In this case, Optuna is being used to tune the model by performing 50 trials with different hyperparameters. This can help to improve the accuracy and efficiency of the model by finding the hyperparameters that result in the best performance. The hyperparameter tuning phase starts by taking a stratified sample of the dataset. This is done for two reasons. Firstly, the dataset is excessively large, so applying hyperparameter tuning on the entire dataset would take too much time. Secondly, the classes in the dataset are imbalanced, so stratified sampling is used instead of random sampling. After hyperparameter tuning is performed, the model is trained with the best performing hyperparameters along with using the whole training data. Last but not least, the model is tested with the testing data and the results are evaluated.

4.4.1 Random Forest (2C, 7C, 15C)

The first classifier used in our study was the random forest algorithm. Random forests are a popular choice for classification tasks due to their ability to handle large datasets and their good performance on a wide range of tasks. In addition, random forests are able to handle imbalanced datasets, which is often the case in anomaly based intrusion detection systems (IDS) as anomalies are typically rare events. The random forest algorithm is able to handle imbalanced datasets by constructing multiple decision trees, each trained on a different subset of the data. This ensures that the classifier is able to learn from all the available data, rather than being biased towards the majority class. Another advantage of using random forests for IDS is that they do not require the data to be standardized before classification, making them a strong choice for use as a classifier in our study.

Parameter	RF-2C	RF-7C	RF-15C
number of estimators	150	170	200

Table 4.3: Random Forest Parameters

Table 4.3 provides the parameters used in the random forest classifier. These parameters were returned after hyperparameter tuning was performed. It was noticed that when increasing the number of parameters to be tuned, the performance degraded. Therefore, n_estimators was the parameter chosen as it has a huge influence on the classifier's performance.

4.4.2 XGBoost (2C, 7C, 15C)

The XGBoost classifier, a potent gradient boosting method that has been demonstrated to perform well on a range of classification tasks, was utilized in addition to the random forest classifier. Building a sequence of decision trees with each tree seeking to boost the performance of the preceding one is how the XGBoost classifier operates. It has been demonstrated that this iterative approach outperforms many existing classification methods in terms of both speed and accuracy, allowing the classifier to generate increasingly accurate predictions.

Parameter	XGB-2C	XGB-7C	XGB-15C
number of estimators	1900	2000	2200
number of jobs	-1	-1	-1
tree method	'hist'	'hist'	'hist'

Table 4.4: XGBoost Parameters

In table 4.4 the parameters of the XGBoost classifier are included. The number of estimators parameter was chosen by the hyperparameter tuning library, where each time a range was entered, the maximum was chosen. The n_jobs parameter specifies the number of CPUs to use when training a model. For example, in the scikit-learn library, the n_jobs parameter can be used to specify the number of CPU cores to use when fitting a model using the fit method. Setting n_jobs to -1 will cause the model to be trained using all available CPU cores, which can further speed up the training process but may also consume more system resources. The tree_method parameter specifies the method to use for constructing the decision trees that make up the gradient boosting model. One of the options for tree_method is 'hist', which stands for "histogram-based". When tree_method is set to 'hist', XGBoost uses an algorithm based on histograms to construct the decision trees. This method is particularly efficient when dealing with high-dimensional data and can lead to faster training times than other tree construction methods, such as the traditional 'exact' method.

4.4.3 CNN (2C, 7C, 15C)

Deep learning models known as convolutional neural networks (CNNs) are frequently employed in computer vision applications, including object detection and image classification. As they can detect patterns and features in data that are indicative of normal activity, they are especially well-suited for anomaly-based intrusion detection systems. An anomaly is signalled and can be recognized as an incursion when an input data point deviates from the learned patterns. CNNs are an effective technique for real-time anomaly detection because they can learn these patterns from training data and generalize to new, unseen data. Before training the data using CNN, standardization was performed first. Convolutional neural networks (CNNs) can be affected by the scaling of the input features, where features with high values may have more influence than those with lower values. This can result in the model being skewed towards these higher-valued features and make it harder to optimize. By standardizing the data, the optimization process can be improved and the model's sensitivity to the input feature's scaling can be reduced.

Parameter	CNN-2C	CNN-7C	CNN-15C
Number of Conv1D layer	4	4	6
Number of MaxPooling layer	2	2	3
Number of Dropout layer	2	2	3
Number of BatchNormalization layer	2	2	3
Activation Function	'relu'	'relu'	'relu'
Optimizer	adam	adam	adam
Batch Size	4096	256	4096
Epochs	50	50	50

Table 4.5: CNN Layers

As shown in table 4.5, a summary of the parameters used in three different convolutional neural network (CNN) models is provided, labeled as "CNN-2C", "CNN-7C", and "CNN-15C". The table includes the number of Conv1D layers, MaxPooling layers, Dropout layers, BatchNormalization layers, activation function, optimizer, batch size, and number of epochs for each of the three models. The CNN-2C model uses 4 Conv1D layers, 2 MaxPooling layers, 2 Dropout layers, 2 BatchNormalization layers, 'relu' activation function, 'adam' optimizer, batch size 4096 and 50 epochs. The CNN-7C model uses 4 Conv1D layers, 2 MaxPooling layers, 2 Dropout layers, 2 BatchNormalization layers, 'relu' activation function, 'adam' optimizer, batch size 256 and 50 epochs. The CNN-15C model uses 6 Conv1D layers, 3 MaxPooling layers, 3 Dropout layers, 3 BatchNormalization layers, 'relu' activation function, 'adam' optimizer, batch size 4096 and 50 epochs.

4.4.4 DNN (2C, 7C, 15C)

As a final classification method in our IDS, deep neural networks (DNNs) are employed. DNNs are a form of artificial neural network that are made up of many layers of interconnected nodes. They can learn from new data by modifying the weights of the connections between the nodes. DNNs are widely employed in numerous applications and have been demonstrated to be exceptionally effective at classification jobs. A procedure known as hyperparameter tuning is used to enhance the performance of the classifiers. The parameters known as hyperparameters control how classifiers behave and can significantly affect performance. Both the performance and the behavior of the classifiers were enhanced by tuning their hyperparameters using a library named Optuna and some manual tuning. Standardization was applied to the training data, as deep neural networks are also sensitive to the scale of the input features¿ Besides, standardizing the data can also make the training process more efficient and lead to better performance of the model.

Parameter	DNN-2C	DNN-7C	DNN-15C
Number of Dense layer	6	7	10
Number of Dropout layer	6	7	10
Activation Function	’relu’	’relu’	’relu’
Optimizer	adam	adam	adam
Batch Size	4096	4096	4096
Epochs	50	50	50

Table 4.6: DNN Layers

Table 4.6 shows the parameters of three deep neural network models (DNN-2C, DNN-7C, DNN-15C) that are used for classifying network intrusions. These models use dense layers as the main building blocks, with the number of dense layers increasing from 2 to 7 and finally to 15 as we move from DNN-2C to DNN-15C. The activation function used in all three models is ’relu’. The models are optimized using the ’adam’ optimizer and are trained using a batch size of 4096 samples for 50 epochs. The number of dropout layers in each model is equal to the number of dense layers. Dropout layers are used to reduce overfitting by randomly setting some of the activations to zero during training.

Chapter 5

Evaluation and Results

In this chapter, the results of the evaluation of the performance of the proposed anomaly based intrusion detection system (IDS) models, along with comparison to state-of-the-art IDS models on the same dataset, are presented. The dataset consists of both binary and multi-class classification tasks, and the models are evaluated using a variety of metrics 2.6, including accuracy and F1 score. The tables provided contain the accuracy and F1 score of each model, with the F1 score being the weighted version, as it is more appropriate for imbalanced datasets like the one used in the study. The weighted F1 score takes into account the relative frequencies of different classes in the data, whereas the macro F1 score does not. Each classification type has its own table comparing the performance of different approaches. The goal is to demonstrate the effectiveness of the proposed models in detecting anomalies and to identify any strengths and weaknesses compared to the state-of-the-art models.

5.1 Binary Classification

Table 5.1 represents the results of binary classification of several anomaly based intrusion detection systems (IDS). The studies include CNN-2C, RF-2C, XGB-2C, DNN-2C, RF-2C-Botnet, XGB-2C-Botnet, Kanimozhi and Prem Jacob [30], Praneeth [43], Seth, Singh, and Chahal [48]. The results show that the highest accuracy and F1-measure for classifying the whole dataset is achieved by RF-2C, with 99.332% and 99.328%. The approach with the second-highest accuracy and F1-measure is XGB-2C with 99.036% and 99.041%. RF-2C-Botnet and XGB-2C-Botnet were done so that we can compare between the performance of Kanimozhi and Prem Jacob's approach [30], where they classify botnet attacks only. As a result, both RF-2C-Botnet and XGB-2C-Botnet outperform Kanimozhi and Prem Jacob's model [30], with accuracies of 99.999% and 99.911% and F1-measures of 99.996% and 99.996%. The IDS with the lowest accuracy and F1-measure is DNN-2C, while the second lowest is Seth, Singh, and Chahal's model [48]. It should be noted that the IDS by Praneeth [43] uses only a subset of the dataset.

Study	Accuracy	F1-Measure	Approach	Notes
CNN-2C	97.997%	97.939%	CNN	
RF-2C	**99.332%**	**99.328%**	Random Forest	
XGB-2C	99.036%	99.041%	XGBoost	
DNN-2C	94.667%	94.234%	DNN	
RF-2C-Botnet	**99.999%**	**99.996%**	Random Forest	the IDS classifies Botnet attacks only
XGB-2C-Botnet	99.911%	99.996%	XGBoost	the IDS classifies Botnet attacks only
Kanimozhi and Prem Jacob [30]	99.97%	99.91%	ANN	the IDS classifies Botnet attacks only
Praneeth [43]	99.57%	98%	DNN	the IDS uses only a subset of the dataset
Seth, Singh, and Chahal [48]	97.73%	97.73%	LightGBM	

Table 5.1: Binary Classification Results

5.2 Multiclass Classification (7 Classes)

Study	Accuracy	F1-Measure	Approach
CNN-7C	97.814%	97.206%	CNN
RF-7C	**99.240%**	**99.135%**	Random Forest
XGB-7C	98.949%	98.626%	XGBoost
DNN-7C	94.244%	93.215%	DNN
Lin, Ye, and Xu [35]	96.2%	85%	LSTM
Karatas, Demir, and Sahingoz [31]	**95.49%**	**99.7%**	Adaboost and gradient boosting

Table 5.2: Multiclass Classification (7 Classes) Results

Table 5.2 presents the results of six different anomaly-based intrusion detection systems (IDSs) applied to classify network intrusions into seven classes. The CNN-7C, RF-7C, XGB-7C, and DNN-7C IDSs use the CNN, Random Forest, XGBoost, and DNN approaches, respectively. The table also includes the results of two additional IDSs proposed by Lin, Ye, and Xu [35] and Karatas, Demir, and Sahingoz [31], which use the LSTM and Adaboost with gradient boosting approaches, respectively. The CNN-7C IDS has an accuracy of 97.814%, while the RF-7C IDS has the highest accuracy at 99.240%. The F1-measure, is also the highest for the RF-7C IDS at 99.135%. The XGB-7C and DNN-7C IDSs have lower F1-measures compared to their accuracy values. The LSTM-based IDS proposed by Lin, Ye, and Xu [35] has a relatively high accuracy of 96.2%, but a lower F1-measure of 85%. The Adaboost and gradient boosting IDS proposed by Karatas, Demir, and Sahingoz [31] has a high F1-measure of 99.7%, but a lower accuracy of 95.49%. Overall, among the seven-class IDSs, the RF-7C IDS appears to have the best balance of accuracy and F1-measure.

5.3 Multiclass Classification (15 Classes)

Table 5.3 represents the results of several anomaly based intrusion detection systems for multiclass classification with 15 classes. The models used include CNN, Random Forest, XGBoost, and DNN. RF-15C has the highest accuracy with 99.239% and the highest F1-measure with 99.124%. DNN-15C has the lowest accuracy with 95.152% and the lowest F1-measure with 94.077%. The other models have accuracy and F1-measure scores that fall in between these two extremes.

Study	Accuracy	F1-Measure	Approach
CNN-15C	97.313%	96.643%	CNN
RF-15C	**99.239%**	**99.124%**	Random Forest
XGB-15C	98.951%	98.628%	XGBoost
DNN-15C	95.152%	94.077%	DNN
Ferrag [20]	97.376%	-	CNN

Table 5.3: Multiclass Classification (15 Classes) Results

Finally, the study by Ferrag [20] uses a CNN model and achieves an accuracy of 97.376% and did not state the F1-measure. Overall, the CNN, Random Forest and XGBoost models perform well in terms of accuracy, but RF-15C and XGB-15C have the best F1-measure. The DNN model has a moderate performance, with relatively lower accuracy and F1-measure compared to the other models.

5.4 Discussion

The results of the three classification tasks show that the performance of the different anomaly-based intrusion detection systems (IDSs) varies depending on the type of classification and the approach used. For binary classification (Table 5.1), the IDSs using Random Forest (RF-2C) and XGBoost (XGB-2C) achieved the highest accuracy and F1-measure values, with 99.332% and 99.328% respectively. These models performed better than the other models, such as the IDS using DNN (DNN-2C) which had the lowest accuracy and F1-measure values. Additionally, when looking at botnet attacks only, both RF-2C-Botnet and XGB-2C-Botnet performed better than Kanimozhi and Prem Jacob's model [30]. For multiclass classification with 7 classes (Table 5.2), the IDS using Random Forest (RF-7C) achieved the highest accuracy and F1-measure values, with 99.240% and 99.135% respectively. The IDS using XGBoost (XGB-7C) and DNN (DNN-7C) had lower F1-measures compared to their accuracy values. Additionally, the LSTM-based IDS proposed by Lin, Ye, and Xu [35] had a relatively high accuracy but a lower F1-measure, and the Adaboost and gradient boosting IDS proposed by Karatas, Demir, and Sahingoz [31] had a high F1-measure but a lower accuracy. The results of the multiclass classification with 15 classes (Table 5.3) show that the Random Forest model (RF-15C) achieved the highest accuracy and F1-measure, with 99.239% and 99.124%, respectively. This indicates that the RF-15C model is able to correctly classify the majority of the instances in the dataset and also achieve a high level of precision and recall. On the other hand, the DNN model (DNN-15C) had the lowest performance in terms of accuracy and F1-measure, with 95.152% and 94.077% respectively. This suggests that the DNN-15C model may have difficulty in correctly classifying instances in this particular dataset. The results of the other models fall in between these two extremes, with CNN-15C, XGB-15C and Ferrag study [20] achieving good accuracy and F1-measure scores. In conclusion, the results of the three classification tasks show that the performance of IDSs depends on the type of classification and the approach used. The RF and XGBoost approaches tend to perform well in terms of accuracy and F1-measure, while the CNN and DNN approaches may not perform as well. It is worth noting that the IDSs by Kanimozhi and Prem Jacob [30], Praneeth [43], and Ferrag [20] have specific limitations that could have affected their performance. Further research is needed to improve the performance of IDSs and to develop approaches that can effectively classify network intrusions into different classes.

Chapter 6

Conclusion and Future Work

In this chapter, the study's main conclusions will be examined, plans for future work on anomaly-based intrusion detection, and the limitations of the classification models that were employed. Some ideas for prospective future research projects that could build on the findings of our investigation will be suggested.

6.1 Conclusion

In this paper, the CSE-CICIDS2018 dataset [13] was used to evaluate the performance of the presented IDS (IntrusionHunter). Before the classification tasks, data cleaning was performed to ensure that the data was of high quality and ready for analysis. This included removing missing or invalid values, and handling imbalanced classes if necessary. In addition to data cleaning, feature selection was also performed using the random forest approach. The IDSs were then trained and tested using four different approaches: random forest, XGBoost, deep neural network, and convolutional neural network. These approaches were chosen due to their popularity and effectiveness in classification tasks. The IDSs were evaluated using three different classification tasks: binary, multiclass classification with 7 classes, and multiclass classification with 15 classes. The results of these classification tasks are discussed in the previous sections. Overall, the results of the three classification tasks show that the performance of the Random Forest IDSs exceeds the other approaches. The Random Forest (RF) and XGBoost approaches have demonstrated superior performance in terms of accuracy and F1-measure, surpassing the state-of-the-art intrusion detection systems (IDSs) in binary classification (2C) and multiclass classification (15C) tasks. Additionally, they have also exhibited favorable results in the multiclass classification (7C) task. On the other hand, the Convolutional Neural Network (CNN) and Deep Neural Network (DNN) approaches did not exhibit the same level of performance. This may be attributed to the imbalance present in the dataset, which resulted in lower accuracy and F1-score values for the CNN and DNN models, unlike Random Forest and XGBoost which have been shown to be robust to imbalanced datasets.

6.2 Future Work

One of the main limitations of the study is the limited size of the dataset used. While the dataset contained numerous records, it may not have been representative of all the different types of attacks that can occur in a real-world scenario. This could have affected the performance of the classification models and limited the ability to generalize the results to other datasets. Another limitation is the fact that the performance of the classification models was only evaluated on a single dataset. It would be useful to test the models on additional datasets to see if the results are consistent across different types of data. Additionally, the fact that the dataset used was imbalanced, could have led to an over-representation of certain types of attacks, and thus affected the performance of the IDSs, particularly the CNN and DNN models. In future work, it would be important to address the imbalance issue, either by oversampling the minority class or using techniques such as cost-sensitive learning. Additionally, it would be valuable to investigate the training and testing time of each approach, as this can provide insight into the suitability of each model for real-time detection. Furthermore, in future work, it would be interesting to evaluate the performance of the classification models on additional datasets and to compare their performance across different types of data to ensure the generalizability of the results. There are several directions that could be pursued in future work to build on the results of the study. One possibility would be to evaluate the performance of the classification models on a larger and more diverse dataset. This could help to improve the generalizability of the results and provide a more comprehensive assessment of the models' performance. Moreover, doing a comparison between each models' training and testing time will give more insight about what is the best machine learning/deep learning model that is suitable for real time detection. Another direction for future work could be to explore other classification models that may be more suitable for anomaly based intrusion detection. For example, there are many different types of neural networks that could be explored, including recurrent neural networks and long short-term memory networks. These models may be able to capture more complex patterns in the data and improve the performance of the anomaly detection system. Finally, it would be interesting to investigate ways to integrate the results of multiple classification models into a single anomaly detection system. This could potentially improve the overall performance of the system by leveraging the strengths of different models and minimizing their limitations.

List of Figures

List of Tables

Bibliography

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, 2014.

[2] Ihsan A. Abu Amra and Ashraf Y. A. Maghari. Students performance prediction using knn and naïve bayesian. In *2017 8th International Conference on Information Technology (ICIT)*, pages 909–913, 2017.

[3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[4] Muder Almi'ani, Alia Abughazleh, Amer Al-rahayfeh, Saleh Atiewi, and Abdul Razaque. Deep recurrent neural network for iot intrusion detection system. *Simulation Modelling Practice and Theory*, 101:102031, 11 2019.

[5] Hamed Alqahtani, Iqbal H. Sarker, Asra Kalim, Syed Md Minhaz Hossain, Sheikh Ikhlaq, and Sohrab Hossain. Cyber intrusion detection using machine learning classification techniques. In Nirbhay Chaubey, Satyen Parikh, and Kiran Amin, editors, *Computing Science, Communication and Security*, Communications in Computer and Information Science, pages 121–131, United States, 2020. Springer, Springer Nature. 1st International Conference on Computing Science, Communication and Security, COMS2 2020 ; Conference date: 26-03-2020 Through 27-03-2020.

[6] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8(1):1–74, 2021.

[7] Mazhar Awan, Umar Farooq, Hafiz Babar, Awais Yasin, Haitham Nobanee, Muzammil Hussain, Owais Hakeem, and Azlan Zain. Real-time ddos attack detection system using big data approach. *Sustainability*, 13:10743, 09 2021.

[8] S. Bernard, S. Adam, and L. Heutte. Using random forests for handwritten digit recognition. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 1043–1047, 2007.

[9] Rohan Bhardwaj, Ankita R. Nambiar, and Debojyoti Dutta. A study of machine learning in healthcare. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 236–241, 2017.

[10] Jack L. Burbank. Security in cognitive radio networks: The required evolution in approaches to wireless network security. In *2008 3rd International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom 2008)*, pages 1–7, 2008.

[11] A.A. Cardenas, J.S. Baras, and K. Seamon. A framework for the evaluation of intrusion detection systems. In *2006 IEEE Symposium on Security and Privacy (SP'06)*, pages 15 pp.–77, 2006.

[12] Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, Moustafa Alzantot, Federico Cerutti, Mani Srivastava, Alun Preece, Simon Julier, Raghuveer M. Rao, Troy D. Kelley, Dave Braines, Murat Sensoy, Christopher J. Willis, and Prudhvi Gurram. Interpretability of deep learning models: A survey of results. In *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–6, 2017.

[13] Canadian Institute for Cybersecurity (CIC) Communications Security Establishment (CSE). Cse-cicids2018 dataset. `https://www.unb.ca/cic/datasets/ids-2018.html`, 2018.

[14] Andrea De Mauro, Marco Greco, and Michele Grimaldi. What is big data? a consensual definition and a review of key research topics. In *AIP conference proceedings*, volume 1644, pages 97–104. American Institute of Physics, 2015.

[15] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603, 2013.

[16] Robert DiPietro and Gregory D Hager. Deep learning: Rnns and lstm. In *Handbook of medical image computing and computer assisted intervention*, pages 503–519. Elsevier, 2020.

[17] P.W. Dowd and J.T. McHenry. Network security: it's time to take it seriously. *Computer*, 31(9):24–28, 1998.

[18] Vibekananda Dutta, Michał Choraś, Marek Pawlicki, and Rafał Kozik. A deep learning ensemble for network anomaly and cyber-attack detection. *Sensors*, 20:4583, 08 2020.

[19] Osama Faker and Erdogan Dogdu. Intrusion detection using big data and deep learning techniques. In *Proceedings of the 2019 ACM Southeast Conference*, ACM SE '19, page 86–93, New York, NY, USA, 2019. Association for Computing Machinery.

[20] Mohamed Amine Ferrag, Leandros Maglaras, Sotiris Moschoyiannis, and Helge Janicke. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50:102419, 2020.

[21] V. Franc and V. Hlavac. Multi-class support vector machine. In *2002 International Conference on Pattern Recognition*, volume 2, pages 236–239 vol.2, 2002.

[22] C. Lee Giles, Gary M. Kuhn, and Ronald J. Williams. Dynamic recurrent neural networks: Theory and applications. *IEEE Transactions on Neural Networks*, 5(2):153–156, 1994.

[23] Aashish Gupta, Shilpa Sharma, Shubham Goyal, and Mamoon Rashid. Novel xgboost tuned machine learning model for software bug prediction. In *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, pages 376–380, 2020.

[24] Jinbao He, Jie Yang, Kangjian Ren, Wenjing Zhang, and Guiquan Li. Network security threat detection under big data by using machine learning. *Int. J. Netw. Secur.*, 21:768–773, 2019.

[25] J.J. Hopfield. Artificial neural networks. *IEEE Circuits and Devices Magazine*, 4(5):3–10, 1988.

[26] Yuxiu Hua, Zhifeng Zhao, Rongpeng Li, Xianfu Chen, Zhiming Liu, and Honggang Zhang. Deep learning with long short-term memory for time series prediction. *IEEE Communications Magazine*, 57(6):114–119, 2019.

[27] Bhupendra Ingre, Anamika Yadav, and Atul Soni. Decision tree based intrusion detection system for nsl-kdd dataset. 03 2017.

[28] Alistair E. W. Johnson, Mohammad M. Ghassemi, Shamim Nemati, Katherine E. Niehaus, David A. Clifton, and Gari D. Clifford. Machine learning and decision support in critical care. *Proceedings of the IEEE*, 104(2):444–466, 2016.

[29] P. Jonsson and C. Wohlin. An evaluation of k-nearest neighbour imputation using likert data. In *10th International Symposium on Software Metrics, 2004. Proceedings.*, pages 108–118, 2004.

[30] V Kanimozhi and T Prem Jacob. Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset cse-cic-ids2018 using cloud computing. In *2019 international conference on communication and signal processing (ICCSP)*, pages 0033–0036. IEEE, 2019.

[31] Gozde Karatas, Onder Demir, and Ozgur Koray Sahingoz. Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset. *IEEE Access*, 8:32150–32162, 2020.

[32] Avita Katal, Mohammad Wazid, and R. H. Goudar. Big data: Issues, challenges, tools and good practices. In *2013 Sixth International Conference on Contemporary Computing (IC3)*, pages 404–409, 2013.

[33] Manish Kumar, M. Hanumanthappa, and T. V. Suresh Kumar. Intrusion detection system using decision tree algorithm. In *2012 IEEE 14th International Conference on Communication Technology*, pages 629–634, 2012.

[34] Jonghoon Lee, Jonghyun Kim, Ikkyun Kim, and Kijun Han. Cyber threat detection based on artificial neural networks using event profiles. *IEEE Access*, 7:165607–165626, 2019.

[35] Peng Lin, Kejiang Ye, and Cheng-Zhong Xu. Dynamic network anomaly detection system by using deep learning techniques. In *International conference on cloud computing*, pages 161–176. Springer, 2019.

[36] Wen-Hui Lin, Hsiao-Chung Lin, Ping Wang, Bao-Hua Wu, and Jeng-Ying Tsai. Using convolutional neural networks to network intrusion detection for cyber threats. In *2018 IEEE International Conference on Applied System Invention (ICASI)*, pages 1107–1110, 2018.

[37] Hongyu Liu, Bo Lang, Ming Liu, and Hanbing Yan. Cnn and rnn based payload classification methods for attack detection. *Knowledge-Based Systems*, 163, 09 2018.

[38] Sam Madden. From databases to big data. *IEEE Internet Computing*, 16(3):4–6, 2012.

[39] Ningrinla Marchang, Raja Datta, and Sajal K. Das. A novel approach for efficient usage of intrusion detection system in mobile ad hoc networks. *IEEE Transactions on Vehicular Technology*, 66(2):1684–1695, 2017.

[40] G.A. Marin. Network security basics. *IEEE Security Privacy*, 3(6):68–72, 2005.

[41] Ali Bou Nassif, Ismail Shahin, Imtinan Attili, Mohammad Azzeh, and Khaled Shaalan. Speech recognition using deep neural networks: A systematic review. *IEEE Access*, 7:19143–19165, 2019.

[42] Xie Niuniu and Liu Yuxun. Notice of retraction: Review of decision trees. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 5, pages 105–109, 2010.

[43] Vipparthy Praneeth, Kontham Raja Kumar, and Nagarjuna Karyemsetty. Security: intrusion prevention system using deep learning on the internet of vehicles. *International Journal of Safety and Security Engineering*, 11(3):231–237, 2021.

[44] Jenny Price, Tatsuya Yamazaki, Kazuya Fujihara, and Hirohito Sone. Xgboost: Interpretable machine learning approach in medicine. In *2022 5th World Symposium on Communication Engineering (WSCE)*, pages 109–113, 2022.

[45] Sunitha R, R. Sreerama Kumar, and Abraham T. Mathew. Online static security assessment module using artificial neural networks. *IEEE Transactions on Power Systems*, 28(4):4328–4335, 2013.

[46] A. Ran and J. Kuusela. Design decision trees. In *Proceedings of the 8th International Workshop on Software Specification and Design*, pages 172–175, 1996.

[47] Tara N. Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584, 2015.

[48] Sugandh Seth, Gurvinder Singh, and Kuljit Kaur Chahal. A novel time efficient learning-based approach for smart intrusion detection system. *Journal of Big Data*, 8(1):1–28, 2021.

[49] Poonam Sharma and Akansha Singh. Era of deep neural networks: A review. In *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–5, 2017.

[50] Yun Shen, Enrico Mariconti, Pierre-Antoine Vervier, and Gianluca Stringhini. Tiresias: Predicting security events through deep learning. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[51] J.S. Sherif and T.G. Dearmond. Intrusion detection: systems and models. In *Proceedings. Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 115–133, 2002.

[52] Ajay Shrestha and Ausif Mahmood. Review of deep learning algorithms and architectures. *IEEE Access*, 7:53040–53065, 2019.

[53] Osvaldo Simeone. A very brief introduction to machine learning with applications to communication systems. *IEEE Transactions on Cognitive Communications and Networking*, 4(4):648–664, 2018.

[54] S. Reema Sree, S.B. Vyshnavi, and N. Jayapandian. Real-world application of machine learning and deep learning. In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 1069–1073, 2019.

[55] Farhan Ullah, Hamad Naeem, Sohail Jabbar, Shehzad Khalid, Muhammad Ahsan Latif, Fadi Al-turjman, and Leonardo Mostarda. Cyber security threats detection in internet of things using deep learning approach. *IEEE Access*, 7:124379–124389, 2019.

[56] Serpil Ustebay, Zeynep Turgut, and Muhammed Ali Aydin. Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier. In *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pages 71–76, 2018.

[57] Satyendra Vishwakarma, Vivek Sharma, and Ankita Tiwari. An intrusion detection system using knn-aco algorithm. *International Journal of Computer Applications*, 171:18–23, 08 2017.

[58] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[59] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5:21954–21961, 2017.

[60] Yali Yuan, Georgios Kaklamanos, and Dieter Hogrefe. A novel semi-supervised adaboost technique for network anomaly detection. pages 111–114, 11 2016.

[61] Huaguang Zhang, Zhanshan Wang, and Derong Liu. A comprehensive review of stability analysis of continuous-time recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 25(7):1229–1262, 2014.

[62] J. Zhang and M. Zulkernine. A hybrid network intrusion detection technique using random forests. In *First International Conference on Availability, Reliability and Security (ARES'06)*, pages 8 pp.–269, 2006.

[63] Min-Ling Zhang and Zhi-Hua Zhou. A k-nearest neighbor based algorithm for multi-label classification. In *2005 IEEE International Conference on Granular Computing*, volume 2, pages 718–721 Vol. 2, 2005.

[64] Shaoyan Zhang and Hong Qiao. Face recognition with support vector machine. In *IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003*, volume 2, pages 726–730 vol.2, 2003.